U2: SISTEMAS OPERATIVOS.

ELEMENTOS, ESTRUCTURA Y FUNCIONES GENERALES

ÍNDICE

1. INTRODUCCIÓN.

El sistema operativo es el software que hace que el equipo informático funcione. Sin él, el hardware no entraría en funcionamiento y no podríamos ejecutar nuestros programas. Se puede decir que nos comunicamos con el ordenador y que este funciona gracias al sistema operativo.

2. CONCEPTO DE SISTEMA OPERATIVO.

Podemos definir sistema operativo (**SO**) como un conjunto de programas que se inician al arrancar el ordenador y cuya función principal es desvincular al usuario de las características hardware de su equipo y facilitarle así la ejecución de otros programas, es decir, simplificar al usuario el uso del ordenador.

Las principales funciones de un sistema operativo son las siguientes:

- Gestionar eficientemente los recursos hardware y software del sistema informático.
- Desvincular al usuario de las particularidades del hardware de su equipo, proporcionándole una interfaz adecuada con la que trabajar.
- Controlar y administrar la ejecución de programas.
- Controlar y administrar el sistema de archivos.
- Detectar e intentar solucionar los errores que se puedan producir.

El usuario se comunica con el ordenador a través de las **interfaces** de usuario. Existen distintos tipos de interfaces, entre las que cabe diferenciar entre interfaces gráficas y en modo de texto.

Inicialmente, los sistemas operativos como el CP/M, VMS, MS-DOS, UNIX, … ofrecían interfaces en modo texto. Posteriormente empezaron a desarrollarse interfaces gráficas, inicialmente como una aplicación de esos sistemas operativos, como es el caso de Windows con respecto a MS-DOS, hasta que se crearon verdaderos sistemas operativos que utilizan interfaces gráficas, como Windows, Linux, Mac OS, aunque siempre es posible en estos sistemas operativos utilizar una interfaz en modo texto si fuera preciso.

3. EVOLUCIÓN HISTÓRICA.

Los primeros sistemas operativos se denominaban monolíticos. La característica fundamental de estos sistemas operativos es que su software básico era prácticamente imposible de modificar una vez creado e instalado en un sistema informático. Cuando los diseñadores del propio sistema operativo, o los usuarios por necesidades específicas, querían introducir modificaciones en él, la labor era realmente complicada, ya que se tenía que reconfigurar todo el SO. A veces eras más práctico rediseñar por completo el SO antes que modificar uno ya existente.

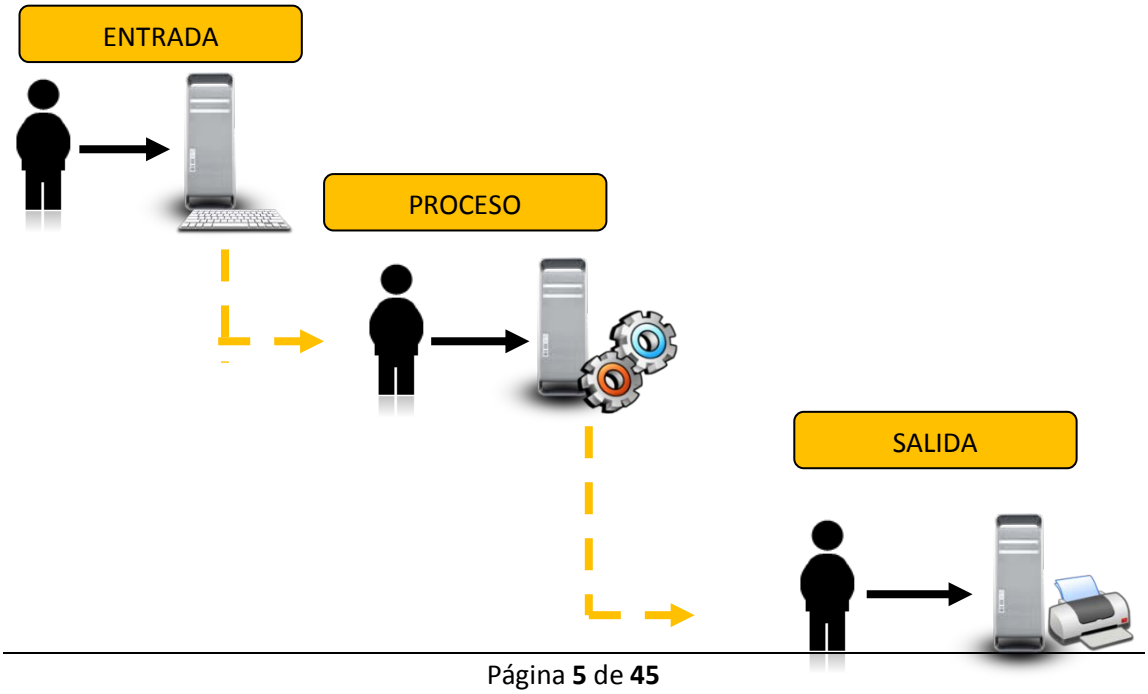
Para ver cómo han evolucionado los sistemas operativos a lo largo de la historia desde la aparición del primero de ellos, tenemos que tener muy presente las arquitecturas de los ordenadores, es decir, la evolución del hardware sobre el que se instalan.

Históricamente se ha hablado de cinco generaciones de ordenadores, quedando definidas las características de cada una de ellas por los componentes hardware de los sistemas informáticos que los componen. Hardware y SO evolucionan conjuntamente y nunca por separado. Si se diseñan sistemas operativos más potentes es debido a que el hardware sobre el que van a funcionar también lo es, y a la inversa, se diseña un hardware más potente y rápido debido a que los sistemas operativos necesitan cada vez mayores prestaciones hardware.

En general, podemos hablar de varias generaciones de sistemas operativos, relacionándolos siempre con la evolución del hardware.

- **PRIMERA GENERACIÓN (1945-1955)**. Se utilizaban **válvulas de vacío** (antiguas resistencia electrónicas). Estas computadoras, que no ordenadores, eran máquinas programadas en lenguaje máquina puro (lenguaje de muy bajo nivel). Eran de gran tamaño, elevado consumo de energía y muy lentas. Las operaciones se reducían a simples cálculos matemáticos.
  No disponían de sistema operativo, empleaban la tarjeta perforada para almacenar información.
  El sistema operativo solo permitía trabajar de forma estrictamente secuencial (IBM 704).

- **SEGUNDA GENERACIÓN (1955-1964).** Aparición de los **transistores**, que se introducen dentro de la arquitectura de las computadoras. Desaparecen las válvulas de vacío, por lo que las computadoras se hacen más pequeñas, baratas, consumen menos y despiden menos calor. Ganaron en potencia, rapidez y fiabilidad.
  Comenzaron a emplearse lenguajes de alto nivel como Cobol, Algol o Fortran. También memorias de núcleo de ferrita, cintas y tambores magnéticos para almacenar la información.
  La actividad se reparte entre dos ordenadores, uno principal que se encarga del cálculo y otro auxiliar que se encarga de la entrada y salida de datos.(IBM 1401)



- **TERCERA GENERACIÓN (1964-1974)**. Aparición de los **circuitos integrados**. Se reduce considerablemente el tamaño y consumo de energía de los ordenadores. Son más baratos y más rápidos y generan menos calor.
  En esta época evolucionó el software de forma considerable, sobre todo en los sistemas operativos, en los que se incluyó la multiprogramación y el tiempo real. Surge el concepto de memoria virtual, que optimiza la memoria principal.
  También evolucionaron considerablemente las unidades de almacenamiento, y aparecieron los discos magnéticos. Comenzaron a utilizarse memorias de semiconductores.
  Es de destacar el **IBM 370** como máquina capaz de realizar cualquier tipo de cálculo.

- **CUARTA GENERACIÓN (1974-1983)**. Aparecen los **microprocesadores** (toda la CPU en un solo circuito integrado). Es la generación de los ordenadores personales. Se perfeccionan las unidades de almacenamiento utilizándose el disquete o disco flexible. Se crearon las redes de ordenadores para la transmisión de datos.

- **QUINTA GENERACIÓN (Desde 1983).** En 1983, Japón lanzó el llamado Programa de la Quinta Generación de Computadoras, con los objetivos explícitos de producir máquinas capaces de comunicarse en un lenguaje más cotidiano y no a través de códigos o lenguajes de control especializados. **Inteligencia Artificial** (Se crean sistemas que pueden aprender con la experiencia). Aparece el **multiprocesador**: computadores con varios procesadores trabajando en paralelo.

4. TIPOS DE SISTEMAS OPERATIVOS.

Podemos hacer diversas clasificaciones de sistemas operativos atendiendo a ciertos criterios:

- Según tiempo de respuesta.
- Según número de usuarios.
- Según número de procesos.
- Según número de procesadores.
- Según trabajo en red.

4.1. SEGÚN TIEMPO DE RESPUESTA.

El tiempo de respuesta es el que transcurre desde que un proceso de un usuario llega al sistema hasta que el usuario obtenga una respuesta o unos resultados. Según este criterio, existe la siguiente clasificación de sistemas operativos:

- **TIEMPO REAL**.
  En estos tipos de sistemas operativos, los procesos requieren un tiempo de respuesta muy bajo, es decir, inmediato. Estos sistemas se utilizan en campos donde un tiempo de respuesta alto implicaría graves consecuencias, como por ejemplo, el tráfico aéreo, en procesos industriales, en sistemas espaciales, en sistemas médicos de monitorización de pacientes críticos, en sistemas bancarios,… y en general en todos los casos donde el tiempo de respuesta sea crucial y repercuta en grandes pérdidas económicas o de vidas humanas.

- **TIEMPO COMPARTIDO**.
  En estos tipos de sistemas operativos, cada proceso utilizará ciclos de la UCP hasta que finalice.

4.2. SEGÚN NÚMERO DE USUARIOS.

Atendiendo al número de usuarios que pueden utilizar los recursos del sistema simultáneamente, podemos dividir los sistemas operativos en:

- **MONOUSUARIO (o monopuesto)**.
  Cuando un solo usuario trabaja con un ordenador. En este sistema todos los dispositivos de hardware están a disposición de ese usuario y no pueden ser utilizados por nadie más hasta que éste no finalice su sesión.
- **MULTIUSUARIO.**
  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. También pueden compartir el acceso a una misma base de datos instalada en el ordenador principal, etc. Recordemos de qué dos formas, los diferentes usuarios pueden utilizar el ordenador principal: mediante terminales, teclado y monitor, o gracias a ordenadores cliente conectados al ordenador servidor mediante el hardware necesario.
  Este tipo de sistemas operativos se caracterizan porque varios usuarios que hagan uso del mismo ordenador, podrán hacer o no las mismas cosas, tendrán acceso o no a los mismo programas, podrán acceder o no a los soportes de almacenamiento externo, etc. Se les denomina multiusuario ya que en un mismo ordenador puede estar configurado de forma diferente para cada usuario que lo utilice.

4.3. SEGÚN NÚMERO DE PROCESOS.

Esta clasificación se hace atendiendo al número de programas que puede ejecutar simultáneamente el ordenador.

- **MONOPROGRAMACIÓN O MONOTAREA.**
  Sólo permite que el usuario ejecute un solo programa cada vez. De esta forma, los recursos están dedicados al programa hasta que finalice su ejecución. DOS o Windows 9X son sistemas operativos claramente monotarea, ya que además de no saber utilizar más de un microprocesador, el hardware que están utilizando al ejecutar un programa está a disposición de ese programa y de ningún otro.
  El que un SO sea monotarea no implica que no pueda ser multiusuario, es decir, varios usuarios pueden intentar ejecutar sus programas en el mismo ordenador, pero de forma sucesiva. Secuencialmente cada usuario esperará a que su proceso se ejecute tras haber finalizado el proceso anterior.

- **MULTIPROGRAMACIÓN O MULTITAREA.**
  En este caso, la cosa es bien distinta. Este tipo de SO puede ejecutar varios programas o procesos concurrentemente (simultáneamente). Esta circunstancia sólo se da en aquellos casos en los que el ordenador o sistema informático cuente con más de un microprocesador. Si el SO sólo tiene un microprocesador, aunque sea multitarea, es evidente que sólo podrá realizar un proceso cada vez. Si el SO se instala en un sistema informático que solamente cuenta con un procesador, la UCP compartirá el tiempo de uso del procesador entre los diferentes programas a ejecutar.
  De esta forma, todos los procesos necesitarán individualmente más tiempo para ejecutarse, pero en comparación con el diseño monotarea, el tiempo medio de espera será mucho menor.
  Son SO multitarea la mayoría de los que son multiusuario y en red, por no decir todos. Tengamos siempre presente que para esta clasificación, no importa el número de procesadores que tenga el ordenador o sistema informático, pero sí es importante recordar que la multitarea real sólo existe en ordenadores con más de un procesador.

4.4. SEGÚN NÚMERO DE PROCESADORES.

Esta clasificación depende del número de procesadores que el SO sea capaz de gestionar. Su clasificación es la siguiente:

- **MONOPROCESADOR.**
  Sólo pueden trabajar con un procesador. Todos los trabajos a realizar pasarán por él.
  Cualquier SO que se instale en un ordenador con un solo procesador siempre será monotarea. Lo que ocurre es que el hecho de que pueda ejecutar varios programas a la vez le confiere la catalogación de multitarea, pero la realidad es que solamente atenderá a un proceso en un instante concreto. Es cierto que si son varios los usuarios ejecutando sus diferentes programas en un ordenador con un solo procesador, este se puede catalogar como multitarea, teniendo en cuanta que los ciclos de UCP se repartirán equitativamente, o no entre los diferentes procesos.

- **MULTIPROCESADOR.**
  El ordenador cuenta con dos o más procesadores. Así determinados SO pueden aprovechar las ventajas de este tipo de hardware.
  Hay dos formas de utilizar los diferentes procesadores por parte del SO:
  a) **Multiproceso simétrico (SMP).** El SO utilizará la potencia de los procesadores de igual forma. Así, irá utilizando poco a poco los dos o más procesadores con los que cuente el sistema de forma simultánea.
  b) **Multiproceso asimétrico (SMP).** El SO reparte las tareas que está realizando a cada procesador con lo que cuenta en el sistema informático. Determinados procesos siempre los realizará un solo procesador y, en el caso de ser dos los procesadores de los que disponga el sistema, el otro procesador solamente se utilizará para realizar procesos o programas de usuario. En este caso, puede ser que un procesador esté siempre trabajando y el otro, en ocasiones, esté totalmente parado.

4.5. SEGÚN TRABAJO EN RED.

Dependiendo de la forma en que el sistema operativo puede trabajar y utilizar los recursos de la red, podemos clasificar los sistemas operativos en:

- **CENTRALIZADOS.**
  El equipo informático no comparte ningún recurso ni utiliza recursos de otros ordenadores por la red. Se utiliza como una máquina aislada que no está en red.

- **EN RED.**
  Los sistemas operativos en red permiten compartir recursos y conectar varios equipos entre sí dentro de una red de ordenadores.

SISTEMAS OPERATIVOS MÁS USUALES

SISTEMA OPERATIVO	NÚMERO DE USUARIOS	TAREAS	PROCESADORES	TIEMPO DE RESPUESTA
MS-DOS 6.22	MONOUSUARIO	MONOTAREA	MONOPROCESADOR	TIEMPO REAL
Microsoft Windows 95	MONOUSUARIO	PSEUDOMULTITAREA	MONOPROCESADOR	TIEMPO REAL
Microsoft Windows 95 Versión OSR2	MONOUSUARIO	PSEUDOMULTITAREA	MONOPROCESADOR	TIEMPO REAL
Microsoft Windows 98	MONOUSUARIO	PSEUDOMULTITAREA	MONOPROCESADOR	TIEMPO REAL
Microsoft Windows Me Millennium Edition	MONOUSUARIO	PSEUDOMULTITAREA	MONOPROCESADOR	TIEMPO REAL
Microsoft Windows NT Workstation	MONOUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Microsoft Windows NT Server Version 3.5	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO
Microsoft Windows 2000 Professional	MONOUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Microsoft Windows 2000 Server	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO
Microsoft Windows XP Professional	MONOUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Microsoft Windows Server 2003 Enterprise Edition	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO
Windows Vista	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Windows 7	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Windows 8	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO REAL
Windows Server 2008	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO
	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO
SCO	MULTIUSUARIO	MULTITAREA	MULTIPROCESADOR	TIEMPO COMPARTIDO

5. ESTRUCTURA DE UN SISTEMA OPERATIVO.

Inicialmente, los sistemas operativos eran monolíticos, es decir estaban escritos como un programa que realizaba todas las funciones del sistema operativo y, además, era muy difícil cambiar cualquier función o incluso corregir cualquier error.

Atendiendo a su estructura interna, los sistemas operativos se pueden dividir en:

- Monolíticos.
- En niveles o capas.
- Máquina virtual.
- Cliente-servidor.

5.1. MONOLÍTICOS.

Los sistemas operativos monolíticos no están estructurados, sino que están formados por un programa que consta de un conjunto de funciones o procedimientos interrelacionados entre sí. Su modificación o la búsqueda de algún error es una tarea bastante dificultosa. A medida que aumentan las funciones del sistema operativo y los recursos del sistema informático se hace necesaria una organización o estructuración de los sistemas operativos.

5.2. EN NIVELES O EN CAPAS.

Los sistemas operativos también pueden estar estructurados en diferentes niveles o capas, cada uno de los cuales tiene una función claramente definida y una interfaz con la que se comunican los niveles adyacentes entre sí.

El primer sistema operativo divido por capas fue **THE** que se dividía en los siguientes niveles:

NIVEL	FUNCIÓN
5	Operador
4	Programas de usuario
3	Gestión de Entrada/Salida
2	Comunicación operador-procesos
1	Gestión de la memoria principal y secundaria
0	Asignación del procesador y multiprogramación
	Hardware

Donde el nivel más bajo es el que interactúa directamente con el hardware. El más alto es donde el usuario ejecuta sus programas y aplicaciones.

5.3. MÁQUINA VIRTUAL.

Los sistemas operativos de máquina virtual presentan a cada proceso una máquina que parece idéntica a la máquina real. Estos sistemas operativos deben ser multitarea, y su objetivo es poder instalar sobre un mismo equipo diferentes sistemas operativos pero dando la sensación de que son diferentes equipos. Estos sistemas operativos pueden presentar diferentes máquinas virtuales, que serán réplicas de una máquina real. No es necesario que sean distintos sistemas operativos, sino que se pueden instalar varias réplicas de máquinas que tengan instaladas el mismo sistema operativo.

5.4. CLIENTE-SERVIDOR.

Es otro tipo de sistema operativo en el que los procesos del sistema operativo pueden ser tanto clientes como servidores. Un programa de aplicación de un usuario se que se está ejecutando se convertiría en un proceso cliente, que pide al sistema operativo servicios que serían los procesos servidores.

6. FUNCIONES DE UN SISTEMA OPERATIVO.

Las funciones principales de un sistema operativo son:

- Gestión de procesos.
- Gestión de memoria.
- Gestión de Entrada/Salida.
- Gestión de archivos.
- Gestión de la seguridad.

6.1. GESTIÓN DE PROCESOS.

Un proceso es un programa que está en ejecución. Cada vez que se manda ejecutar un programa se crearía un proceso. Es sistema operativo debe realizar una gestión adecuada de los recursos del sistema para la correcta ejecución de los procesos.

6.1.1. Procesos

Cada vez que se ejecuta un programa se crea una estructura de datos llamada **BLOQUE DE CONTROL DE PROCESO** (BCP). El sistema operativo identifica a cada proceso con esta estructura, que lo diferencia unívocamente de cualquier otro proceso y que puede contener, dependiendo del sistema operativo del que se trate, la siguiente información: **PID** (Procces Identificador) número entero único y diferente para cada proceso, **estado** en que se encuentra, **prioridad**, **recursos asignados**, **valores de los registros del procesador**, **propietario**, **premisos**,…

Dependiendo de si el proceso lo ha lanzado un usuario o es un proceso del sistema operativo, podemos distinguir entre:

- **Procesos del sistema**. Son procesos que generalmente se ejecutan al iniciar el sistema y suelen estar en ejecución para proporcionar servicios a los usuarios del sistema.
- **Procesos de los usuarios**. Son los procesos que manda ejecutar el usuario que entre en el sistema. Si el sistema es multiusuario, puede haber varios procesos de distintos usuarios ejecutándose a la vez.

Estados de los procesos.

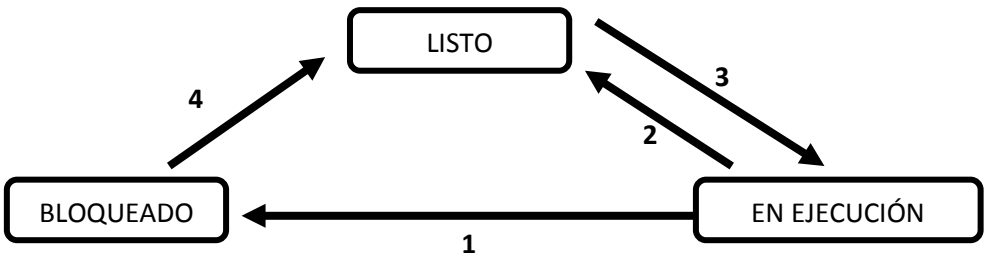
Los **estados** en los que se encuentra un **proceso** son:

- **Listo, en espera o preparado.** El proceso está preparado para ejecutarse, es decir, está en espera de que el proceso que se está ejecutando deje libre la UCP.
- **Bloqueado.** El proceso está esperando un recurso que está siendo utilizado por otro proceso en ese momento.
- **En ejecución.** El proceso está ejecutando sus instrucciones en ese momento, es decir, está ocupando la UCP.

Transición de los procesos.

Cuando un proceso pasa de un estado a otro, se produce una transición de estado. Podemos tener cuatro transiciones:

1. **De ejecución a bloqueado:** ocurre cuando el programa que está en ejecución necesita algún elemento, señal, dato, etc., para continuar ejecutándose.
2. **De ejecución a listo:** ocurre cuando un programa o proceso ha utilizado el tiempo asignado por la UCP para su ejecución y tiene que dejar paso al siguiente proceso.
3. **De listo a en ejecución:** ocurre cuando el proceso que está preparado pasa al proceso de ejecución, es decir, cuando al proceso le llega una nueva disposición de tiempo de la UCP para poder ejecutarse.
4. **De bloqueado a listo:** ocurre cuando el proceso recibe una orden o señal que estaba esperando para pasar al estado de preparado y, posteriormente, tras la transición, a estado de ejecución.



A la operación de desalojar un proceso de la UCP para que otro empiece a ejecutarse, se le llama **CAMBIO DE CONTEXTO**.

6.1.2. Algoritmos de planificación de procesos.

En sistemas operativos multitarea los procesos aparentemente se ejecutan a la vez, pero si sólo hay un procesador, este se tiene que distribuir entre todos los procesos que se estén ejecutando.

A la forma en que la UCP se distribuye para ejecutar los procesos se le llama **planificación**.

En cada algoritmo sabremos para cada proceso:

- **Tiempo de entrada o de llegada al sistema (T_l),** es el momento en que el proceso entra en el sistema.
- **Tiempo de ejecución (T_x),** es el tiempo que el proceso necesita para su ejecución total.

Nos interesa obtener para cada proceso los siguientes datos:

- **Tiempo de respuesta o de retorno (T_R),** es el tiempo que pasa desde que el proceso llega al sistema hasta que se obtienen los resultados.
- **Tiempo de espera (T_E),** es el tiempo que el proceso pasa dentro del sistema en espera.

$$T_E = T_R - T_X$$

- Además también querremos saber los **tiempos medios** tanto de respuesta como de espera, para poder saber si para ciertos datos de entrada el algoritmo es más o menos óptimo, en general.

Entre los distintos algoritmos de planificación podemos destacar:

a) Algoritmo primero en entrar primero en salir: FIFO (First Input First Output).

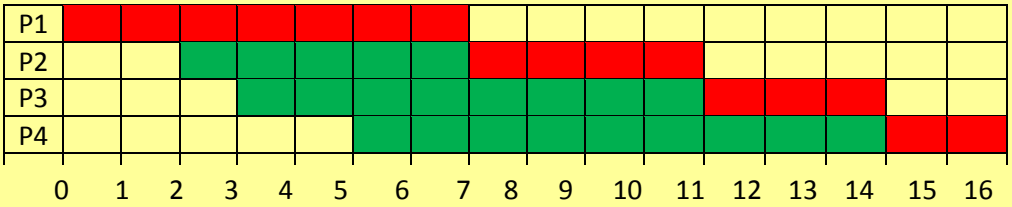
Utilizando este algoritmo, los procesos se ejecutan en el orden de llegada. El primero que llega se empieza a ejecutar, y los siguientes deberán esperar su turno para poder empezar a ejecutarse.

Para los siguientes procesos, con el tiempo de llegada y el tiempo de ejecución necesario que se indica, calcula los tiempos de espera y respuesta de cada proceso y los tiempos medios, utilizando el algoritmo FIFO.

PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
P1	0	7
P2	2	4
P3	3	3
P4	5	2

Solución

Vamos a realizarlo gráficamente mediante un cronograma, para que se vea en color verde el tiempo de espera de cada proceso (tiempo que está en estado de espera) y en rojo el tiempo de uso de la UCP (tiempo que está en ejecución). El tiempo de respuesta será el tiempo transcurrido desde que el proceso llega al sistema hasta que termina, sale de la UCP y se obtienen los resultados.



PROCESO	TIEMPO DE ESPERA	TIEMPO DE RESPUESTA
P1	0	7
P2	5	9
P3	8	11
P4	9	11
TIEMPOS MEDIOS	**5,5**	**9,5**

b) Algoritmo primero el más corto: SJF (short job first)

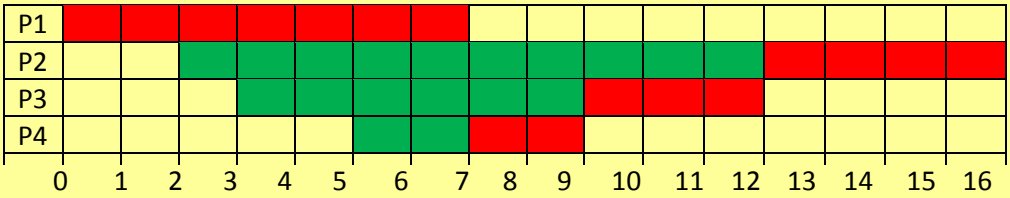
Este algoritmo coge el proceso más corto de los que están esperando para usar la UCP. En caso de igual tiempo, se aplica el FIFO. Favorece a los procesos que tarden menos tiempo en ejecutarse.

Para los siguientes procesos, con el tiempo de llegada y el tiempo de ejecución necesario que se indica, calcula los tiempos de espera y respuesta de cada proceso y los tiempos medios, utilizando el algoritmo SJF.

PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
P1	0	7
P2	2	4
P3	3	3
P4	5	2

Solución

Vamos a realizarlo gráficamente mediante un cronograma, para que se vea en color verde el tiempo de espera de cada proceso (tiempo que está en estado de espera) y en rojo el tiempo de uso de la UCP (tiempo que está en ejecución). El tiempo de respuesta será el tiempo transcurrido desde que el proceso llega al sistema hasta que termina, sale de la UCP y se obtienen los resultados.



PROCESO	TIEMPO DE ESPERA	TIEMPO DE RESPUESTA
P1	0	7
P2	10	14
P3	6	9
P4	2	4
TIEMPOS MEDIOS	**4,5**	**8,5**

c) Algoritmo primero el tiempo restante más corto: SRTF (short remaining time first)

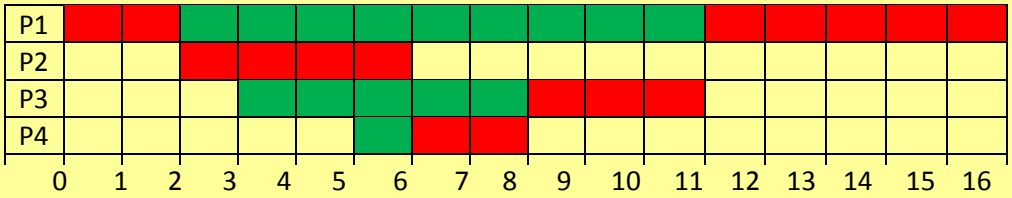
Este algoritmo va seleccionando de los procesos que están en espera al que le quede menor tiempo para terminar. En caso de empate, se utiliza FIFO. Este algoritmo es apropiativo, ya que si mientras se está ejecutando un proceso llega otro que le quede menos tiempo para acabar que el que está utilizando la UCP en ese momento lo desplaza, es decir se produce un cambio de contexto. Favorece a los procesos con menor tiempo de ejecución y mejora los tiempos medios en general. Sin embargo, perjudica a los que necesiten más tiempo.

Para los siguientes procesos, con el tiempo de llegada y el tiempo de ejecución necesario que se indica, calcula los tiempos de espera y respuesta de cada proceso y los tiempos medios, utilizando el algoritmo SRTF.

PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
P1	0	7
P2	2	4
P3	3	3
P4	5	2

Solución

Vamos a realizarlo gráficamente mediante un cronograma, para que se vea en color verde el tiempo de espera de cada proceso (tiempo que está en estado de espera) y en rojo el tiempo de uso de la UCP (tiempo que está en ejecución). El tiempo de respuesta será el tiempo transcurrido desde que el proceso llega al sistema hasta que termina, sale de la UCP y se obtienen los resultados.



PROCESO	TIEMPO DE ESPERA	TIEMPO DE RESPUESTA
P1	9	16
P2	0	4
P3	5	8
P4	1	3
TIEMPOS MEDIOS	**3,75**	**7,75**

d) Algoritmo de operación por rondas: RR (Round Robin)

Este algoritmo va dando tiempo de ejecución a cada proceso que esté en espera. Se debe establecer un tiempo o quantum (q), tras el cual el proceso abandona la UCP y da paso al siguiente, siguiendo el orden FIFO. Para este algoritmo es necesario que el cambio de contexto sea muy rápido. Si no hubiera procesos en espera el que está utilizando la UCP seguiría hasta que llegara alguno. Es un algoritmo expulsivo.

Para los siguientes procesos, con el tiempo de llegada y el tiempo de ejecución necesario que se indica, calcula los tiempos de espera y respuesta de cada proceso y los tiempos medios, utilizando el algoritmo de operación por rondas con q=2.

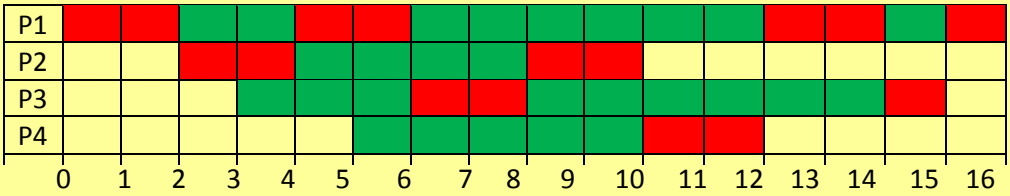
PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
P1	0	7
P2	2	4
P3	3	3
P4	5	2

Solución

Vamos a realizarlo gráficamente mediante un cronograma, para que se vea en color verde el tiempo de espera de cada proceso (tiempo que está en estado de espera) y en rojo el tiempo de uso de la UCP (tiempo que está en ejecución). El tiempo de respuesta será el tiempo transcurrido desde que el proceso llega al sistema hasta que termina, sale de la UCP y se obtienen los resultados.

Los procesos van llegando en el siguiente orden a la cola de listos: P1 (llega al sistema), P2 (llega al sistema), P1 (acaba su quantum), P3 (llega al sistema), P2 (termina su quantum), P4 (llega al sistema), P1 (termina su quantum), P3 (termina su quantum), P1 (termina su quantum).

El orden de los procesos que salen de la cola de listos a ejecución siguiendo el orden FIFO en dicha cola sería: P1, P2, P1, P3, P2 (finaliza), P4 (finaliza), P1, P3 (finaliza), P1 (finaliza). Los procesos que acaban no se incorporan a la cola de listos.



PROCESO	TIEMPO DE ESPERA	TIEMPO DE RESPUESTA
P1	9	16
P2	4	8
P3	9	12
P4	5	7
TIEMPOS MEDIOS	**6,75**	**10,75**

Lista de espera

P1
P3
P2
P4
P1
P3
P1

e) Algoritmo por Prioridades o multinivel

Es uno de los más complejos y eficaces. Asigna los tiempos de la CPU según una lista de prioridades. El tiempo de ejecución del procesador, se irá destinando, en primer lugar, de forma secuencial a los procesos de mayor nivel o prioridad. Si existen procesos de la misma prioridad, se aplica un intervalo de tiempo de uso de la CPU para cada proceso. Terminados los de mayor prioridad, se ejecutarán los procesos de nivel o prioridad inferior, y así sucesivamente, hasta llegar a los procesos de nivel más bajo.

Ejemplo:

TAREAS	Ciclos de CPU	Orden de llegada	Prioridad
A	7	0	Alta
B	5	1	Baja
C	8	2	Alta

Quantum = 1

Ejecución	A	A	C	A	C	A	C	A	C	A	C	A	C	C	C	B	B	B	B	B
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Este algoritmo, puede ser apropiativo, si cuando llega un proceso con mayor prioridad que la del que se está ejecutando, se casa de la CPU y se mete el nuevo de mayor prioridad; o no apropiativo, si cuando se da la circunstancia anterior, se deja el proceso que se está ejecutando hasta que termine.

f) **Algoritmo de colas multinivel con realimentación**

Está basado en el algoritmo de colas multinivel, que consiste en un sistema que posee un número de colas de diferente prioridad. Cada cola se gestiona mediante un algoritmo de planificación. Los procesos llegan a una cola dependiendo del nivel de prioridad que tengan. Primero se ejecutarán los procesos de la cola de mayor prioridad. Esto puede producir inanición si no paran de llegar procesos a las colas de mayor prioridad, ya que las de menor prioridad no podrán ejecutar sus procesos.

Para solucionar este problema, se creo el algoritmo de colas multinivel con realimentación, donde un proceso se puede mover entre las diferentes colas.

Un algoritmo de planificación de colas multinivel con realimentación está definido por los siguientes parámetros:

- número de colas
- algoritmos de planificación para cada cola
- método usado para determinar cuándo promover un proceso a una cola de mayor prioridad
- método usado para determinar cuándo degradar un proceso a una cola de menor prioridad
- método usado para determinar en qué cola ingresará un proceso cuando necesite servicio

Ejemplo:

Tenemos tres colas:

- *Q*0 – RR con quantum 8 ms
- *Q*1 – RR con quantum 16 ms
- *Q*2 – FCFS
-

La planificación se realiza de la siguiente manera:

- Un proceso que entra en la cola de procesos listos ingresa en la cola *Q0* . Cuando obtiene la CPU se le asignan 8 ms. Si no termina su ráfaga de CPU en ese tiempo se pasa a *Q*1.
- En *Q*1 se asignan 16 ms de CPU al proceso. Si no termina en ese tiempo es expropiado y colocado en la cola *Q*2.

	Apropiativo	**Productividad**	**Tiempo de respuesta**	**Sobrecarga**	**Efecto sobre los procesos**	**Inanición**
FIFO	NO	No relevante	Puede ser alto	Mínima	Penaliza los procesos cortos y los que tiene E/S	NO
RR	Si (en los quantum de tiempo)	Puede ser baja si el quantum es muy pequeño	Bueno para procesos cortos	Mínima	Trato equitativo	NO
SJF	NO	Alta	Bueno para procesos cortos	Puede ser alta	Penaliza los procesos largos	Posible
SRTF	SI	Alta	Bueno	Puede ser alta	Penaliza los procesos largos	Posible
Prioridades	SI	Alta	Bueno	Puede ser alta	Buen equilibrio	Posible
M con realimenta.	Si	No relevante	No relevante	Puede ser alta	Buen equilibrio	Posible

Todos los algoritmos vistos anteriormente se pueden implementar teniendo en cuenta las operaciones de E/S que realizan los procesos. Las operaciones de E/S también duran instantes de tiempo. Cuando un proceso hace una operación de E/S, se saca de la cola y vuelve a entrar cuando termina de hacer la E/S. A partir de este momento el algoritmo se puede implementar de dos formas:

A) Si en el mismo instante entra un proceso nuevo y uno que viene de una operación de E/S, se le da prioridad al ponerlo en la cola al proceso nuevo.
B) Al contrario.

6.2. GESTIÓN DE MEMORIA.

Un sistema operativo debe ser capaz de gestionar la memoria del sistema eficazmente. Los procesos que se ejecutan en el sistema necesitan que se les asigne una zona de memoria para su ejecución, que se le proteja esa zona de otros accesos o poderla compartir si fuera necesario y una vez que terminan el sistema debe liberar la memoria que le asignó al proceso.

6.2.1. Memoria principal.

Un sistema informático tiene una memoria principal o memoria RAM donde se cargará el programa que va a entrar a ejecutarse.

6.2.2. Gestión de la memoria en monoprogramación y multiprogramación.

Dependiendo de si el sistema es monotarea o multitarea, las técnicas de gestión de memoria serán distintas.

En los sistemas monotarea, se van ejecutando los procesos uno a uno. Cuando uno termina entra otro. Tan sólo hay que tener en cuenta que en la memoria debe haber una zona para la parte residente del sistema operativo y otra para el proceso que se esté ejecutando en ese momento.

Si el sistema es multitarea, es decir que se pueden ejecutar varios procesos a la vez, será necesario dividir la memoria entre los procesos, con lo que habrá que crear deferentes particiones y asignarle una a cada proceso.

Técnicas de gestión de memoria.

Las siguientes técnicas se utilizan en sistemas operativos multitarea:

- **Particionamiento**. La forma de particionar la memoria principal podrá ser mediante la:
  - **Paginación**: las particiones son de tamaño fijo. A cada zona de memoria se la llama marco de página y el proceso se divide en trozos iguales llamados páginas.
  - **Segmentación**: las particiones son de tamaño variable. A cada zona de memoria se le llama segmento.
- **Intercambio** (swapping). Cuando un proceso queda suspendido porque ha terminado su cuanto de tiempo asignado por el sistema o bien porque está esperando unos recursos asignado por otro proceso, hay que desalojarlo de la memoria para alojar en ella otro proceso. El proceso desalojado se guarda en la memoria secundaria y el nuevo proceso se carga en la memoria principal.
- **Compartición.** Cuando se ejecutan dos o más procesos iguales o procesos que ayuden a la ejecución de un mismo programa, se debe permitir que los procesos compartan memoria para evitar la redundancia de procesos. Esta técnica hay que realizarla de forma muy controlada, para proteger los procesos.
- **Reubicación.** Un proceso cuando se descargue porque quede suspendido, cuando vuelva a cargarse en memoria no necesita que sea la misma partición, sino que se puede cargar en otra partición diferente.

- **Memoria virtual.** Surge este concepto para poder ejecutar programas que no estén cargados totalmente en memoria, y poder así cargar varios procesos a la vez cuyos tamaños sean superiores a la memoria física o bien un proceso cuyo tamaño sea superior a la memoria física del sistema. Consistiría en cargar en memoria principal la parte del proceso y la que no se ejecute se cargará en la memoria secundaria.

Un problema que suele suceder en multiprogramación al particionar la memoria es:

- **Fragmentación**. Es el desaprovechamiento de memoria. Si la fragmentación es muy alta, puede suceder que haya gran cantidad de memoria desocupada y, sin embargo, un proceso no encuentre memoria disponible. La fragmentación puede ser:
  - **Interna**: se suele dar con la paginación, como las páginas son de un tamaño fijo, puede que un proceso de menor tamaño que la página se cargue en un marco de página, desaprovechando, por lo tanto la memoria sobrante. Es la memoria que no se utiliza interna a la partición.
  - **Externa**: se suele dar con la segmentación. Como los segmentos son de tamaño variable, cada proceso se carga en un segmento con un tamaño concreto, pero puede suceder con el tiempo con que no encuentre una zona de memoria libre lo suficientemente grande para almacenar un proceso y, sin embargo, que el tamaño libre total sea mayor que el tamaño de proceso. Es la memoria que no se utiliza externa a la partición.

6.2.3. Esquemas de memoria basados en asignación contigua

Consiste en asignar a cada proceso una zona contigua de memoria. Este esquema está obsoleto. Se distinguen dos modelos:

A. **ASIGNACIÓN ESTÁTICA DE MEMORIA PARTICIONADA**: También llamado multiprogramación con particiones fijas o estáticas. Realiza una división de la memoria en particiones fijas a partir del momento en que se crea el sistema. El tamaño de las particiones se ajusta al tamaño de los procesos que use el sistema fundamentalmente. El número de particiones está en función de la capacidad de la memoria y del número de procesos que vayan a ejecutarse a la vez, denominándose este número *grado de multiprogramación*. (una memoria de 2Mb con particiones de 256KB)

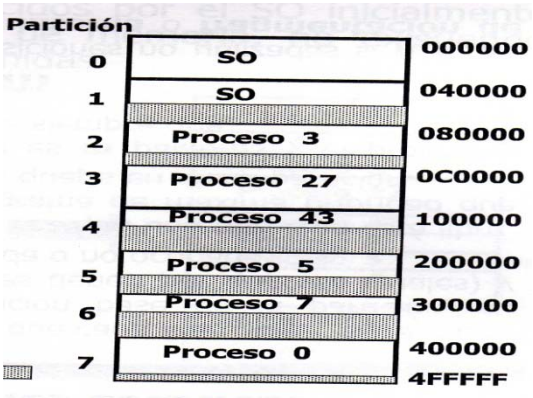
   El S.O. mantienen una tabla llamada *tabla de descripción de particiones (DTP)* que almacena información sobre las particiones como el nº de partición, posición, tamaño y estado.

   Cuando un proceso es activado, el S.O. intenta asignar una partición de la memoria que esté libre y sea de tamaño suficiente, para lo cual consulta la DTP; si la encuentra, le cambia el estado de libre a asignada y carga el proceso en esa partición.

   Para que el S.O. sepa qué proceso hay en cada partición, en el momento de la carga se graba en el *bloque de control de procesos (BCP)* el proceso y la partición asignada.

Las principales características de este tipo de asignación son:

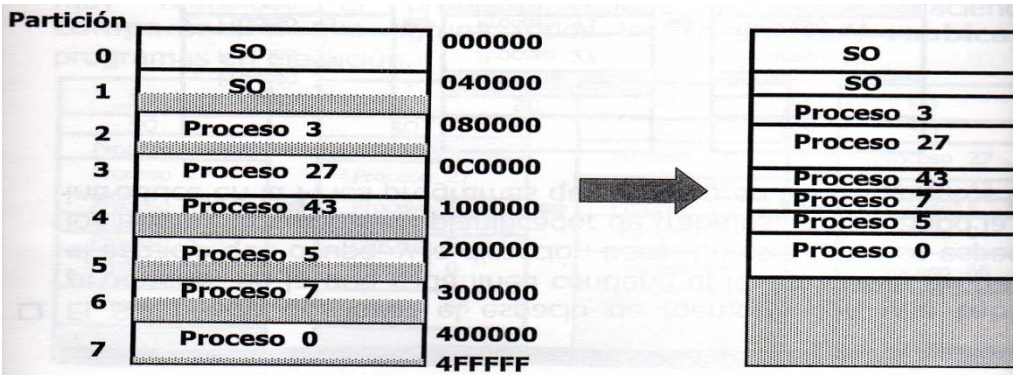
- Es la forma más sencilla de gestión de memoria en multiprogamación. Idónea para procesos de tamaño y características conocidas.
- Es inflexible para estructuras que impliquen un crecimiento dinámico.
- Se puede producir *fragmentación interna*, es decir, cuando dentro de una partición el proceso asignado no la ocupa completamente. Si es muy alta, se desaprovecha gran parte de la memoria.
- Se puede producir una mínima fragmentación externa.



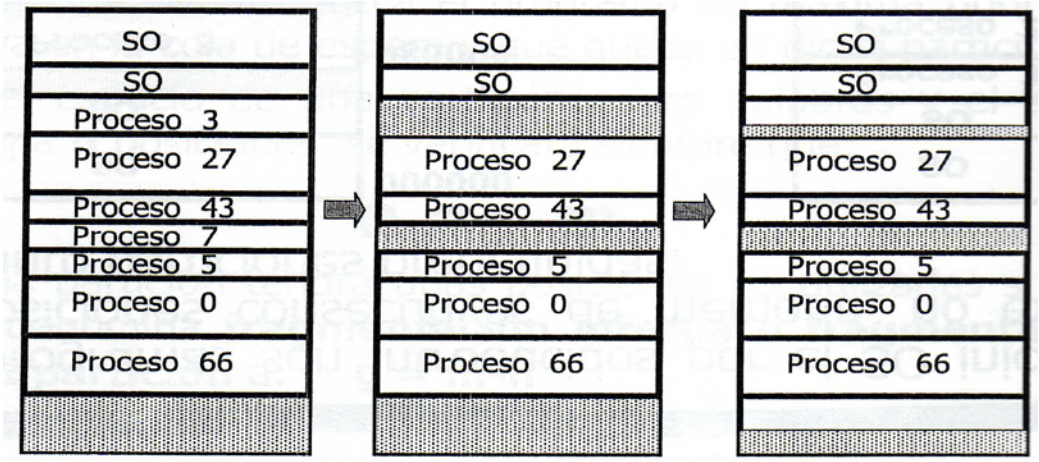
Hay varias formas de satisfacer una solicitud de tamaño *n* partiendo de una lista de huecos:

- Primer ajuste (First-fit): Se asigna el *primer* hueco lo suficientemente grande.
- Mejor ajuste (Best-fit): Se asigna el hueco *más pequeño* que es lo suficientemente grande; hay que buscar en la lista entera de huecos (salvo si está ordenada por tamaño). Desperdicia el menor espacio posible.

B. **ASIGNACIÓN DINÁMICA DE MEMORIA PARTICIONADA**: También llamada multiprogramación con particiones variables o dinámicas. En este tipo de asignación de memoria, el particionamiento en zonas de ésta se va redefiniendo cada vez que un proceso termina y se saca de la memoria. Los procesos son introducidos por el S.O. inicialmente en posiciones consecutivas de memoria, no existiendo particiones predefinidas. Primero se crean particiones del tamaño del programa solicitante hasta que se llena la memoria o hasta que el programa que tiene que entrar es mayor que el fragmento que queda libre.

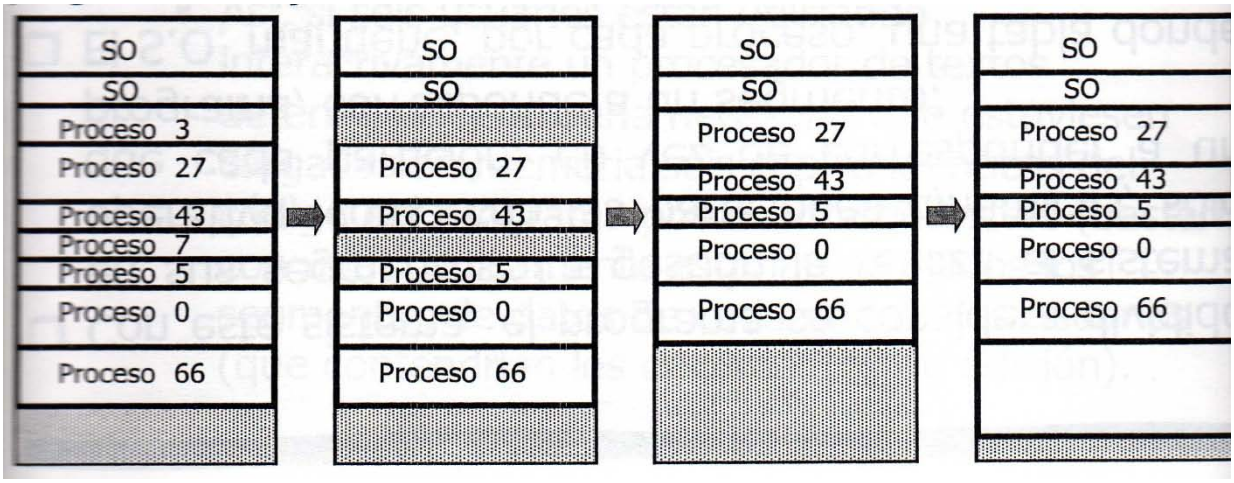


A continuación, a medida que salen los procesos de la memoria dejando espacio libre, se crean nuevas zonas en dicho espacio del tamaño de los procesos entrantes. Ningún programa ocupa más espacio del que necesita, por lo que sólo existe *fragmentación externa*, que consiste en la existencia de regiones de memoria sin usar demasiado pequeñas para los procesos existentes. Puede haber suficiente memoria para satisfacer una petición, pero esa memoria no es contigua. No existe límite para el número de particiones.



Para controlar las partes libres y asignadas, el S.O. mantiene una *tabla de descripción de particiones*; cada vez que se crea una partición, se introduce en dicha tabla su tamaño, su estado y su dirección.

Al ir saliendo y entrando procesos, crece el número de fragmentos libres pero disminuye el tamaño de estos, llegando el momento en que el porcentaje de memoria aprovechado es muy reducido. Este problema se resuelve, haciendo una *compactación* cada cierto tiempo que agrupe todos los fragmentos libres, *reubicando* los programas en ejecución.



Hay varias formas de satisfacer una solicitud de tamaño *n* partiendo de una lista de huecos:

- Primer ajuste (First-fit): Se asigna el *primer* hueco lo suficientemente grande
- Mejor ajuste (Best-fit): Se asigna el hueco *más pequeño* que es lo suficientemente grande; hay que buscar en la lista entera de huecos (salvo si está ordenada por tamaño). Da lugar al hueco más pequeño.
- Peor ajuste (Worst-fit): Se asigna el hueco *más grande*; hay que buscar en la lista completa de huecos (salvo si está ordenada por tamaño). Da lugar al hueco más grande

6.2.4. Esquemas de memoria basados en asignación no contigua

Están basados en la técnica de memoria virtual, que se ocupa de la transferencia de información entre la memoria principal y la secundaria. Esta técnica se utiliza ya que es muy común que el espacio que se necesita para almacenar programas y datos es mayor que la memoria de que se dispone, por lo que el S.O. deja en memoria sólo lo que necesite y el resto en disco, intercambiando partes entre memoria y disco con forme se vayan necesitando.

Los procesos presentan la propiedad de *proximidad de referencias*, que permite que un proceso genere muy pocos fallos aunque tenga en memoria principal sólo una pequeña parte de el, la que se necesita para ejecutarse en ese momento concreto. El objetivo del sistema de memoria virtual es intentar que la información que esté usando un proceso en un momento determinado esté residente en memoria principal, es decir, que en M.P. esté la parte del proceso en ejecución y los datos que necesita esa parte.

Con esta técnica se consiguen las siguientes ventajas:

- Aumento del grado de multiprogramación ya que no es necesario que todo el mapa de memoria de un proceso esté en M.P. para poder ejecutarse. Esto mejora el rendimiento del sistema. Si el grado de multiprogramación se hace muy alto se produce una hiperpaginación, es decir, aumenta el número de fallos de página y baja el rendimiento del sistema.

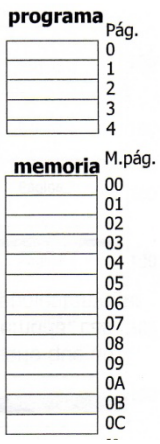
- Se pueden ejecutar programas más grandes que la M.P. de la que se dispone.

Se pueden distinguir tres esquemas de asignación de memoria no contigua, como son la paginación, la segmentación y la paginación segmentada.

A. **PAGINACIÓN**: El mapa de memoria (espacio de direcciones) de cada programa está dividido bloques de tamaño fijo llamados *páginas*; cada página dentro del programa se identifica con un número correlativo.

Por ejemplo, si la página es de 4KB, y el programa de 64KB, éste estaría formado por 16 páginas (0 a F). La memoria principal está dividida en zonas del mismo tamaño de las páginas denominadas *marcos de páginas*; cada marco se identifica con un número relativo. Por ejemplo, si el tamaño de la memoria es de se 1MB, y el tamaño del marco de 4KB, habría 256 marcos de página (del 00 al FF). Un marco de página de memoria contendrá en un instante determinado una página de memoria de un proceso.

El fundamento de la paginación, está en que no es necesario que un programa se almacene en posiciones consecutivas de memoria. Las páginas se almacenan en marcos de página libres independientemente de que estén o no contiguos.



División del programa y de la memoria

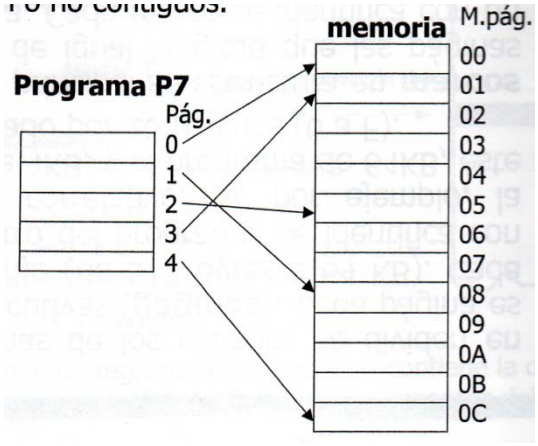
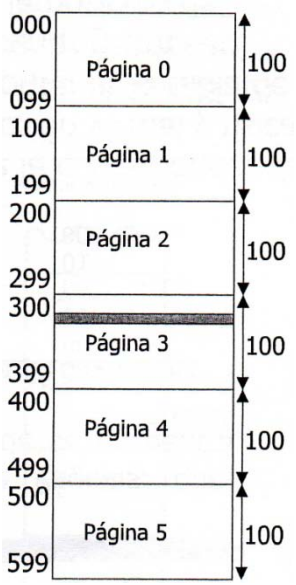


Tabla de páginas que mantiene la asignación de marcos en la memoria

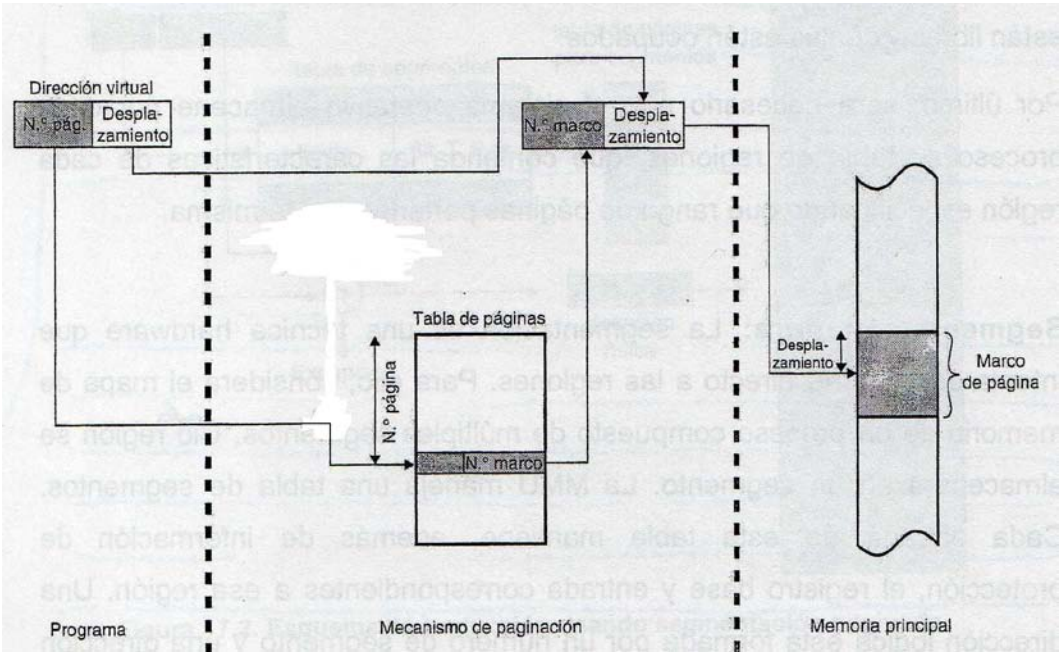


Una dirección lógica o virtual generada por un proceso está formada por el *número de página* y un *desplazamiento* dentro de la página. Por ejemplo, si las páginas son de 100 posiciones, y tenemos la dirección 328, esta corresponderá a la página 3, y, dentro de ella tendrá una posición relativa (desplazamiento) de 28 posiciones.

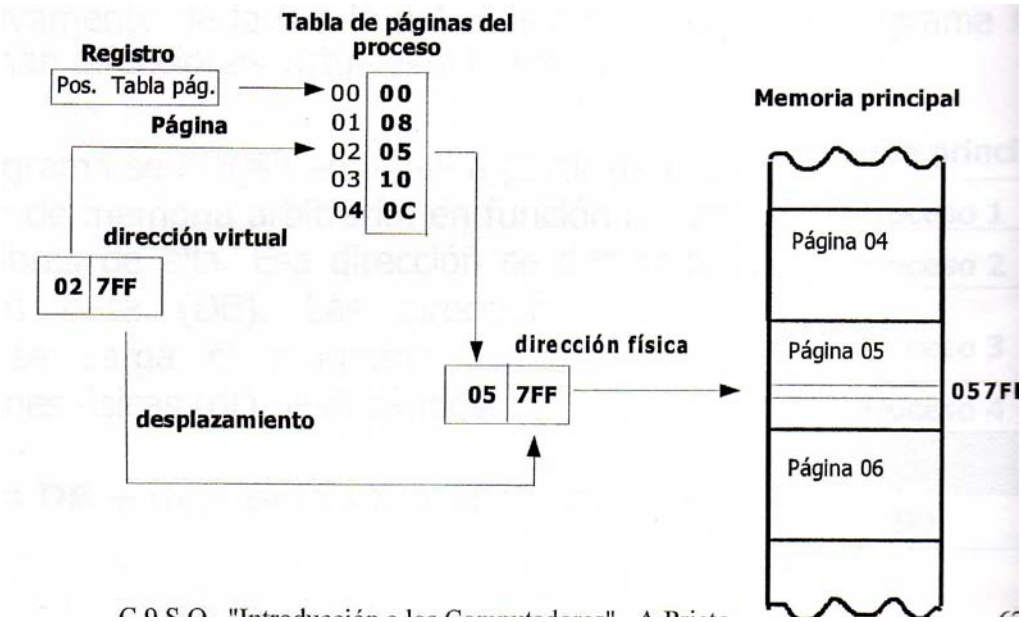
La tabla de páginas es la estructura de datos que relaciona cada página con el marco donde está almacenada.

La MMU (Memory-Management Unit, Unidad de Gestión de Memoria) es un dispositivo hardware que transforma las direcciones virtuales que genera un programa en direcciones físicas de la memoria.

La dirección física se obtiene a partir de la dirección virtual, consultando la tabla de páginas y concatenando al marco de página resultante el desplazamiento.



Hardware de paginación



Ejemplo de acceso a memoria en paginación

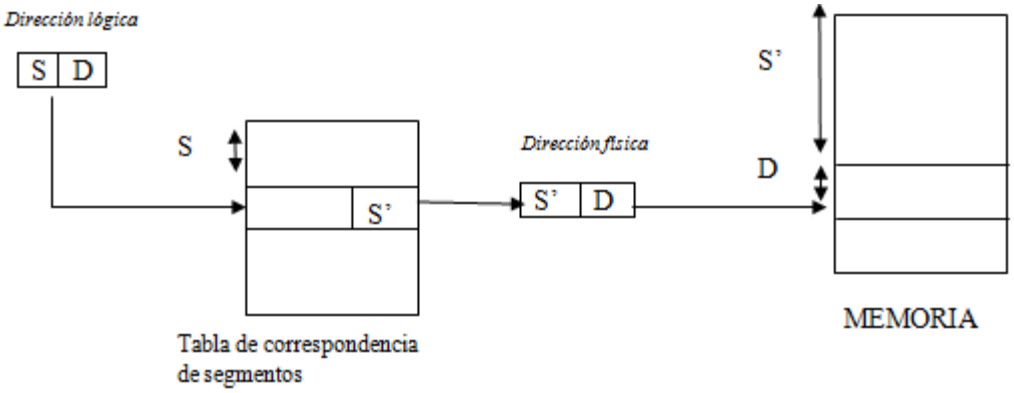
B. **SEGMENTACIÓN PURA**: La segmentación es una técnica hardware que permite definir los bloques de memoria de tamaño variable. A estos bloques se les denomina segmentos. Los segmentos pueden tener distinta longitud, según las necesidades del programa.

Considera que el mapa de memoria de un proceso está compuesto de múltiples segmentos. La MMU maneja una tabla de segmentos.

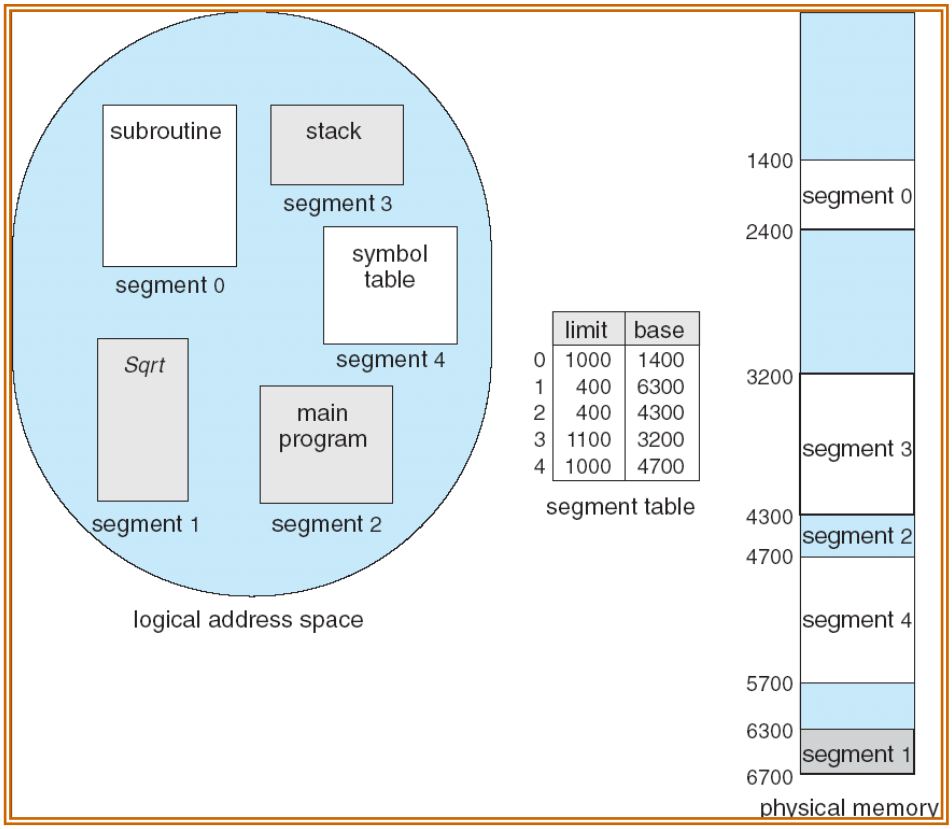
Una dirección lógica está formada por un número de segmento y una dirección dentro del segmento (desplazamiento). Un segmento se coloca en memoria en cualquier zona cuyo tamaño sea suficiente para colocarlo. Cuando llega una dirección lógica, se busca en la tabla de correspondencia de segmentos la dirección de memoria correspondiente a ese segmento.

La tabla de segmentos contiene información sobre la ubicación de los segmentos en memoria; cada entrada de la tabla tiene:

- **base** – contiene la dirección física en la que comienza el segmento
- **límite** – especifica la longitud del segmento



Esquema de traducción usando segmentación pura



Ejemplo de segmentación

Este esquema presenta *fragmentación externa* ya que cada segmento se almacena en memoria de forma contigua, por lo que cuando vayan saliendo los procesos de memoria irán creando huecos que si son muy pequeños puede que otros procesos no los puedan ocupar. Este esquema no se suele usar.

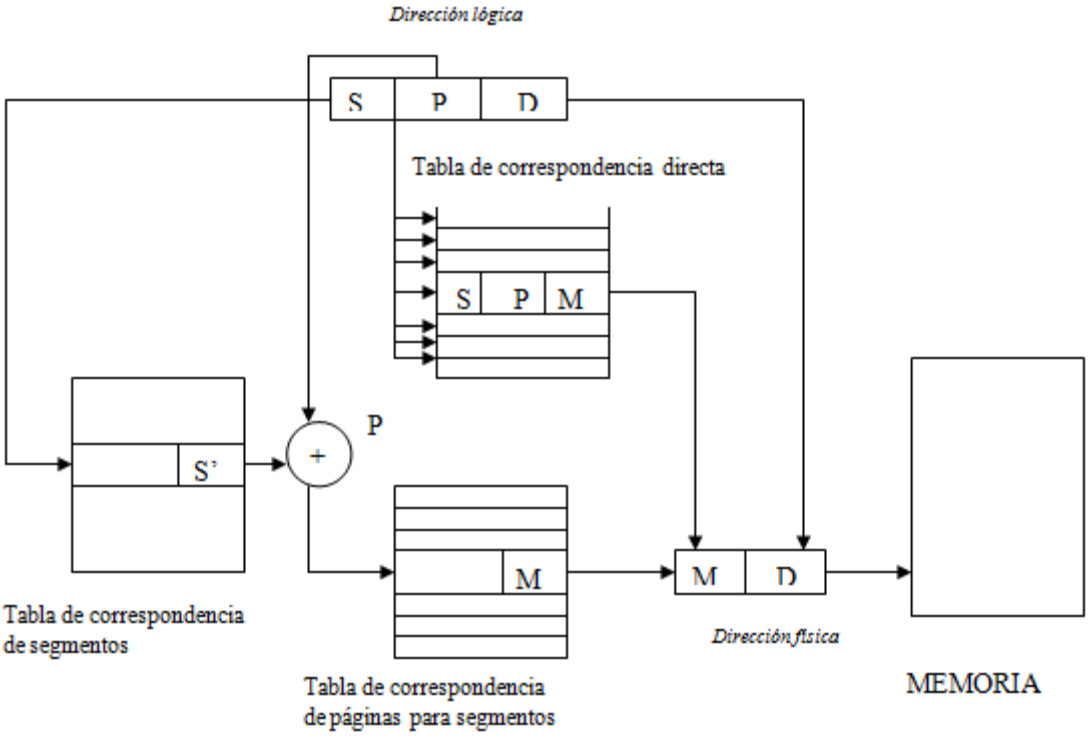
C. **SEGMENTACIÓN PAGINADA**: Este esquema intenta coger lo mejor de los dos esquemas vistos anteriormente. Con esta técnica, un segmento está formado por un conjunto de páginas y no tiene que estar todas en memoria ni de forma contigua. La MMU usa una tabla de segmentos donde cada entrada de la tabla apunta a una tabla de páginas.

Una dirección lógica tiene la siguiente forma:

S	P	D

Donde S es el segmento, P es la página dentro del segmento y D el desplazamiento dentro de la página.

Existe una pequeña *tabla de correspondencia directa* que almacena las direcciones más usadas. Cuando llega una dirección lógica, primero se busca en esta tabla para obtener el nº de marco de página correspondiente directamente. Si no se encuentra, se usa el nº de segmento de la dirección lógica como entrada para la *tabla de correspondencia de segmentos* para obtener la dirección de la *tabla de correspondencia de páginas* para dicho segmento, que junto con la página de la dirección lógica se obtiene el marco para dicha página en memoria. A la salida de la tabla de correspondencia de páginas se le añade el desplazamiento para obtener la dirección física de memoria.



Esquema de traducción usando segmentación paginada

6.2.5. Estrategias de gestión de memoria

6.2.5.1.- Políticas de sustitución

Estas políticas determinan que información hay que sacar de la memoria principal, es decir, son las responsables de la creación de zonas de memoria libre:

a) ***Políticas de sustitución para sistemas paginados**: Los bloque de información a sustituir son páginas. Los algoritmos de sustitución más usados son:*

- ✓ *Algoritmo del primero en entrar, primero en salir (FIFO): Sustituye la página que hay estado residente en memoria más tiempo. El algoritmo FIFO es fácil de comprender y programar, pero su rendimiento no siempre es bueno. Además, este algoritmo presenta una irregularidad denominada anomalía de Belady, que dice que se puede dar la circunstancia al utilizar este algoritmo de que aumente el número de fallos de página al aumentar el número de marcos.*

- ✓ *Algoritmo óptimo: Una consecuencia del descubrimiento de la anomalía de Belady fue la búsqueda de un algoritmo de reemplazo de páginas óptimo. El algoritmo de reemplazo de páginas óptimo sería aquel que eligiera la página de la memoria que vaya a ser referenciada más tarde. Por desgracia, este algoritmo es irrealizable, pues no hay forma de predecir qué páginas referenciará un proceso en el futuro. Como resultado de esto, el algoritmo óptimo se utiliza sobre todo para estudios comparativos.*

- ✓ *Algoritmo del utilizado menos recientemente (LRU-Least Recently Used): Sustituye la página que haga más tiempo que no ha sido utilizada. La principal diferencia entre los algoritmos FIFO y OPT es que el primero utiliza el instante en que entró una página en memoria, y el segundo, utiliza el tiempo en el que se usará la página. Una buena aproximación al algoritmo óptimo se basa en el principio de localidad de referencia de los programas. Las páginas de uso frecuente en las últimas instrucciones se utilizan con cierta probabilidad en las siguientes instrucciones. De manera recíproca, es probable que las páginas que no hayan sido utilizadas durante mucho tiempo permanezcan sin ser utilizadas durante cierto tiempo. Es decir, se utiliza el pasado reciente como aproximación de lo que sucederá en el futuro. El algoritmo LRU explota esta idea: al ocurrir un fallo de página se sustituye la página que no haya sido utilizada hace más tiempo.*

- ✓ *Algoritmo del utilizado menos frecuentemente (LFU- Least Frequently Used): Sustituye la página que se haya usado menos veces en el transcurso de un intervalo de tiempo. Mantiene un contador del número de referencias que se han hecho para cada página. Se reemplaza la página con el menor recuento. Este algoritmo tiene problemas cuando una página se usa mucho en la fase inicial de un proceso, pero después ya no se utiliza. Como se usó bastantes veces, tiene un recuento alto y permanece en memoria aunque ya no se necesite.*

EJEMPLOS:

Para determinar el número de fallos de página para una serie de referencias y un algoritmo de reemplazo de página, necesitamos también el número de marcos de página por proceso. Si aumenta el número de marcos, se reducirá el número de fallos de página. Para comprobar los algoritmos de reemplazo de páginas, usaremos la siguiente secuencia de referencias 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.

A. *Reemplazo FIFO: Este algoritmo asocia a cada página el instante en el que se trajo a memoria. Cuando hay que reemplazar una página, se elige la más antigua. Para la cadena de referencias de ejemplo, y suponiendo un total de tres marcos, aplicamos este algoritmo con el resultado de 15 fallos de página.*

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
	0	0	0		3	3	3	2	2	2			1	1			1	0	0
		1	1		1	0	0	0	3	3			3	2			2	2	1

Para ilustrar la anomalía de Beladay, supongamos que ahora tenemos la siguiente serie de referencias 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5. Sorprendentemente el número de fallos para cuatro marcos (diez) es mayor que para tres marcos (nueve). Esto significa que con el algoritmo FIFO la tasa de fallos puede aumentar al incrementar el número de marcos asignados.

B. *Reemplazo OPTIMO: Sustituye la página que va a tardar más tiempo en ser referenciada. Es muy difícil que el S.O. conozca esta información. En este ejemplo se producen 9 fallos de página si se usa este algoritmo.*

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

C. *Reemplazo Menos Recientemente Usado o LRU: El resultado de aplicar este algoritmo a nuestro ejemplo produce 12 fallos.*

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		

D. *Reemplazo Menos Frecuentemente Utilizado o LFU: El resultado de aplicar este algoritmo a nuestro ejemplo produce 13 fallos.*

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
7	7	7	2	2	2	2	4	4	3	3	3	3	3	3	3	1	7	7	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	1	2	2	2	2	2	2

b) ***Políticas de sustitución para sistemas no paginados****: Los bloques de información a sustituir son segmentos. No todos los segmentos ocupan la misma cantidad de memoria, por lo que el segmento a relegar a la memoria secundaria vendrá influido por el tamaño del segmento a cargar.*

El algoritmo más sencillo consiste en sustituir el segmento que junto con los espacios libres adyacentes a este, libere suficiente memoria para el nuevo segmento. Si hay varios segmentos de estas características, tendrá que intervenir alguno de los algoritmos anteriores para elegir cual de ellos saldrá. Si no hay ningún segmento, habrá que sustituir varios buscando el menor conjunto de segmentos que dejen libre el espacio necesario.

6.2.5.2.- Políticas de carga

Estas políticas determinan cuando hay que cargar información en la memoria principal. Se pueden distinguir dos categorías:

a) ***Las de demanda****: La falta de un bloque provoca una petición de carga, con lo que los algoritmos de ubicación y/o sustitución harán sitio en la memoria para el nuevo bloque.*

b) ***Las anticipatorias****: Cargan los bloques por adelantado; se basa en predicciones del comportamiento futuro del programa. Estas predicciones pueden realizarse en base a 2 criterios:*

- ✓ *La naturaleza de la construcción del programa.*
- ✓ *La deducción basada en el comportamiento pasado.*

6.2.5.3.- Políticas de ubicación

Estas políticas determinan en qué lugar de la memoria principal hay que colocar la información leída, es decir, deben elegir una parte de la zona de memoria libre.

a) ***Políticas de ubicación para sistemas no paginados**: Se tendrá una lista de huecos (espacios libres en memoria). La tarea consiste en decidir que hueco usar y actualizar la lista después de cada inserción.*

Si el bloque a cargar es menor que el hueco, este se colocará en la parte izquierda o extremo inferior del hueco. Esta técnica minimiza la fragmentación de este hueco. Si el bloque es mayor que el hueco, el algoritmo de ubicación deberá trasladar los bloques que estén en memoria para crear un hueco lo suficientemente grande. Los algoritmos más usados son:

- ✓ *Algoritmo del mejor ajuste: El proceso se coloca en le hueco que deje menor cantidad de espacio libre.*
- ✓ *Algoritmo del peor ajuste: El proceso se coloca en el hueco que deje mayor cantidad de espacio libre. Mayor posibilidad de que otro proceso ocupe el hueco dejado.*
- ✓ *Algoritmo del primer ajuste: El proceso se coloca en el primer hueco que se encuentre que quepa.*

b) ***Políticas de ubicación para sistemas paginados**: Para colocar k páginas, sólo hay que recurrir al algoritmo de sustitución para liberar k páginas físicas.*

6.2.6. Jerarquía de almacenamiento.

Las unidades de almacenamiento se pueden clasificar jerárquicamente atendiendo a su tiempo de acceso. Cuanto más bajo sea el tiempo de acceso, es decir, se acede más rápidamente, más cara es la memoria. Por lo tanto, a mayor nivel en la jerarquía de la memoria, el tiempo de acceso es más lento y la cantidad de memoria es mayor.

NIVEL	TIPO DE MEMORIA		
1	Registros	Menor tiempo de acceso	Menor tamaño
2	Caché		
3	Memoria principal		
4	Caché de disco		
5	Memoria auxiliar (discos duros, CD-DVD, memoria flash)	Mayor tiempo de acceso	Mayor tamaño

Actualmente los ordenadores tienen una memoria de almacenamiento temporal o memoria caché, para que cada vez que se accede a la memoria principal o a la memoria auxiliar, la posición a la que se accede más las adyacentes, se copian en al caché, por la probabilidad que existe de que si se accede a una zona de memoria es frecuente que se acceda posteriormente a sus posiciones más cercanas.

6.3. GESTIÓN DE E/S.

El sistema operativo debe gestionar los dispositivos E/S, como periféricos y las memorias auxiliares, de manera que facilite su uso al usuario.

La gestión de E/S se realiza por el sistema operativo a través de las direcciones de E/S.

6.3.1. Interrupción y rutina de atención.

Una interrupción se produce cuando algún elemento hardware produce una señal al sistema para llamar su atención. Se las llama IRQ (Interrupt Request), y tienen como función interrumpir el trabajo del procesador para destinarlo a otra actividad. Ejemplos de interrupciones son las producen el ratón y el teclado cada vez que se utilizan.

6.3.2. Acceso directo a memoria (DMA).

El acceso directo a memoria, DMA (Direct Memory Access), se realiza por ciertos periféricos cuando la cantidad de información que quiere transferir es grande. Consiste en suprimir el acceso mediante las IRQ y las direcciones de E/S, para realizar la transferencia de información a través de unas líneas especiales llamadas DRQ (DMA Request), con lo cual el acceso a memoria se hace más rápidamente, dejando libre al procesador para que siga haciendo otras tareas.

Las tarjetas gráficas y de sonido utilizan el acceso directo a memoria, al igual que los discos duros, que son los dispositivos que más cantidad de información transfieren. El objetivo es liberar al procesador de la carga de trabajo, con lo cual el sistema funciona más rápida y eficazmente.

6.3.3. Caching, buffering y spooling.

Vamos a ver distintas técnicas que mejoran el rendimiento del sistema con respecto a la gestión de E/S:

- **Caching**. Mejora las prestaciones del sistema. Consiste en almacenar en una caché temporal, de rápido acceso, los datos más frecuentemente solicitados o enviados a un dispositivo de E/S. Se ha utilizado desde hace tiempo, y actualmente en todos los ordenadores se encuentran distintos niveles de memoria caché lo que evita distintos elementos de retardo como: transmisión de la información desde un dispositivo hacia otro dentro de la red, congestión del dispositivo de almacenamiento por tener que estar enviando información.
- **Buffering**. Esta técnica consiste en utilizar un área de memoria como buffer, simulando un dispositivo o periférico lógico, que hará de dispositivo intermedio entre el periférico real y el procesador. El buffer es independiente del dispositivo de entrada y/o salida, por lo que permite que el procesador comience a trabajar leyendo o almacenando en el buffer mientras la información del periférico se va almacenando o extrayendo del buffer. Esto evita que un periférico lento afecte al rendimiento del equipo informático.
- **Spooling**. La palabra deriva de las siglas en inglés de SPOOL (Simultaneous Peripheral Operation On.line), que viene a significar que mediante esta técnica la E/S hacia los periféricos del ordenador pueden simultanear.

6.4. GESTIÓN DE ARCHIVOS.

Un archivo o fichero es un objeto que representa la unidad lógica de almacenamiento de información. Se representan mediante un nombre. Los directorios se pueden considerar un tipo de fichero especial que contiene a otros ficheros o directorios.

Los ficheros se caracterizan mediante una serie de atributos como el nombre, el tamaño, la fecha de creación y modificación, propietario, permisos (lectura, escritura, ejecución), dirección o direcciones de la memoria secundaria donde está almacenado, ya que un fichero no tiene por qué estar físicamente en direcciones contiguas de memoria secundaria.

Con un fichero se pueden realizar una serie de operaciones, como crear, abrir, leer, escribir, cerrar, borrar y obtener información de él.

Con un directorio se puede crear, entran en él, salir de él, borrarlo, añadir o eliminar en él más archivos y directorios, leer su contenido.

6.4.1. Sistema de archivos.

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

Los sistemas de archivos organizan los archivos en bloques de varios bytes, y después almacenan la información de qué bloques están libres y la dirección física de cada bloque dentro del dispositivo de almacenamiento.

Los sistemas de archivos se pueden clasificar en cuatro categorías:

- De **disco**
- De **red**: para que un equipo pueda trabajar con un sistema de archivos e otro equipo como si fuera local (nfs) o compartir archivos por la red (smbfs)
- **Especiales**: swap, para la partición de intercambio de memoria en Linux.
- **Virtuales**: para que una determinada aplicación pueda tener acceso a diferentes sistemas de archivos (virtual FUSE, vfat, …)

Los sistemas de archivos más utilizados son:

a. **FAT:** taba de asignación de ficheros (File Allocation Table). Sistema de archivos introducido a partir del MS-DOS. Existen dos tipos de sistemas de archivos FAT.
   - **FAT16:** originalmente se llamaba simplemente FAT, pero después se llamó FAT16 para diferenciarlo de su sucesor FAT32. Tenía limitaciones, como el tamaño de las particiones, que no podría ser superior a 2Gb, y el nombre de los ficheros, que tenía que ser de 8 caracteres para el nombre y 3 caracteres para la extensión.
   - **FAT32:** este sistema de archivos se introdujo para el sistema operativo Windows 95, y fue usado por Windows 98 y Me. Es una mejora del anterior, al poder soportar particiones de mayor tamaño, el nombre de los archivos puede tener más caracteres y los archivos pueden ser de mayor tamaño.

b. **NTFS:** sistema de archivos de nueva tecnología. Mejora el sistema de ficheros al introducir mayor seguridad, mayor estabilidad, mayor tamaño de los archivos. Se utilizan para sistemas operativos Windows NT, 2000, XP, Vista, 7 y los Server 2000, 2003 y 2008.

c. **EXT2, EXT3, EXT4:** sistemas de archivos soportados por la mayoría de las distribuciones Linux. La versión ext3 es una mejora de la ext2, con algunas características nuevas, como el **journaling**, y el ext4 es una mejora de la ext3, con más mejoras como soportar volúmenes y ficheros más grandes, mejora de la fiabilidad, rapidez, … aunque los tres sistemas de archivos son compatibles entre sí.

El **journaling** es una técnica en la que se registra a diario los cambios en el sistema de archivos para poder recuperar los datos en caso de fallo.

Los sistemas de archivos ext2, ext3 y ext4 utilizan lo que se conoce como inodo (i-node). Cada fichero se identifica por un número entero de i.nodo único en el sistema de archivos. Es una tabla con estructura de datos para cada fichero, donde se recoge dónde está almacenado, su tamaño, dirección de los bloques usados, propietario, fecha de creación, fecha de modificación, …

d. **REISERFS:** es un sistema de ficheros soportado por algunas distribuciones de Linux. A partir de la nueva versión, Reiser4, dejó de ser soportado por más distribuciones Linux.

e. **CDFS:** sistemas de archivos de CD-ROM (sistema de archivos compatible con la norma ISO-9660).

f. **UDF:** formato de disco universal (Universal Disk Format) es el sistema de archivos definido por la norma ISO-13346, utilizado por las grabadoras de CD, DVD y Blu Ray.

g. **HFS:** sistema de archivos jerárquico (Hierarquical File System). Es el sistema de archivos desarrollado por Apple para su uso con el sistema operativo Mac OS. Se diseñó para ser usado con disquetes, discos duros y unidades de CD-ROM.

h. **HFS+:** sustituyó al anterior. Tiene algunas mejoras, como el journaling, el sistema de codificación de caracteres que utiliza es el Unicode para el nombre de los archivos y estos pueden ser de mayor tamaño que en el HFS.

6.5. GESTIÓN DE DISPOSITIVOS

6.5.1. Métodos de asignación de archivos

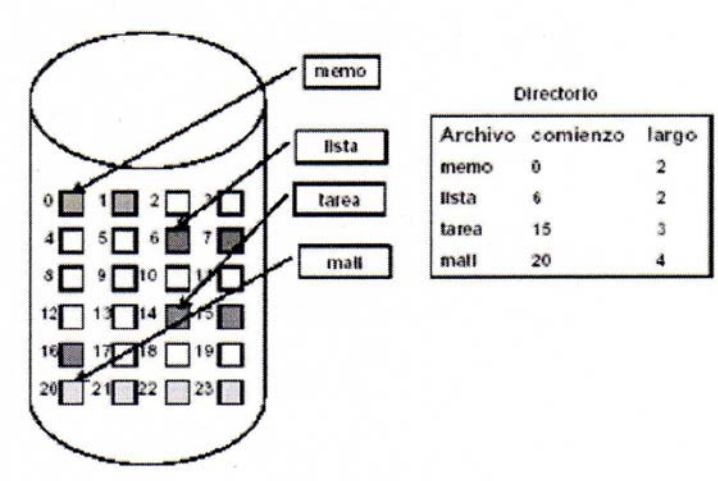
Un disco duro es como una secuencia de bloques (de 0 a n-1) de tamaño fijo a los que se tiene acceso directo. El movimiento del brazo hace más lento leer dos bloques separados que dos contiguos.

Existen diferentes formas de cómo almacenar los archivos en disco, alguna de las cuales son las siguientes.

6.5.1.1.- Asignación contigua

Este método requiere que cada archivo se almacene en un conjunto contiguo de bloques. De esta forma es fácil implementar el acceso directo. Además, se tiene un buen rendimiento cuando los archivos se leen enteros de forma secuencial ya que el movimiento del brazo del disco es el mínimo posible.

El problema aparece cuando el archivo crece, ya que no se sabe cuanta memoria se debe reservar de antemano; otro problema sería la existencia de fragmentación externa.



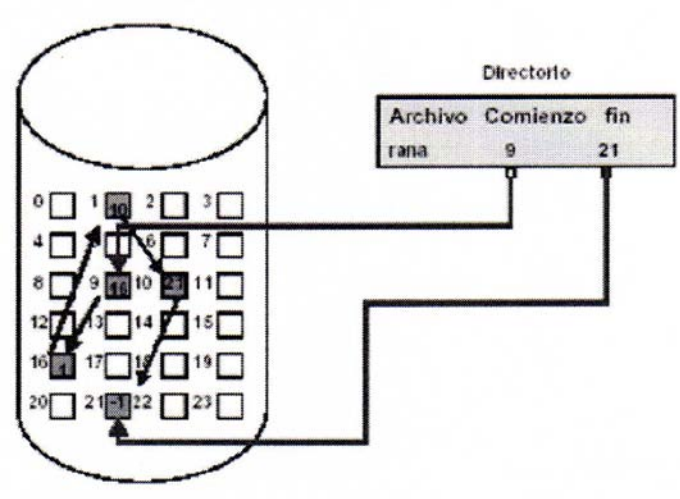
Directorio

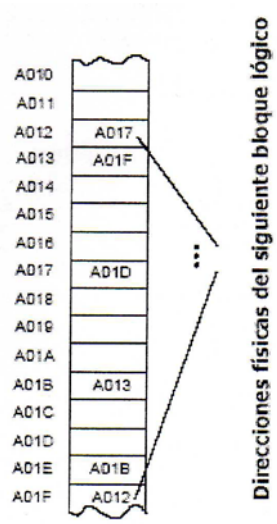
Archivo	comienzo	largo
memo	0	2
lista	6	2
tarea	15	3
mail	20	4

Los espacios vacíos están reservados por si el archivo crece.

6.5.1.2.- Lista encadenada

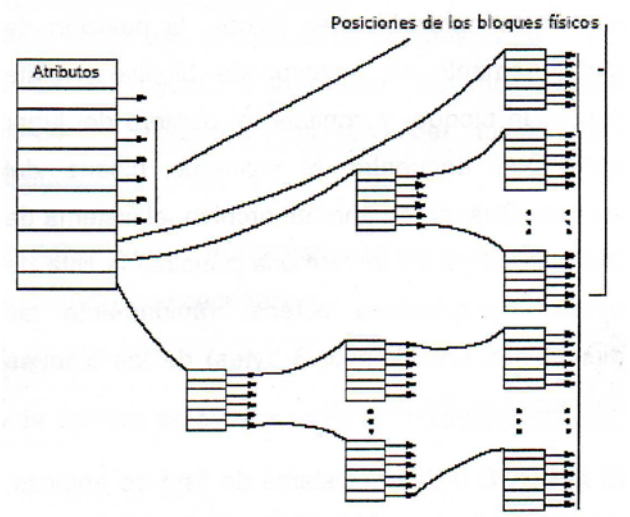
Los archivos se almacenan en bloques no necesariamente consecutivos, encadenándose un bloque con otro mediante un puntero. Se accede al fichero conociendo sólo la dirección del bloque inicial. Con este sistema no hay fragmentación, pero tiene el problema de que para acceder a n bloque es necesario acceder a él secuencialmente, no existiendo acceso directo a bloque.





6.5.1.3.- Lista de enlaces

Cada disco tiene una tabla con tantos elementos como bloques, donde la posición de cada elemento se corresponde con cada bloque y contiene el puntero donde se encuentra el siguiente bloque del 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 bloque consecutivos del archivo. WINDOWS 98 usa este sistema mediante una FAT (tabla de localización de archivos).



6.5.1.4.- I-nodos (nudo de índices)

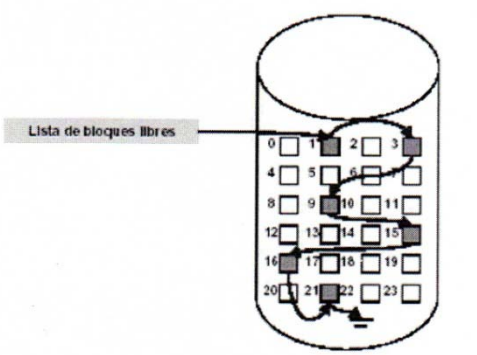
Es la forma de gestionar los archivos del Sistema Operativo UNIX. Cada archivo tiene asociado un nudo de índices, que es una pequeña tabla de tamaño fijo que contiene los atributos del archivo y las direcciones de un número determinado de los primeros bloques del archivo. Los tres últimos elementos de la tabla indican las siguientes direcciones de los bloques del archivo, pero de forma indirecta: con simple dirección, doble dirección y triple dirección respectivamente.

6.5.2. - Gestión del espacio libre

Para gestionar el espacio libre de un disco hay que disponer además de la tabla de asignación de archivos, de otra tabla o lista donde se registren los bloques que hay libres. Se pueden utilizar dos métodos para tener controlado el espacio libre del disco:

a) *Mediante el uso de un mapa de bits*: Se usa un mapa de bits de los bloques libres, que consistirá en un mapa con tantos bits como bloques tenga el disco y que almacenará el valor 1 si el bloque está libre y un 0 si el bloque está ocupado (o viceversa). Este método no requiere mucho espacio, por lo que podrá mantenerse en memoria principal. Es muy efectivo para localizar bloques libres contiguos.

b) *Mediante listas enlazadas*: Consiste en usar una lista enlazada de bloques que almacenarán la dirección de cuantos bloques libres queden, haciendo que un puntero señale siempre al primer bloque de esta lista. Es eficiente si el disco está casi lleno, sino ocupa demasiado espacio.



6.6. PLANIFICACIÓN DE DISCOS

6.6.1. - Parámetros de rendimiento del disco

a) *Tiempo de búsqueda*: Tiempo necesario para mover el brazo del disco hasta llegar a la pista solicitada. El tiempo de búsqueda consta de dos componentes claves: el tiempo de arranque inicial y el tiempo que se tarda en recorrer las pistas una vez que el brazo haya cogido la velocidad. El tiempo de recorrido no es una función lineal del número de pistas, sino que incluye un tiempo de arranque y un tiempo de colocación, que es el tiempo transcurrido desde que la cabeza se sitúa en el comienzo de una pista hasta que la identificación de la pista se confirma.

b) *Retardo de giro*: Los discos magnéticos, excepto los flexibles, tienen una velocidad de rotación que está entre 5400 y 10000 rpm. A una velocidad de 10000 rpm, el retardo medio de giro es de 3 ms. Los discos flexibles tienen un retardo medio entre 100 y 200 ms.

c) *Tiempo de transferencia*: El tiempo de transferencia con el disco depende de la velocidad de rotación, y se calcula usando la siguiente fórmula: ***T = b / (r * N)***

Donde, ***T*** es el tiempo de transferencia, ***b*** es el nº de bytes a transferir, ***r*** es la velocidad de rotación en revoluciones por segundo, y ***N*** es el nº de bytes por pista.

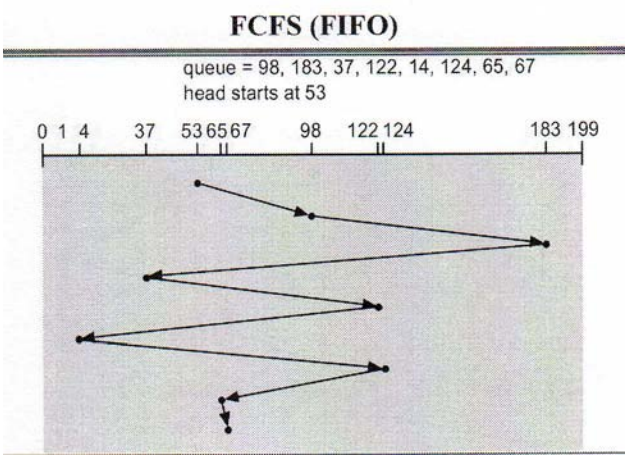
6.6.2. - Políticas de planificación del disco

En los sistemas basados en multiprogramación se producen peticiones de leer o escribir mucho más rápidas de lo que pueden atender los discos. Por esta razón, es frecuente encontrar colas de espera en los dispositivos de disco.

Existen muchos algoritmos de planificación de colas, unos tratando de reducir el tiempo de búsqueda y otros, mejorando el tiempo de latencia (el tiempo de rotación de las cabezas, tiempo para que el disco gire hasta ponerse sobre el sector deseado).

6.6.2.1.- Algoritmo Primero en entrar, primero en salir (FIFO o FCFS)

Los elementos que se encuentran en la cola, se procesan en orden secuencial, es decir, las solicitudes son atendidas en el orden de llegada. Esta política tiene un buen rendimiento si la mayoría de las solicitudes son a sectores agrupados de un archivo.



6.6.2.2.- Algoritmo Prioridad (PRI)

Este algoritmo no persigue la optimización del uso del disco, sino cumplir con otros objetivos del Sistema Operativo.

Permite que el sistema haga salir más rápidamente a muchos trabajos cortos y pueda proporcionar un buen tiempo de respuesta interactiva. Sin embargo, los trabajos más largos pueden tener que esperar excesivamente.

Es una estrategia poco favorable para sistemas de bases de datos.

6.6.2.3.- Algoritmo Último en entrar, primero en salir (LIFO)

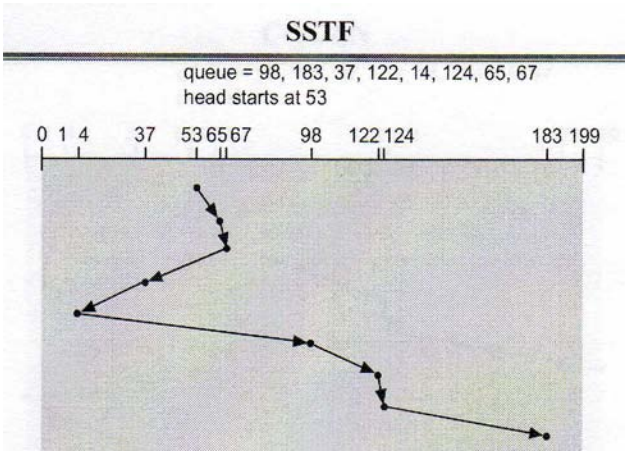
Esta estrategia toma la solicitud más reciente. Debido a esto, conceder el dispositivo al último usuario, hace que se tengan pocos movimientos del brazo en el recorrido de un archivo secuencial.

Sin embargo, si el disco está ocupado por una carga de trabajo larga, puede producirse inanición.

Tanto FIFO como LIFO sólo tienen en cuenta la cola, sin embargo, si se conoce la posición de la pista actual, puede usarse una planificación en función del elemento demandado. Las políticas que vienen a continuación usan esta posibilidad.

6.6.2.4.- Algoritmo Primero el tiempo de servicio más corto (SSTF)

Esta política elige la solicitud de E/S a disco que requiera el menor movimiento posible del brazo del disco desde su posición actual. De esta forma, siempre se elige usando el mínimo tiempo de búsqueda. Ofrece un rendimiento mejor que FIFO. También puede provocar inanición.



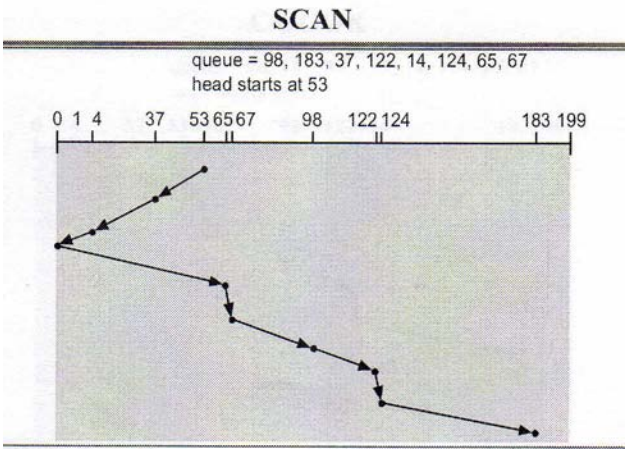
6.6.2.5.- Algoritmo del ascensor (SCAN)

Con la excepción de FIFO, todas las políticas descritas hasta ahora pueden dejar alguna solicitud incumplida hasta que se vacíe la cola entera, es decir, pueden llegar siempre nuevas solicitudes que se elegirán antes que una solicitud existente. Una alternativa que previene la inanición es la que propone el algoritmo SCAN.

Con SCAN, el brazo sólo se puede mover en un sentido, resolviendo todas las solicitudes pendientes de su ruta, hasta que alcance la última pista o hasta que no haya más solicitudes. Cuando llega al extremo del disco, cambia entonces la dirección de servicio y el rastro sigue en sentido opuesto, volviendo a recoger todas las solicitudes en orden.

Esta estrategia pretende evitar desplazamientos atrás y delante de la cabeza.

Esta política favorece a los trabajos con solicitudes de pistas más cercanas a los cilindros más interiores y exteriores, y a los últimos trabajos en llegar. El primer problema se soluciona con la versión C-SCAN.

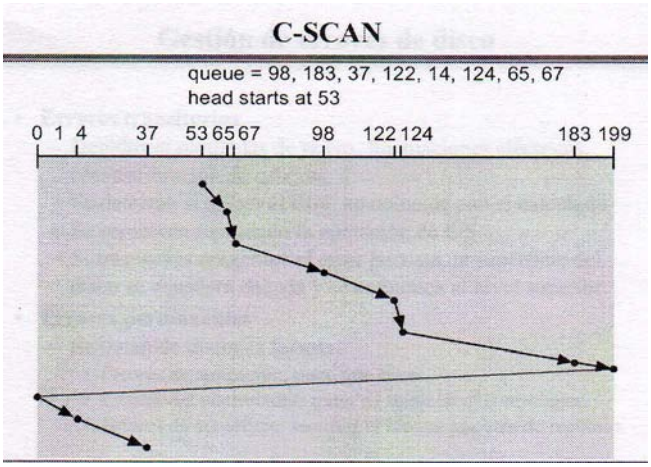


6.6.2.6.- Algoritmo del ascensor cíclico (C-SCAN)

Esta política restringe el rastreo a una sola dirección. Cuando se haya visitado la última pista en un sentido, el brazo vuelve al extremo opuesto del disco y comienza a recorrerlo de nuevo. Cuando llega al final, la cabeza vuelve inmediatamente al principio sin servir ninguna petición para comenzar de nuevo el recorrido.

Esta estrategia educe el retardo máximo sufrido por las nuevas solicitudes. Tiene un tiempo de espera más uniforme que el SCAN.

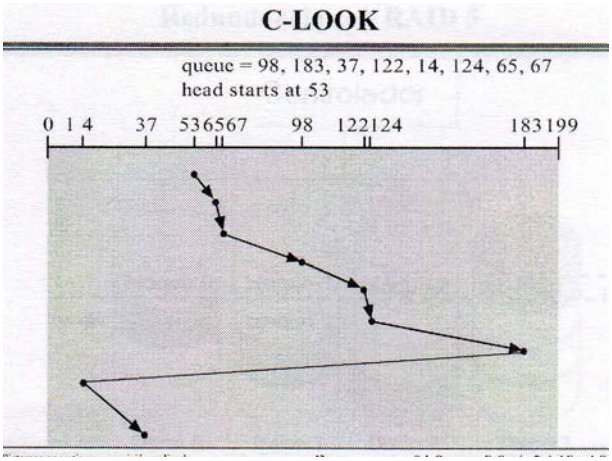
Es el algoritmo estándar que se usa en casi todos los sistemas.



6.6.2.7.- Algoritmo LOOK y C-LOOK

Son planificadores SCAN en los que la cabeza de lectura/escritura no llega hasta los extremos del disco, sino sólo hasta que no queden más solicitudes en la dirección actual en la que se está moviendo la cabeza. LOOK es equivalente a SCAN y C-LOOK a C-SCAN.

El planificador C-LOOK, cuando termina en una dirección, no vuelve al principio, sino que empieza en la petición más próxima al principio.



6.7. GESTIÓN DE LA SEGURIDAD.

El sistema operativo se debe encargar de controlar el acceso de los procesos o usuarios a los recursos del sistema.

El primer virus conocido fue un programa llamado Creeper (enredadera). Atacó una máquina IBM360 y lo creó en 1972 Robert Thomas Morris.

Era un programa que cada cierto tiempo mostraba por pantalla el mensaje: "I'm a creeper… catch me if you can!" (¡Soy una enredadera…agárrame si puedes!). Para eliminarlo se creó otro programa, el primer antivirus conocido, cuyo nombre fue Reaper (Segadora)

Para ello debe tomar medidas de seguridad. Cada vez los sistemas operativos controlan más la seguridad del equipo informático ya sea por parte de un error, por un uso incorrecto del sistema operativo o del usuario, o bien por un acceso no controlado físicamente o a través de la red, o por un programa maligno, como los virus, espías, troyanos, gusanos, phising, …

Es casi imposible que sea totalmente seguro, pero se puede tomar ciertas medidas para evitar daños a la información o a la privacidad de esta. Uno de los principales peligros a un sistema informático le puede venir por Internet, también por compartir información con otro equipo por la red o a través de un archivo infectado que entre en el sistema mediante una memoria secundaria, como un USB, DVD, disco duro externo, …

Los servicios del sistema operativo pueden ser otro punto especial de ataque, ya que son procesos que están en ejecución y que suelen tener privilegios especiales.

Para mantener la seguridad en un sistema informático se pueden utilizar diversas técnicas, como el uso de contraseñas, encriptación de la información, uso de programas antivirus, cortafuegos, …

Además de la seguridad mediante el sistema operativo, o **seguridad lógica**, es necesario tener en cuenta la **seguridad física** del sistema informático, como uso de SAI (UPS), guardias de seguridad del edificio, cámaras de seguridad, controles de acceso al edificio como tarjetas de identificación, …

Los **SAI** (Sistema de Alimentación Ininterrumpida), también conocidos por su siglas en inglés, UPS (Uninterruptible Power Supply), son dispositivos que tras un corte de suministro eléctrico proveen esta a los equipos del sistema informático, además, proporciona seguridad ante subidas de tensión en la red.