Sistema Operativo

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

Los principales objetivos de los sistemas operativos son:

- **Abstraer al usuario de la complejidad del hardware:** El sistema operativo hace que el ordenador sea más fácil de utilizar.
- **Eficiencia**: Permite que los recursos del ordenador se utilicen de la forma más eficiente posible. Por ejemplo, se deben optimizar los accesos a disco para acelerar las operaciones de entrada y salida.
- **Permitir la ejecución de programas**: Cuando un usuario quiere ejecutar un programa, el sistema operativo realiza todas las tareas necesarias para ello, tales como cargar las instrucciones y datos del programa en memoria, iniciar dispositivos de entrada/salida y preparar otros recursos.
- **Acceder a los dispositivos entrada/salida**: El sistema operativo suministra una interfaz homogénea para los dispositivos de entrada/salida para que el usuario pueda utilizar de forma más sencilla los mismos.
- **Proporciona**r una estructura y conjunto de operaciones para el sistema de archivos.
- **Controlar el acceso al sistema y los recursos**: en el caso de sistemas compartidos, proporcionando protección a los recursos y los datos frente a usuarios no autorizados.
- **Detección y respuesta ante errores**: El sistema operativo debe prever todas las posibles situaciones críticas y resolverlas, si es que se producen.
- **Capacidad de adaptación**: Un sistema operativo debe ser construido de manera que pueda evolucionar a la vez que surgen actualizaciones hardware y software.
- **Gestionar las comunicaciones en red**: El sistema operativo debe permitir al usuario manejar con facilidad todo lo referente a la instalación y uso de las redes de ordenadores.
- **Permitir a los usuarios compartir recursos y datos**: Este aspecto está muy relacionado con el anterior y daría al sistema operativo el papel de gestor de los recursos de una red.

Podemos decir que sobre procesamiento de la información al cual le sumamos un conjunto de estructuras, como: políticas, estratégicas, seguridad y objetivos; más los conflictos que deben ser resueltos con eficiencia; podemos definir el término Sistema Operativo.

Se pueden imaginar un Sistema Operativo como los programas, instalados en un sistema computacional o en un firmware, que hacen utilizable al hardware. El hardware proporciona la "capacidad bruta de cómputo"; los sistemas operativos ponen dicha capacidad de cómputo al alcance de los usuarios o programas mediante servicios y administran cuidadosamente los recursos del sistema para lograr un buen rendimiento.

Los Sistemas Operativos son básicamente administradores de recursos. El principal recurso que administran es el hardware del computador; además de los procesadores, los medios de almacenamiento, los dispositivos de entrada/salida, los dispositivos de comunicación, administran los procesos, el procesamiento y los datos.

Un Sistema Operativo es un programa que actúa como intermediario o interfase entre el usuario y el hardware del computador y su propósito es proporcionar el entorno en el cual el usuario pueda ejecutar programas. Entonces, el objetivo principal de un Sistema Operativo es, lograr que el sistema de computación se use de manera adecuada y cómoda, y el objetivo secundario es que el hardware del computador se emplee de manera eficiente.

Evolución histórica de los sistemas operativos.

El hardware y el software de los sistemas informáticos han evolucionado de forma paralela y conjunta en las últimas décadas. Por lo que la evolución que vamos a ver de los sistemas operativos está estrechamente relacionada con los avances en la arquitectura de los ordenadores que se produjo de cada generación.

Primera generación (1945-1955)

Los primeros ordenadores estaban construidos con tubos de vacío. En un principio no existían sistemas operativos, se programaba directamente sobre el hardware. Los programas estaban hechos directamente en código máquina y el control de las funciones básicas se realiza mediante paneles enchufables.

Hacia finales de 1950 aparecen las tarjetas perforadas que sustituyen los paneles enchufables. Las tarjetas perforadas supusieron un enorme paso ya que permitían codificar instrucciones de un programa y los datos en una cartulina con puntos que podía interpretar el ordenador. La mayoría de los programas usaban rutinas de E/S y un programa cargador (automatizaba la carga de programas ejecutables en la máquina) esto constituía una forma rudimentaria de sistema operativo.

2ª Generación (1955-1965)

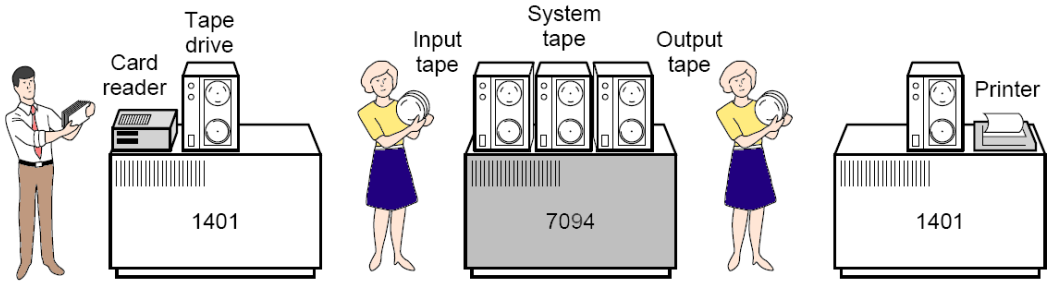
Esta generación se caracteriza por la aparición de los transistores que permitieron la construcción de ordenadores más pequeños y potentes. La programación se realizaba en lenguaje ensamblador y en FORTRAN sobre tarjetas perforadas. Otro aspecto importante de esta generación es el procesamiento por lotes, en el cual mientras el sistema operativo está ejecutando un proceso, éste último dispone de todos los recursos hasta su finalización. La preparación de los trabajos se realiza a través de un lenguaje de control de trabajos conocido como JCL. El sistema operativo residía en memoria y tenía un programa de control que interpretaba las tarjetas de control, escritas JCL. Dependiendo del contenido de la tarjeta

de control el sistema operativo realizaba una acción determinada. Este programa de control es un antecedente de los modernos intérpretes de órdenes.

Procesamiento Fuera de línea (Offline)

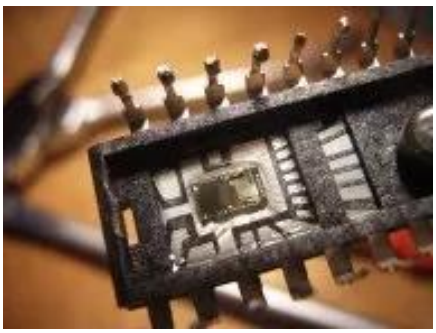
Como mejora del procesamiento por lotes surgió el procesamiento fuera de línea (off-line), en el cual las operaciones de carga de datos y salida de resultados de un proceso podían realizarse de forma externa y sin afectar al tiempo que el procesador dedicaba a los procesos. A esto ayudó la aparición de las cintas magnéticas y las impresoras de líneas. Ejemplos de sistemas operativos de la época son FMS (Fortran Monitor System) y IBSYS.

3ª



Generación (1965-1980)

La aparición de los circuitos integrados (CI) supuso una mejora consiguiendo un menor tamaño y relación precio/rendimiento respecto de las máquinas de generaciones anteriores. En relación con los sistemas operativos, la característica principal de esta generación fue el desarrollo de la multiprogramación y los sistemas compartidos. En los sistemas multiprogramados se cargan varios programas en memoria simultáneamente y se alterna su ejecución. Esto maximiza la utilización del procesador. Como evolución de aparecen los sistemas de tiempo compartido donde el tiempo del procesador se comparte entre programas de varios usuarios pudiendo ser programas interactivos. Algunos de los sistemas operativos de esta generación son OS/360, CTSS, MULTICS y UNIX.



4ª Generación (1980-hasta hoy)

En esta generación se producen grandes avances en la industria hardware como la creación de los circuitos LSI (integrados a gran escala). También aparecen los ordenadores personales, entre finales de la anterior generación y principios de la presente. Ejemplos de sistemas operativos de los primeros ordenadores personales son MS-DOS, desarrollado por Microsoft, Inc., para el IBM PC y MacOS de Apple

Computer, Inc. Steve Jobs, cofundador de Apple, apostó por la primera interfaz gráfica basada en ventanas, iconos, menús y ratón a partir de una investigación realizada por Xerox. Siguiendo esta filosofía aparecería MS Windows. Durante los 90 apareció Linux a partir del núcleo desarrollado por Linus Torvalds. Los sistemas operativos evolucionan hacia sistemas interactivos con una interfaz cada vez más amigable al usuario. Los sistemas Windows han ido evolucionando, con diferentes versiones tanto para escritorio como para servidor (Windows 3.x, 98, 2000, XP, Vista, 7, Windows Server 2003, 2008, etc), al igual que lo han hecho Linux (con multitud de distribuciones, Ubuntu, Debian, RedHat, Mandrake, etc) y los sistemas Mac (Mac OS 8, OS 9, OS X, Mac OS X 10.6 "Snow Leopard", entre otros).

Un avance importante fue el desarrollo de redes de ordenadores a mediados de los años 80 que ejecutan sistemas operativos en red y sistemas operativos distribuidos. En un sistema operativo en red los usuarios tienen conocimiento de la existencia de múltiples ordenadores y pueden acceder a máquinas remotas y copiar archivos de un ordenador a otro. En un sistema distribuido los usuarios no saben dónde se están ejecutando sus programas o dónde están ubicados sus programas, ya que los recursos de procesamiento, memoria y datos están distribuidos entre los ordenadores de la red, pero todo esto es transparente al usuario.

Actualmente, existen sistemas operativos integrados, para una gran diversidad de dispositivos electrónicos, tales como, teléfonos móviles, PDAs (Personal Digital Assistant, Asistente Digital Personal u ordenador de bolsillo), otros dispositivos de comunicaciones e informática y electrodomésticos. Ejemplos de este tipo de sistemas operativos son PalmOS, WindowsCE, Android OS, etc. Haremos una referencia especial al último, Android OS, se trata de un sistema operativo basado en Linux. Fue diseñado en un principio para dispositivos móviles, tales como teléfonos inteligentes y tablets, pero actualmente se encuentra en desarrollo para su aplicación también en netbooks y PCs.

Tipos de sistemas operativos.

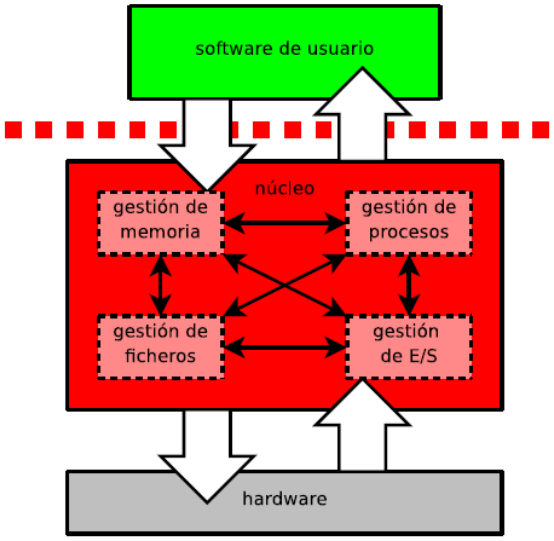
Ahora vamos a clasificar los sistemas operativos en base a su estructura, servicios que suministran y por su forma.

Tipos de sistemas operativos

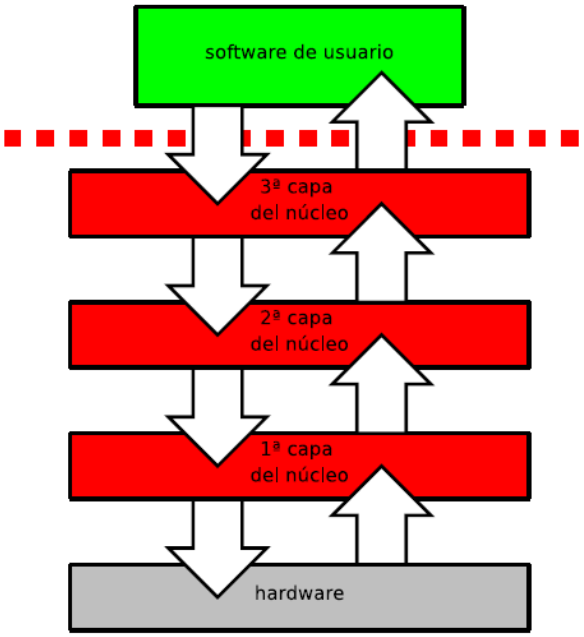
Por estructura	Por sus servicios	Por su forma
Monolíticos	Monousuario	Sistema operativo en red
Jerárquicos	Multiusuario	Sistema operativo distribuido
Máquina Virtual	Monotarea	
Microkernel o Cliente-Servidor	Multitarea	
Monolíticos	Monoprocesador	
	Multiprocesador	

A) Sistemas operativos por su estructura

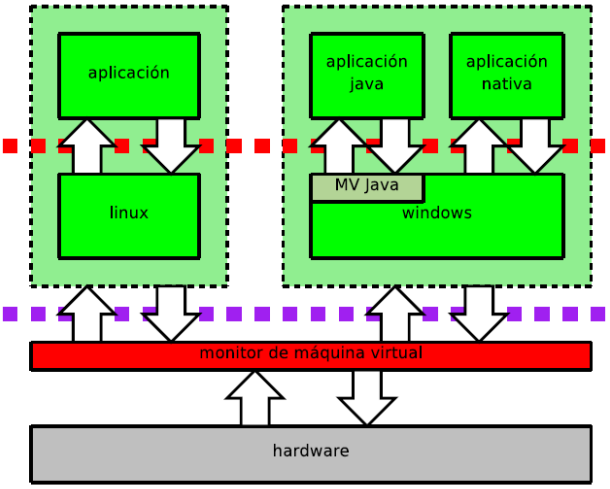
Monolíticos: Es la estructura de los primeros sistemas operativos, consistía en un solo programa desarrollado con rutinas entrelazadas que podían llamarse entre sí. Por lo general, eran sistemas operativos hechos a medida, pero difíciles de mantener.



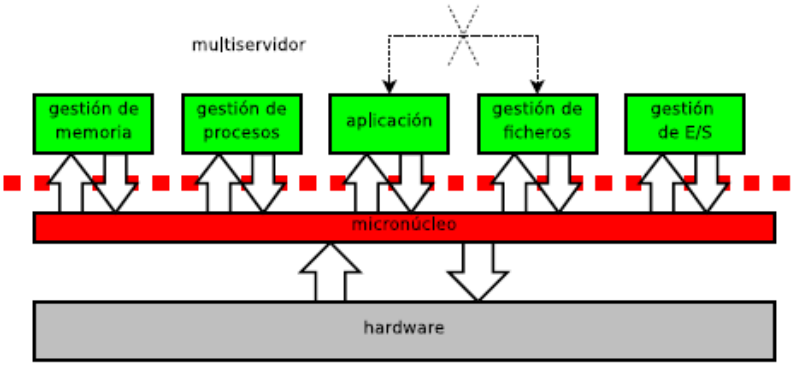
Jerárquicos: Conforme las necesidades de los usuarios aumentaron, los sistemas operativos fueron creciendo en complejidad y funciones. Esto llevó a que se hiciera necesaria una mayor organización del software del sistema operativo, dividiéndose en partes más pequeñas, diferenciadas por funciones y con una interfaz clara para interoperar con los demás elementos. Un ejemplo de este tipo de sistemas operativos fue MULTICS.



Máquina Virtual: El objetivo de los sistemas operativos es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes. Presentan una interfaz a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente. Estas máquinas no son máquinas extendidas, son una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario. VMware y VM/CMS son ejemplos de este tipo de sistemas operativos.



Microkernel o Cliente-Servidor: El modelo del núcleo de estos sistemas operativos distribuye las diferentes tareas en porciones de código modulares y sencillas. El objetivo es aislar del sistema, su núcleo, las operaciones de entrada/salida, gestión de memoria, del sistema de archivos, etc. Esto incrementa la tolerancia a fallos, la seguridad y la portabilidad entre plataformas de hardware. Algunos ejemplos son MAC OS X o AIX.



B) Sistemas operativos por sus servicios

Monousuario: Son aquellos que soportan a un usuario a la vez, sin importar el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Ejemplos de sistemas operativos de este tipo son MS-DOS, Microsoft Windows 9x y ME, MAC OS, entre otros.

Multiusuario: Son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas al ordenador o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que puede ejecutar cada usuario simultáneamente. Algunos ejemplos serán UNIX, GNU/Linux, Microsoft Windows Server o MAC OS X.

Monotarea: Sólo permiten una tarea a la vez por usuario. Se puede dar el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios simultáneamente pero cada uno de ellos puede ejecutar sólo una tarea en un instante dado. Ejemplos de sistemas monotarea son MS-DOS, Microsoft Windows 3.x y 95 (estos últimos sólo simulan la multitarea).

Multitarea: Permite al usuario realizar varias tareas al mismo tiempo. Algunos ejemplos son MAC OS, UNIX, Linux, Microsoft Windows 98, 2000, XP, Vista y 7.

Monoprocesador: Es aquel capaz de manejar sólo un procesador, de manera que si el ordenador tuviese más de uno le sería inútil. MS-DOS y MAC OS son ejemplos de este tipo de sistemas operativos.

Multiprocesador: Un sistema operativo multiprocesador se refiere al número de procesadores del sistema, éste es más de uno y el sistema operativo es capaz de utilizarlos todos para distribuir su carga de trabajo. Estos sistemas trabajan de dos formas: simétricamente (los procesos son enviados indistintamente a cualquiera de los procesadores disponibles) y asimétricamente (uno de los procesadores actúa como maestro o servidor y distribuye la carga de procesos a los demás).

C) Sistemas operativos por su forma

Sistemas operativos en red: Estos sistemas tienen la capacidad de interactuar con los sistemas operativos de otras máquinas a través de la red, con el objeto de intercambiar información, transferir archivos, etc. La clave de estos sistemas es que el usuario debe conocer la ubicación de los recursos en red a los que desee acceder. Los sistemas operativos modernos más comunes pueden considerarse sistemas en red, por ejemplo: Novell, Windows Server, Linux, etc.

Sistemas operativos distribuidos: Abarcan los servicios de red, las funciones se distribuyen entre diferentes ordenadores, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, etc.) en una sola máquina virtual que es a la que el usuario accede de forma transparente. En este caso, el usuario no necesita saber la ubicación de los recursos, sino que los referencia por su nombre y los utiliza como si fueran locales a su lugar de trabajo habitual. MOSIX es un ejemplo de estos sistemas operativos.

Spooling

En el campo de la Informática, el spooling (Simultaneous Peripheral Operations On-Line) se refiere al proceso mediante el cual la computadora introduce trabajos en un buffer (un área especial en memoria o en un disco), de manera que un dispositivo pueda acceder a ellos cuando esté listo.

El spooling es útil en caso de dispositivos que acceden a los datos a distintas velocidades. El buffer proporciona un lugar de espera donde los datos pueden estar hasta que el dispositivo (generalmente más lento) los procesa. Esto permite que la CPU pueda trabajar en otras tareas mientras que espera que el dispositivo más lento acabe de procesar el trabajo.

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

El uso de un almacenamiento intermedio permite que varios procesos en paralelo estén generando datos para el dispositivo, sin que se mezcle el resultado, ni que tengan que esperar a que finalice la operación

con el periférico. En consecuencia, se obtiene una comunicación indirecta entre los programas que escriben los datos y los que los leen. Se suele usar este mecanismo cuando un dispositivo escribe datos a diferente velocidad de la que la lee el dispositivo receptor, lo cual permite que un dispositivo más lento lo procese a su ritmo.

También se puede referir a un dispositivo de almacenamiento que incorpora un spool físico, como una unidad de cinta.

Los recursos que administra el sistema operativo

El sistema operativo necesita administrar los recursos para tener control sobre las funciones básicas del ordenador. Los principales recursos que administra el sistema operativo son:

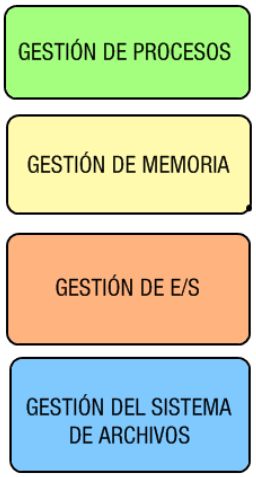
- **El procesador.**
- **La memoria.**
- **Los dispositivos de entrada/salida.**
- **El sistema de archivos.**

Núcleo

Para gestionar todos estos recursos, existe una parte muy importante del sistema operativo, el núcleo o kernel. El núcleo normalmente representa sólo una pequeña parte de todo lo que es el sistema operativo, pero es una de las partes que más se utiliza. Por esta razón, el núcleo reside por lo general en la memoria principal, mientras que otras partes del sistema operativo son cargadas en la memoria principal sólo cuando se necesitan.

Resumiendo, el núcleo supone la parte principal del código de un sistema operativo y se encarga de controlar y administrar los servicios y peticiones de recursos. Para ello se divide en distintos niveles:

- Gestión de procesos
- Gestión de memoria
- Gestión de la entrada/salida (E/S)
- Gestión del Sistema de archivos



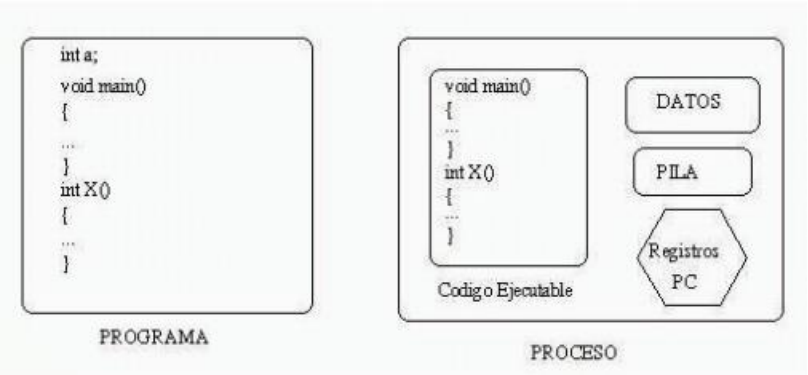
Entre las principales tareas del sistema operativo está la de **administrar los procesos del sistema**.

Un proceso en un programa en ejecución. Un proceso simple tiene un **hilo de ejecución** (o subproceso), en ocasiones, un proceso puede dividirse en varios subprocesos. Un **hilo** es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea. Por lo que los **hilos** de ejecución permiten a un programa realizar varias tareas a la vez.

Un proceso es una abstracción de un programa en ejecución, compuesto por el código ejecutable, una sección de datos que contiene las variables globales, una sección de stack o pila que contiene datos temporales, tales como parámetros de subrutinas, direcciones de retornos y variables temporales; y el estado de los registros del procesador. El programa corresponde a una entidad pasiva, en cambio el proceso corresponde a una entidad activa. Hablando de los recursos que utilizan ambos conceptos, podemos decir,

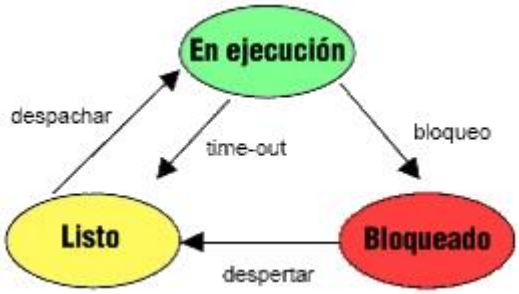
que el programa utiliza únicamente memoria secundaria, en cambio el proceso utiliza memoria principal y procesador.

En la siguiente imagen se puede apreciar ambos conceptos:



En los sistemas operativos modernos los procesos pueden tener diferentes estados, según el momento de creación, si están en ejecución, si se encuentran a la espera de algún recurso, etc. Pero podemos hacer una simplificación, y un proceso, en un instante dado, puede estar en uno de los tres estados siguientes:

- Listo.
- En ejecución.
- Bloqueado.



Los procesos en **estado listo** son los que pueden pasar a estado de ejecución si el planificador del sistema operativo los selecciona, esto es, cuando llegue su turno (según el orden de llegada o prioridad).

Los procesos en **estado de ejecución** son los que se están ejecutando en el procesador en un momento dado.

Los procesos que se encuentran en **estado bloqueado** están esperando la respuesta de algún otro proceso para poder continuar con su ejecución, por ejemplo, una operación de entrada/salida.

El sistema operativo sigue la pista de en qué estado se encuentran los procesos, decide qué procesos pasan a ejecución, cuáles quedan bloqueados, en definitiva, gestiona los cambios de estado de los procesos. Los procesos pueden comunicarse entre sí o ser independientes. En el primer caso, los procesos necesitarán sincronizarse y establecer una serie de mecanismos para la comunicación; por ejemplo, los procesos que pertenecen a una misma aplicación y necesitan intercambiar información. En el caso de procesos independientes estos, por lo general, no interactúan y un proceso no requiere información de otros.

Los procesos pueden tomar cualquiera de los estados mencionados y la transición entre uno y otro estado tiene su explicación. Las siguientes son las transiciones posibles:

1. **Ejecución - Bloqueado:** Un proceso pasa de ejecución a bloqueado cuando ejecuta una instrucción que implica la espera de un evento, por ejemplo, debe esperar que ocurra efectivamente la E/S, espera por la lectura de un registro en disco, imprimir un documento, etc.
2. **Listo - Ejecución**: Un proceso pasa de listo a ejecución cuando el sistema le otorga un tiempo de CPU.
3. **Ejecución - Listo:** Un proceso pasa de ejecución a listo, cuando se le acaba el tiempo asignado por el planificador de procesos del sistema, en este momento el sistema debe asignar el procesador a otro proceso.

4. **Bloqueado - Listo:** Un proceso pasa de bloqueado a listo cuando el evento externo que esperaba sucede.

Tabla de Procesos (PCB)

Para manejar la información de todos los procesos, el sistema operativo maneja una tabla de procesos, la que contiene una entrada por cada proceso. A cada una de estas entradas en la tabla de procesos se le conoce con el nombre de PCB (Process Control Block) o simplemente Bloque de Control de Procesos.

Por lo general esta estructura posee diversa información asociada al proceso, la que genéricamente incluye:

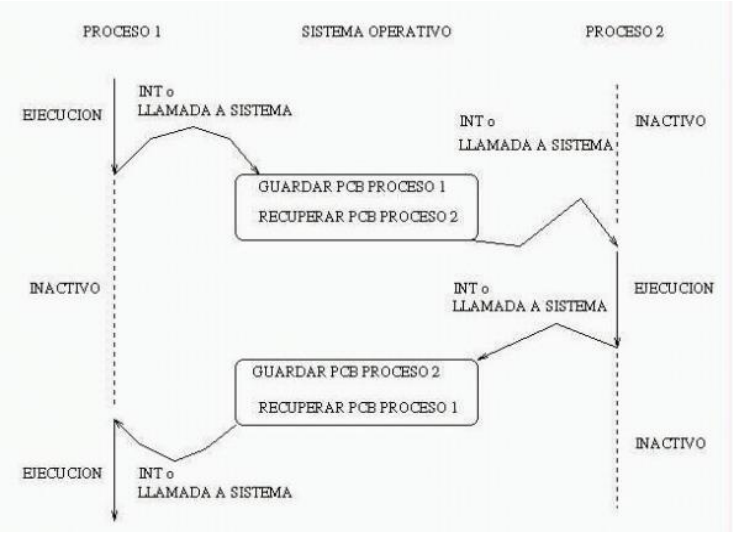
- Estado del proceso: El estado puede ser en ejecución, listo o bloqueado.
- Contador de programas: El PC contiene la dirección de la siguiente instrucción a ejecutar por el proceso.
- Información de planificación: Esta información incluye prioridad del proceso, apuntadores a colas de planificación, etc.
- Información contable: Esta información incluye cantidad de tiempo de CPU asignado, hora de inicio del proceso, etc.
- Información de planificación de memoria: Esta incluye información de registros límites de acceso, punteros a segmentos de datos, códigos, etc.
- Información del Sistema de archivos: Esta información incluye protecciones, identificación de usuario, grupo, etc.
- Información del estado de E/S2: Esta información incluye, solicitudes pendientes de E/S, dispositivos de E/S asignados al proceso, etc.

Cambio de Contexto

Otro concepto muy importante que incluye el concepto de proceso, es el denominado cambio de contexto. Este concepto está directamente relacionado con la idea de los sistemas de tiempo compartido. En un sistema de tiempo compartido, existen muchos procesos ejecutándose concurrentemente, el sistema

se encarga de asignar a cada uno de los procesos un tiempo fijo de ejecución o bien se ejecuta hasta una operación de E/S. Cuando el proceso no termina de ejecutarse en ese tiempo, el sistema debe guardar la información que le corresponde a dicho proceso para poder recuperarla posteriormente cuando le asigne otra cantidad de tiempo de ejecución.

Se denomina conmutación o cambio de contexto: al mecanismo mediante el cual el sistema almacena la información del proceso que se está ejecutando y recupera la información del proceso que ejecutará enseguida. A continuación, se presenta un esquema explicativo de cómo se cargan y guardan los PCB de los procesos.



Jerarquía de Procesos

El diseño y funcionamiento de un sistema operativo que maneje el concepto de proceso debe ser flexible y ofrecer capacidades a los programadores. En este sentido, debe proveer los mecanismos mediante los cuales los programadores puedan crear aplicaciones en donde puedan trabajar con más de un proceso. Para lograr esto, el sistema operativo proporciona llamadas al sistema mediante los cuales los programadores pueden crear o destruir procesos.

Particularmente, el sistema operativo UNIX proporciona **la llamada a sistema fork(),** la cual permite crear un proceso, que se ejecuta concurrentemente con el proceso que lo creó. Cuando un proceso crea a otro, al proceso creado se le denomina proceso hijo y al proceso que creó, se le denomina proceso padre. El nuevo proceso hijo también puede crear otros procesos. De esta manera es posible tener una jerarquía de procesos en donde se tienen (entre comillas), procesos abuelos, procesos padres y procesos hijos; o alguna jerarquía de mayor anidación.

EL sistema Unix, proporciona varias **llamadas al sistema** que le permite a los programadores desarrollar aplicaciones que manejen varios procesos. La llamada **fork**(), crea un proceso hijo, copiando toda la caracterización del proceso padre en el hijo (en otro espacio de dirección), es decir, el hijo resulta ser una copia del padre, con el mismo código, datos, pila y conjunto de registros.

Cuando se ejecuta el **fork**() retorna el valor 0 para el hijo y un valor mayor que cero para el padre, este valor corresponde al identificador del proceso **(pid)** hijo. Una vez que el proceso hijo es creado, tanto el padre como el hijo comienzan a ejecutarse en forma concurrente. Luego si alguno de estos procesos modifica sus respectivos datos, estos cambios no son reflejados en el otro proceso. Sin embargo, los recursos obtenidos por el padre son heredados al hijo, así, por ejemplo, si el padre antes de crear al hijo tenía un archivo abierto, éste permanecerá abierto en el hijo.

Si el proceso padre desea cambiar la imagen del proceso hijo, es decir, quiere asignarle otra tarea, entonces debe ejecutar otra llamada al sistema que le permita hacerlo. Esto es posible mediante la llamada exec y sus variantes.

- ✓ Hay un punto importante en la relación de procesos padres e hijos. El proceso padre no debe terminar antes que cualquiera de sus hijos, si esto sucede, entonces los procesos hijos quedan huérfanos (defunct o zombie). Este tipo de proceso no es posible de eliminar del sistema, sólo se pueden matar bajando el sistema. Existen otras llamadas al sistema que les permite a los procesos padres esperar por la muerte de sus hijos: wait, wait y waitpid. Por otro lado, cada hijo le avisa a su padre cuando termina a través de una llamada a sistema exit.

Planificación del procesador.

La planificación (scheduling) es la base para lograr la multiprogramación.

- ✓ Un sistema multiprogramado tendrá varios procesos que requerirán el recurso procesador a la vez.
- ✓ Esto sucede cuando los procesos están en estado ready (pronto).
- ✓ Si existe un procesador disponible y existen procesos en estado ready, se debe elegir el que será asignado al recurso para ejecutar.
- ✓ El componente del sistema operativo que realiza la elección del proceso es llamado planificador (scheduler).
- ✓ Despachador: módulo del SO que da el control de la CPU al proceso seleccionado por el planificador de corto plazo
  Esto implica
  - Cambio de contexto: Salvar registros del procesador en PCB del proceso saliente. Cargar los registros con los datos del PCB del proceso entrante.

- Cambiar el bit de modo a usuario.
- Saltar a la instrucción adecuada que había quedado el proceso que se asignó a la CPU (registro program counter).

✓ La latencia del despachador debe ser la menor posible
✓ El planificador es el responsable de seleccionar el próximo proceso a ejecutarse

En la planificación del procesador se decide cuánto tiempo de ejecución se le asigna a cada proceso del sistema y en qué momento. Si el sistema es monousuario y monotarea no habrá que decidir, pero en el resto de los sistemas multitarea esta decisión es fundamental para el buen funcionamiento del sistema, ya que determinará la correcta ejecución de los distintos programas de aplicación que se estén ejecutando.

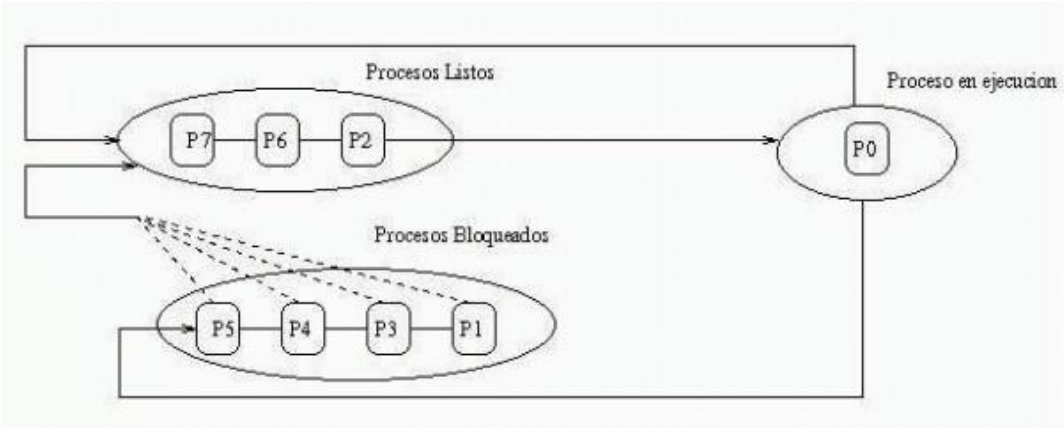
El control del sistema operativo debe asegurar que los procesos no sean interferidos unos con otros. Esto significa que el S.O. debe garantizar que los procesos tengan acceso a los recursos de memoria y CPU, y además debe asegurar la protección, de manera que un proceso no pueda arbitrariamente modificar el estado de otro.

La utilización de la multiprogramación tiene como consecuencia la mayor utilización de la CPU y una mayor productividad. La productividad es una medida que está relacionada con la cantidad de trabajos realizados en un rango de tiempo. De esta manera si un proceso se ejecuta completamente en 2 minutos,

pero durante ese tiempo el tiempo que espera por E/S es de 1,5 minutos, entonces se está desperdiciando 1,5 minutos de tiempo de CPU. Con la multiprogramación ese tiempo perdido puede ser aprovechado con la ejecución de otros procesos que requieran de atención de CPU.

El sistema operativo utiliza para la planificación de procesos Colas de planificación. Para explicar cómo el sistema operativo opera con colas, recordemos que los procesos pueden encontrarse en uno de tres estados (en ejecución, listo y bloqueado). En este sentido, si un proceso está en ejecución implica que está utilizando el recurso de CPU; si un proceso está en estado listo significa que está esperando por la CPU, pero como pueden ser más de un proceso, este estado permite que exista una cola de procesos que estén esperando por la CPU; si un proceso está bloqueado significa que espera por un evento externo, por lo que este estado también puede tener una cola de procesos que esperen por los eventos o acciones que les corresponden.

El siguiente diagrama que muestra cómo funciona:



En la imagen se aprecian los siguientes cambios con respecto al diagrama de estados de un proceso:

1. Si a un proceso en ejecución se le acaba su tiempo asignado pasa a la cola de procesos listos (retroalimentación)

2. Si un proceso en ejecución requiere esperar por un evento, pasa a la cola de bloqueados para ser atendido por el dispositivo de E/S que provoca el evento.
3. Si un proceso de la cola de bloqueados recibe el evento esperado, pasa a la cola de procesos listos.

El sistema operativo almacena en una tabla denominada tabla de control de procesos con la información relativa a cada proceso que se está ejecutando en el procesador. Ésta es:

- Identificación del proceso.
- Identificación del proceso padre.
- Información sobre el usuario y grupo que lo han lanzado.
- Estado del procesador. El contenido de los registros internos, contador de programa, etc. Es decir, el entorno volátil del proceso.
- Información de control de proceso.
- Información del planificador.
- Segmentos de memoria asignados.
- Recursos asignados.

Una **estrategia de planificación** debe buscar que los procesos obtengan sus turnos de ejecución de forma apropiada (momento en que se le asigna el uso de la CPU), junto con un buen rendimiento y minimización de la sobrecarga (overhead) del planificador mismo. En general, se buscan cinco objetivos principales:

- Todos los procesos en algún momento obtienen su turno de ejecución o intervalos de tiempo de ejecución hasta su terminación con éxito.
- El sistema debe finalizar el mayor número de procesos por unidad tiempo.
- El usuario no percibirá tiempos de espera demasiado largos.
- Evitar el aplazamiento indefinido, los procesos deben terminar en un plazo finito de tiempo. Esto es, el usuario no debe percibir que su programa se ha parado o “colgado”.

La carga de trabajo de un sistema informático a otro puede variar considerablemente, esto depende de las características de los procesos. Nos podemos encontrar:

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

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

Por otro lado, las funciones relacionadas con la planificación de procesos obedecen a los siguientes objetivos:

1. Equidad: Este objetivo consiste en compartir la CPU equitativamente, sin privilegiar notoriamente algún tipo de proceso.
2. Maximizar la utilización de la CPU: Las funciones que realice el planificador de procesos tienden a mantener la CPU utilizada la mayor parte del tiempo.
3. Maximizar la productividad: La productividad es una medida del rendimiento, que se refleja con la cantidad de tareas que puede realizar la CPU en un intervalo de tiempo.
4. Minimizar el tiempo de espera: Este tiempo corresponde al tiempo en que un proceso está en la cola de procesos listos, es un tiempo de espera por asignación de CPU.

5. Minimizar el tiempo de retorno: Este tiempo corresponde al tiempo total que se utiliza para la ejecución completa del proceso.
6. Minimizar el tiempo de respuesta: Este tiempo está relacionado con los tiempos de respuesta parciales de los procesos interactivos. Puesto que estos procesos se caracterizan porque interactúan con el medio constantemente durante la ejecución completa del proceso.

Criterios de Planificación:

Los algoritmos de planificación tendrán distintas propiedades y favorecerán cierta clase de procesos.

- Es necesario definir criterios para poder evaluar los algoritmos de planificación:
  - Utilización de CPU (CPU utilization): Es el porcentaje de uso (en cuanto a ejecución de tareas de usuario o del sistema que son consideradas útiles) que tiene un procesador.
  - Rendimiento (Throughput): Es el número de procesos que ejecutaron completamente por unidad de tiempo (una hora p.ej.).
  - Tiempo de retorno (Turnaround time): Es el intervalo de tiempo desde que un proceso es cargado hasta que este finaliza su ejecución.
  - Tiempo de espera (Waiting time): Es la suma de los intervalos de tiempo que un proceso estuvo en la cola de procesos listos (ready queue).
  - Tiempo de respuesta (Response time): Es el intervalo de tiempo desde que un proceso es cargado hasta que brinda su primera respuesta. Es útil en sistemas interactivos.

En términos generales, la planificación de procesos de un sistema operativo puede adoptar uno de dos conceptos de planificación. Estos conceptos de tipo de planificación se presentan a continuación:

La **planificación no apropiativa** (en inglés, no preemptive) es aquélla en la que, cuando a un proceso le toca su turno de ejecución, ya no puede ser suspendido; es decir, no se le puede arrebatar el uso de la CPU, hasta que el proceso no lo determina no se podrá ejecutar otro proceso. Este esquema tiene sus problemas, puesto que, si el proceso contiene ciclos infinitos, el resto de los procesos pueden quedar aplazados indefinidamente. Otro caso puede ser el de los procesos largos que penalizarían a los cortos si entran en primer lugar.

La **planificación apropiativa** (en inglés, preemptive) supone que el sistema operativo puede arrebatar el uso de la CPU a un proceso que esté ejecutándose. En la planificación apropiativa existe un reloj que lanza interrupciones periódicas en las cuales el planificador toma el control y se decide si el mismo proceso seguirá ejecutándose o se le da su turno a otro proceso.

En ambos enfoques de planificación se pueden establecer distintos algoritmos de planificación de ejecución de procesos. Algunos de los algoritmos para decidir el orden de ejecución de los procesos en el sistema son:

- ✓ Round Robin (**apropiativo**)
- ✓ Por prioridad (**apropiativo**)
- ✓ El tiempo restante más corto (**apropiativo**)
- ✓ El trabajo más corto primero (**no apropiativo**)
- ✓ El primero en llegar, primero en salir -FIFO- (**no apropiativo**)

La planificación **no apropiativa** es aquélla en la que, cuando a un proceso le toca su turno de ejecución, ya no puede ser suspendido; es decir, no se le puede arrebatar el uso de la CPU, hasta que el proceso no lo determina no se podrá ejecutar otro proceso.

La planificación **apropiativa** supone que el sistema operativo puede arrebatar el uso de la CPU a un proceso que esté ejecutándose.

Cada uno de los algoritmos indicados utilizará un criterio diferente para establecer el orden de asignación de la CPU a los procesos (de forma iterativa, por prioridad, por tiempo restante de ejecución, por tiempo de llegada, etc).

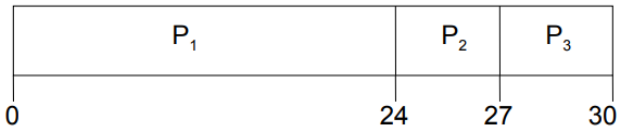
FCFS o FIFO (First Come, First Served o First input, first output):

Planificación de servicio por orden de llegada

- Los procesos son ejecutados en el orden que llegan a la cola de procesos listos.
- La implementación es fácil a través de una cola FIFO.
- Es adecuado para sistemas por lotes (batch).
- Es un algoritmo no expropiativo: una vez que el procesador le es asignado a un proceso este lo mantiene hasta que termina o se bloquea (por ejemplo, al generar un pedido de E/S).
- El tiempo de espera promedio por lo general es alto.

Proceso	Burst Time
P1	24
P2	3
P3	3

Ráfaga de CPU



- Tiempo de espera: P1 = 0; P2 = 24; P3 = 27
- Tiempo de espera promedio: (0 + 24 + 27)/3 = 17

Shortest Job First (SJF - Trabajo más corto primero)

El planificador elige aquellos procesos que requieran menor cantidad de tiempo para ejecutarse, y luego asigna CPU de acuerdo a ese orden. En el caso en que dos o más procesos demoren el mismo tiempo, se usa el criterio de FIFO.

El algoritmo asocia a los procesos el largo de su próximo CPU-burst(RAFAGA DE CPU).

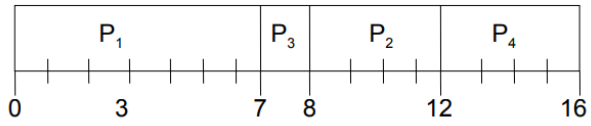
- Cuando el procesador queda disponible se le asigna al proceso que tenga el menor CPU-burst.
- Si dos procesos tienen el mismo CPU-burst se desempata de alguna forma.
- Su funcionamiento depende de conocer los tiempos de ejecución lo cual en la mayoría de los casos no sucede.
- Es adecuado para sistemas por lotes (batch).
- Dos esquemas:
  - No apropiativo: una vez que se le asigna el procesador a un proceso no se le podrá quitar.
  - Apropiativo: Si un nuevo proceso aparece en la lista de procesos listos con menor CPU-burst, se le quita la CPU para asignarla al nuevo proceso.
- Este algoritmo es óptimo para el tiempo de espera, pero requiere que todos los procesos participantes estén al comienzo (si no es apropiativo) y además hay que saber el tiempo del próximo CPU-burst.

- Es usado para planificación de largo plazo más que para planificación de corto plazo.

Shortest Job First (SJF) – No apropiativo

Proceso	Tiempo de arribo	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

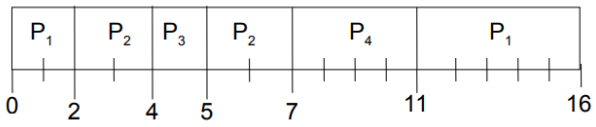
- Tiempo de espera promedio: (0 + 6 + 3 + 7)/4 = 4



Shortest Job First (SJF) – Apropiativo

Proceso	Tiempo de arribo	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

- Tiempo de espera promedio: (9 + 1 + 0 + 2)/4 = 3

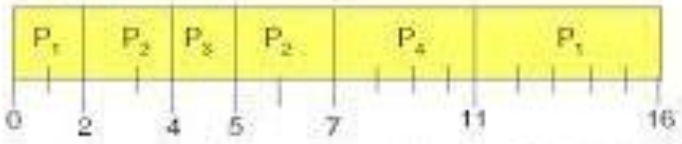


Shortest Remaining Time First (SRTF) - Apropiativo

- Es una versión apropiativa de SJF.
- Cada vez que entran trabajos se interrumpe el actual y se compara el tiempo restante de éste con el de los entrantes.
- Si hay un trabajo nuevo más corto que lo que le falta al actual en CPU, echamos el actual y metemos el nuevo.
- De nuevo, se supone que se conocen los tiempos de uso futuro de CPU de antemano. Una versión práctica debe hacer uso de una estimación.

Procesos	Llegada	Tiempo CPU (ms)
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SRTF (*expulsivo*)



- Tiempo de espera medio = (9 + 1 + 0 +2)/4 = 3

- Expulsivo = Apropiativo

Basados en Prioridad

- A cada proceso se le asigna un número entero que representa su prioridad.
- El planificador asigna el procesador al proceso con la más alta prioridad.
- Se utiliza en general un esquema apropiativo ya que si un proceso con mayor prioridad que el que esta ejecutando arriba a la lista de procesos listos (ready queue), será asignado al procesador.
- SJF se puede ver como un algoritmo de prioridad donde la prioridad está dada por el próximo CPU-burst.
- Es adecuado para sistemas interactivos.
- Sufre de posposición indefinida ya que un proceso de baja prioridad quizás no pueda ejecutar nunca.
- La solución es utilizar prioridades dinámicas de envejecimiento: incrementar la prioridad según pasa el tiempo sin ejecutar.
- La prioridad de un proceso para el uso del recurso procesador deberá ser inversamente proporcional al uso que el proceso haga del mismo.
- Por lo tanto un proceso tipo I/O-bound deberá tener, en general, mayor prioridad que uno tipo CPU-bound

Round Robin (RR)

EJEMPLO: PLANIFICACIÓN CON PRIORIDAD

Proceso	Ráfaga	Prioridad
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

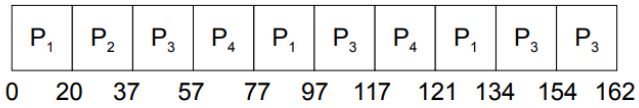
- La carta de Gantt

- Tiempo de espera promedio= 8.2 msec

- Cada proceso toma una pequeña unidad de tiempo de CPU (quantum). Luego de este tiempo el proceso es quitado de la CPU y agregado a la cola de listos.
- Si hay n procesos en la cola de listos y el tiempo del quantum es q, entonces cada proceso toma 1/n del tiempo de CPU en rebanadas de a lo sumo q unidades de tiempo a la vez. Los procesos no esperan mas que (n-1)q unidades de tiempo.
- Rendimiento
- q largo/grande => Primero-Entrar, Primero-Salir(FIFO)
- q chico => q debe ser grande con respecto al cambio de contexto, sino la sobrecarga es demasiado grande.
- Con un Quantum PEQUEÑO se incrementan los Cambios de Contexto
- Es ideal para sistemas de tiempo compartido.

Proceso	Burst Time
P1	53
P2	17
P3	68
P4	24

quantum = 20



- Por lo general, tiene un mayor tiempo de retorno que el *SJF*, pero mejora el tiempo de respuesta.