TABLA DE CONTENIDOS

INTRODUCCIÓN....3

ADMINISTRACIÓN DE PROCESOS....3

Conceptos fundamentales....3

Proceso....3
Estados de un Proceso....5
Tabla de Procesos (PCB)....6
Cambio de Contexto....7
Jerarquía de Procesos....8

PLANIFICACIÓN DE PROCESOS....10

ALGORITMOS DE PLANIFICACIÓN....13

Algoritmos de Planificación No Apropiativa....13

Algoritmo FIFO o FCFS (First-Come First-Served)....13
Algoritmo SJF (Short Job First)....15

Algoritmos de Planificación Apropiativa....16

Algoritmo RR (Round Robin)....17
Algoritmo por Prioridades....18
Ejercicio....20
Algoritmo de colas de Prioridades múltiples....20
Algoritmo de colas de múltiples niveles con realimentación....21
Planificación de dos niveles....21

EJERCICIOS PLANTEADOS....23

EJERCICIO RESUELTOS....25

DIRECCIONES DE INTERNET....29

EVALUACIÓN DEL MÓDULO 2....1

INTRODUCCIÓN

En este segundo módulo, nos concentraremos en realizar un análisis detallado de la forma y métodos que utiliza el Sistema Operativo para administrar el procesador.

Como ustedes recordarán es una de las áreas mas importantes que debe atender, conjuntamente con la administración de memoria, dispositivos de entrada/salida y administración de archivos; aspectos que serán tratados en forma detallada e individualmente en los sucesivos módulos.

ADMINISTRACIÓN DE PROCESOS

Los conceptos de *Proceso*, *Procesamiento concurrente* y *Procesamiento paralelo* son fundamentales para el entendimiento y comprensión de los sistemas operativos modernos.

Un proceso es una abstracción de un programa en ejecución y es la unidad de trabajo del sistema. El sistema, a su vez, está formado por un conjunto de procesos que se ejecutan concurrentemente, y básicamente el sistema maneja tanto procesos del sistema operativo (aquellos que ejecutan código del sistema), como procesos de los usuarios (aquellos que ejecutan el código de los usuarios). En este módulo hablaremos del concepto de proceso como modelo e identificaremos los *estados* posibles en que éste se puede encontrar.

Además, este módulo contempla la exposición de varios métodos que se utilizan para la administración de procesos, planificación de la CPU, la comunicación y sincronización de procesos.

Conceptos fundamentales

El primer concepto que analizaremos en esta sección corresponde a *proceso*, que es la base para entender muchos otros conceptos que se manejan en el sistema operativo.

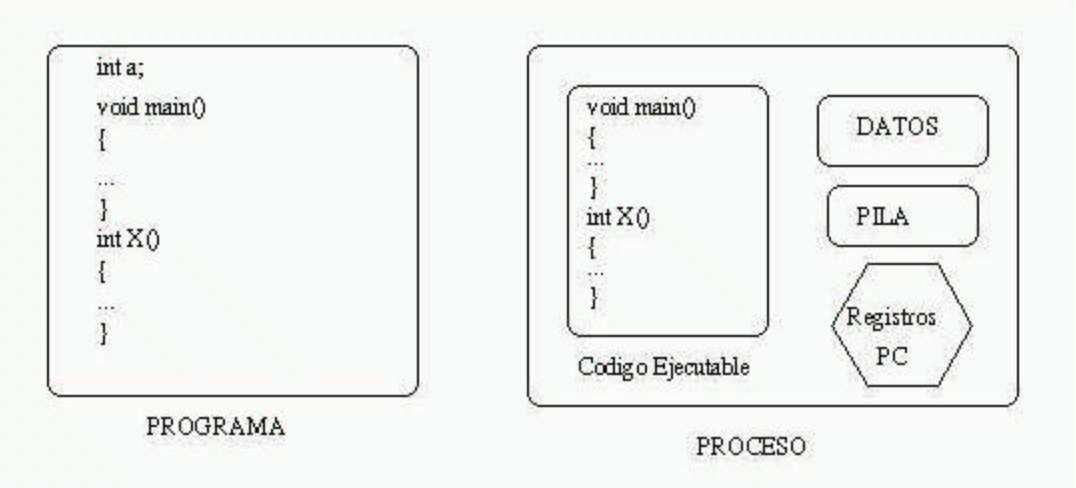
Proceso

Como ya se mencionó en los párrafos anteriores, 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.

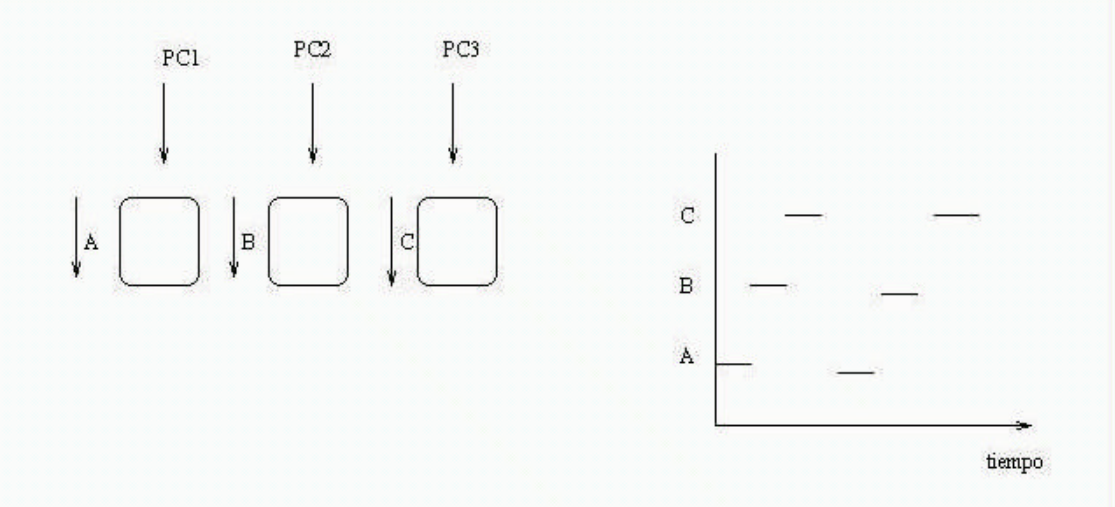
Hablando de los recursos que utilizan ambos conceptos, podemos decir, que el programa utiliza únicamente memoria secundaria, en cambio el proceso utiliza memoria principal y procesador.

En la siguiente figura se aprecian ambos conceptos.



En los sistemas multiprogramados, de tiempo compartido que operan sobre un computador con un procesador, se produce el fenómeno denominado *Procesamiento concurrente*. Este fenómeno, consiste en que la CPU alterna la ejecución de los procesos en porciones fijas de tiempo, también conocido como *Seudoparalelismo*, en el sentido que, ante los ojos de los usuarios dueños de los procesos, la ejecución de sus procesos es paralela; esta ilusión es producto de que el tiempo fijo que asigna el sistema a cada uno de los procesos es muy pequeño, y por lo tanto difícil de ser percibido por el hombre. Cuando el sistema computacional está provisto de varios procesadores, entonces él puede realizar lo que se denomina *Procesamiento paralelo*, esto implica que los procesos pueden ser ejecutados efectivamente en distintos procesadores en forma paralela.

En la siguiente figura se refleja el fenómeno de la concurrencia:

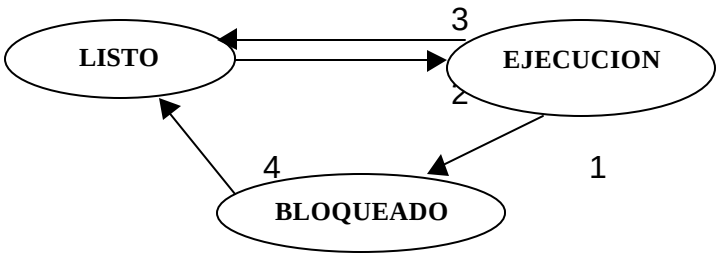


La alternancia de los procesos en el sistema es regulada por el administrador de procesos y obedece a un algoritmo de planificación de la CPU.

Estados de un Proceso

Dado que en un sistema de tiempo compartido conviven un conjunto de procesos en el sistema, en donde se incluyen procesos del sistema y procesos de los usuarios, esta concurrencia produce que los procesos se puedan encontrar en diferentes estados, ya que por ejemplo, en un momento dado solo un proceso puede estar en ejecución por el hecho de existir un solo procesador.

A continuación se presenta una figura que refleja la relación entre los distintos estados posibles que puede presentar un proceso.



Los tres estados básicos son:

1. **En ejecución**: El Proceso esta utilizando la CPU, las instrucciones se están ejecutando.

2. **Listo**: El Proceso está en condiciones de ejecutarse, pero esta esperando que se le asigne tiempo de CPU.

3. **Bloqueado**: El Proceso está esperando que ocurra un evento[1] externo (como la terminación de una E/S).

[1] Acontecimiento que tiene que ocurrir para continuar la ejecución de instrucciones

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.

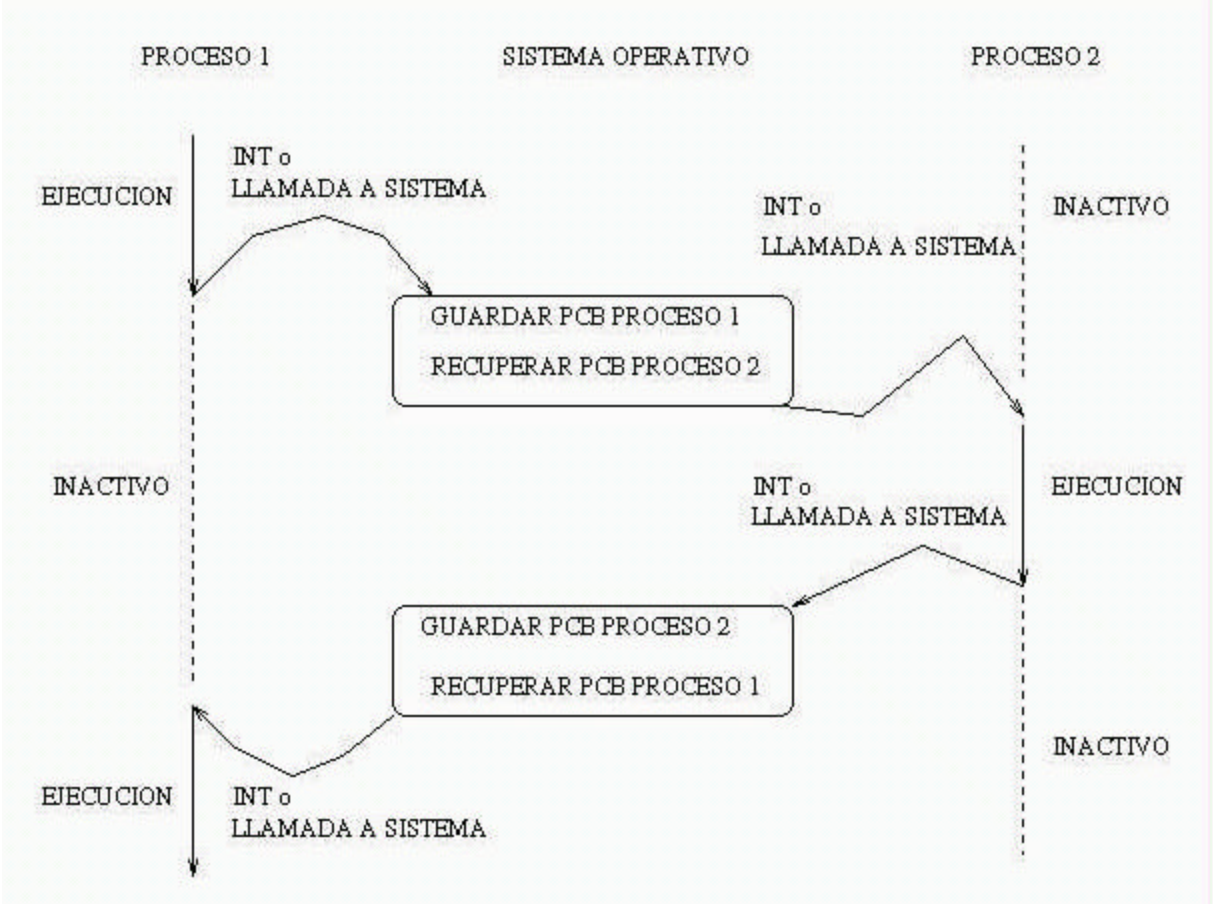
- **Información del estado de E/S**[2]: Esta información incluye, solicitudes pendientes de E/S, dispositivos de E/S asignados al proceso,etc.

Cambio de Contexto

Otro concepto muy importante que incluye el concepto de proceso, es el denominado *cambio de contexto*. Este concepto esta 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.

[2] Impresora, teclado, sistema de archivo, pantalla, etc.



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***[3] 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

[3] Los programas de usuarios solicitan sevicios al sistema operativo mediante llamadas al sistema. Por ej. Hay llamadas para crear procesos, controlar la memoria, leer y escribir en dispositivos de Entrada/Salida, etc.

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 DE PROCESOS

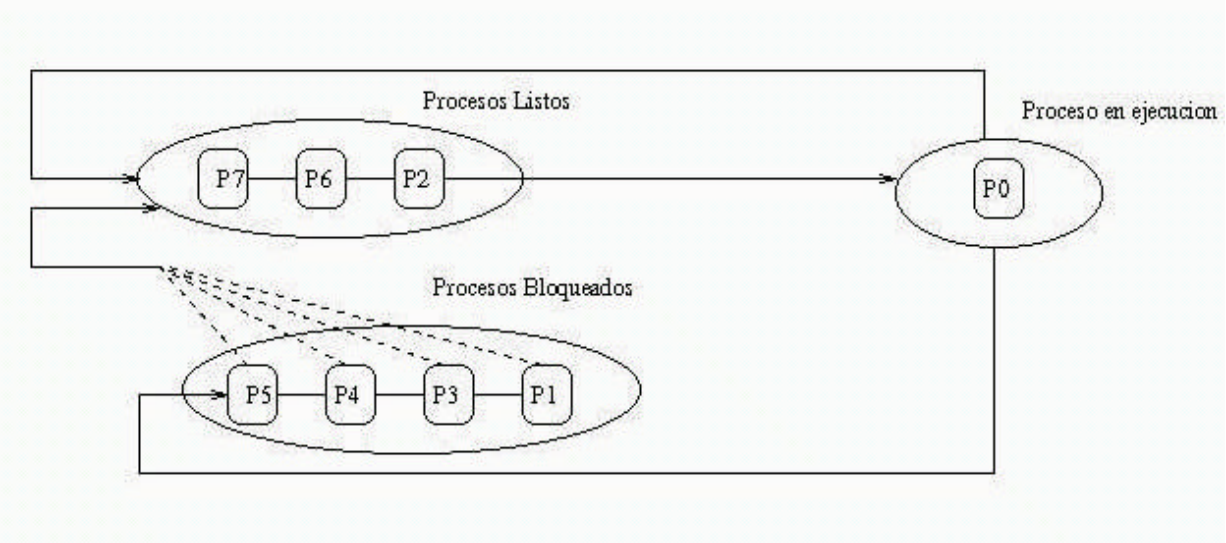
En un sistema multiprogramado, de tiempo compartido, existe la posibilidad de tener muchos procesos ejecutándose en el sistema concurrentemente. La concurrencia de procesos se logra en base a la compartición de recursos. Dado que un proceso incluye recursos, tales como memoria, para almacenar datos, pila y código ejecutable, también involucra el recurso para el procesamiento que está reflejado en el procesador y los registros que el maneja. Esta compartición es regulada y controlada por el sistema operativo.

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 esta 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 esta 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 esta 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.

A continuación se presenta un diagrama que muestra como funciona esto.



En la figura 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

Las funciones de la planificación de procesos son realizadas mediante un módulo principal del sistema operativo denominado "*Planificador de Corto Plazo"*, el que toma en cuenta determinados criterios para tomar la decisión de que proceso de la cola de listos es el que obtendrá CPU.

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 interactuan con el medio constantemente durante la ejecución completa del proceso.

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:

1. **Planificación No apropiativa**: Este tipo consiste en que el planificador de procesos *puede quitar la CPU a un proceso, sólo en dos casos: cuando un proceso cambia de estado de ejecución a bloqueado, y cuando un proceso termina.* Si observamos el diagrama de estados de un proceso, ***nunca*** se dará la transición 2 puesto que el SO no se puede apropiar de la CPU.

2. **Planificación Apropiativa**: Este tipo consiste en que el planificador tiene las facultades de quitar la CPU a un proceso que está en ejecución. Con este tipo de planificación es totalmente factible que se cumpla la transición 2 del diagrama de estados.

Cada uno de estos tipos de planificación tienen asociados un conjunto de algoritmos, de los cuales analizaremos los mas representativos de cada grupo.

ALGORITMOS DE PLANIFICACIÓN

Algoritmos de Planificación No Apropiativa

Como ya se mencionó en la sección anterior, este tipo de planificación no posee las facultades de quitar la CPU al proceso que la está utilizando, sino que sólo lo puede hacer cuando el proceso cambia de ejecución a bloqueado o cuando el proceso termina. Este tipo de característica no es atractiva para los sistemas multiusuarios, en donde, muchos de los procesos que ellos ejecutan son de naturaleza interactiva, es decir que poseen mucha comunicación con el usuario utilizando por lo general tiempos cortos de procesamiento intercalado con E/S.

Por otro lado, en este tipo de planificación, puede darse repetidamente que si un proceso utiliza mucho tiempo de ejecución, sin cambios de estado, inhabilita la ejecución del resto de los procesos del sistema.

Sin embargo, este tipo de planificación es sencilla en cuanto a su implementación, puesto que no es necesario ocupar el manejo de interrupciones que le permitan al sistema interrumpir la ejecución de los procesos.

Existen diversos algoritmos de planificación no apropiativa, cada uno de ellos tiene propiedades diferentes que pueden favorecer a un tipo de proceso, muchas veces en desmedro de otros.

Algoritmo FIFO o FCFS (First-Come First-Served)

Este algoritmo consiste en que se otorga la CPU al primer proceso que la requiere. Su implementación es sencilla, pues basta tener una cola FIFO. Cuando un proceso entra a la cola de procesos listos, su PCB se agrega al final de la cola, y cuando la CPU queda libre es asignada al primer proceso de la cola de listos.

El tiempo de espera promedio en este tipo de algoritmo es muy variable, pues el tiempo de espera de cada uno de los procesos que están en la cola de listos depende del orden de dichos procesos en la cola. Consideremos el siguiente ejemplo:

Proceso	Tiempo ejecución
P1	15
P2	5
P3	7

Según la tabla presentada, el tiempo de espera por proceso es el siguiente:

Tpo espera P1 = 0
Tpo espera P2 = 15
Tpo espera P3 = 20
Tpo espera Prom = 11,66

Es decir, el primer proceso no espera nada ya que inmediatamente es atendido, cuando termina este se atiende al siguiente, por eso el segundo espera lo que demoró el primero en ejecutarse y así sucesivamente hasta el final.

Si quisiéramos representar los tiempos de ejecución en forma completa desarrollaríamos un esquema como el siguiente:

											0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	**5**	**4**	**3**	**2**	**1**	**0**																					
2																											
3																											

El gráfico lo interpretamos de la siguiente manera: Las columnas representan los tiempos del procesador, desde el instante 0 hasta el último que nos interesa. Las filas representan los procesos con los que estamos trabajando, uno por cada fila. En la intersección de las filas con las columnas se coloca la cantidad de tiempo que cada proceso necesita en el instante indicado. Así para el instante 0 están los tres procesos con todo el tiempo requerido. De letra resaltada y con fondo más oscuro marcamos cuál de los procesos presentes en un determinado instante es el que tiene el procesador (el que está en ejecución). Esto va a variar de un planificador a otro, pero lo que siempre será igual es que aquel que tenga la ejecución en una determinada columna, para la columna siguiente le faltará un instante menos para terminar su ejecución debido a que de todo el tiempo que solicitó ya se le dio un instante.

Los instantes donde el proceso está presente, es decir figura en la columna, pero no tiene la ejecución, son aquellos considerados como espera.

Así, en el instante 16 el proceso 1 ya está terminado, el proceso 2 está en ejecución y el 3 está listo para ejecutarse pero espera el turno.

Si cambiamos el orden de llegada de los procesos, produciéndose la siguiente secuencia de llegada P2, P3, P1 el tiempo de espera promedio, de la siguiente manera:

Tpo espera P1 = 12
Tpo espera P2 = 0
Tpo espera P3 = 5
Tpo espera Prom = 5,66

Luego, se puede concluir que el tiempo de espera promedio no es mínimo y puede variar mucho debido, por una parte a las diferencias por requerimiento de tiempo de CPU y por el orden de llegada de los procesos a la cola de procesos listos.

La utilización del algoritmo FCFS no es apta para los sistemas interactivos, puesto que estos requieren que se le otorgue un pequeño tiempo a cada proceso en forma regular.

Algoritmo SJF (Short Job First)

Este algoritmo consiste en que 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 FCFS.

Este algoritmo tiene la ventaja que se minimiza el tiempo de espera promedio, pero tiene el grave inconveniente de que no es posible predecir cual será el siguiente tiempo requerido por algún proceso. Sin embargo, existen algunos mecanismos matemáticos que hacen posible una aproximación considerando datos históricos de la ejecución.

Este algoritmo puede ser del tipo No apropiativo o Apropiativo. Esto significa que si es No apropiativo, la decisión del orden de ejecución en un momento es fija, no varía de acuerdo a los nuevos requerimientos; en cambio, en una planificación apropiativa el orden de ejecución de los procesos puede variar dependiendo de los siguientes requerimientos o de los procesos que se necesiten ejecutar en el futuro.

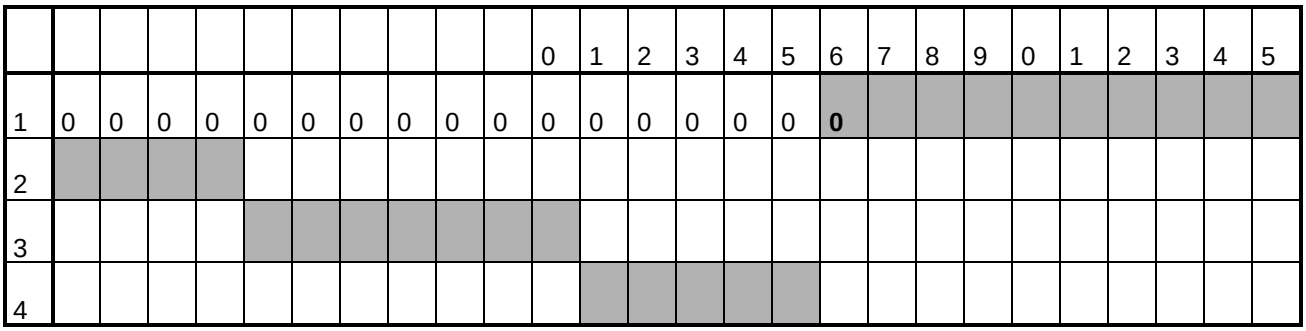
A continuación se presenta un ejemplo:

Proceso	Momento de llegada	Tiempo ejecución
P1	0	10
P2	0	4
P3	1	7
P4	5	5

Caso: No apropiativo:

Tiempo espera P1	=	16
Tiempo espera P2	=	0
Tiempo espera P3	=	3
Tiempo espera P4	=	6
Tiempo espera prom	=	6,25

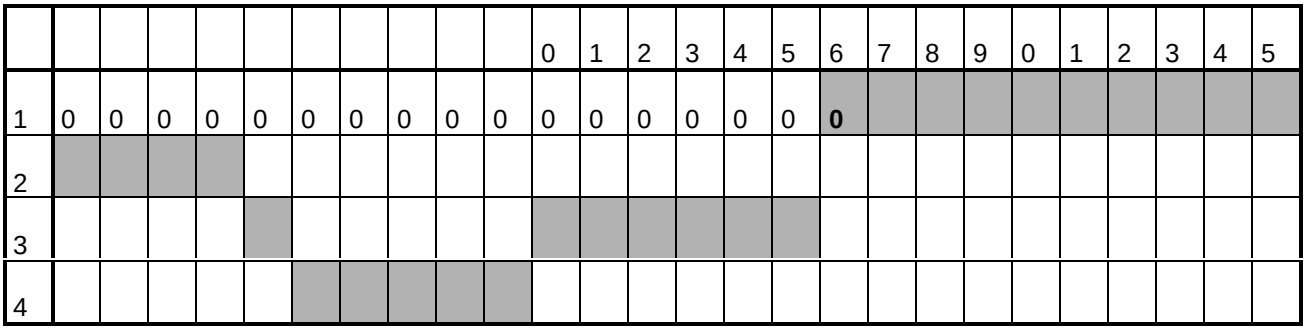
En este caso el proceso 2 es el que menos tiempo requiere para trabajar de los que están en la cola en el momento inicial, el momento 0, entonces es el primero que trabaja durante todo el tiempo requerido, es decir 4 instantes.



Hay que observar en este caso que en el instante 0 solo están presentes los procesos 1 y 2 y se decide sobre estos dos quién es el que va a ser ejecutado. Una vez que termina el proceso 2 que fue el elegido inicialmente ya apareció el proceso 3 y entonces también entra en la decisión para saber cual será el siguiente y así hasta el final.

Caso: Apropiativo:

Tiempo espera P1	=	16
Tiempo espera P2	=	0
Tiempo espera P3	=	8
Tiempo espera P4	=	0
Tpo espera promedio	=	6



En este caso se observa la apropiación en el instante 5 cuando llega el proceso 4. En este momento el proceso nuevo, que acaba de llegar requiere menos tiempo que el que se estaba ejecutando (el proceso 3), entonces el planificador decide que sea el que trabajo y lo suspende momentáneamente al proceso 3 hasta que termine el 4

Algoritmos de Planificación Apropiativa

Como ya se mencionó anteriormente la planificación apropiativa implica que el sistema tiene las facultades o suficiente "autoridad" como para quitar el

procesador al proceso que lo está ejecutando. Este tipo de planificación es el más adecuado para sistemas multiusuarios e interactivos, pues en este caso el sistema puede hacer que los procesos de los usuarios se ejecuten de tal manera que el usuario tenga respuestas rápidas a sus requerimientos. Sin embargo, la implementación de este tipo de planificación, comparativamente es mas compleja que la no apropiativa, pues debe proporcionar los mecanismos mediante el cual el sistema pueda efectuar la interrupción de procesos y la conmutación de contexto en los momentos que defina el algoritmo particular.

A continuación se presentan algunos de los algoritmos apropiativos mas conocidos:

Algoritmo RR (Round Robin)

Este algoritmo fue diseñado para sistemas de tiempo compartido y está basado en la asignación de un tiempo de procesador a cada proceso del sistema; a este tiempo se le denomina **Quantum**.

El algoritmo consiste en tener una cola tipo FIFO de procesos listos, entonces, el proceso que llega a la cola se ubica al final de ella. El planificador de procesos toma el primer proceso de la cola, activa un cronómetro para que interrumpa después de un quantum de tiempo y asigna la CPU al proceso.

Posteriormente puede suceder una de dos alternativas.

- Si el proceso ocupa en ejecución menos tiempo que el quantum, entonces el proceso libera voluntariamente la CPU y el planificador continua con la planificación del siguiente proceso listo.

- Si el proceso requiere para su ejecución mas tiempo que el otorgado a través de un quantum, se produce la interrupción del cronómetro cuando su tiempo expira, en ese momento el planificador detiene la ejecución del proceso, lo pone al final de la cola de procesos listos, y realiza la conmutación de contexto para la ejecución, luego asigna a el siguiente proceso de la cola de listos un quantum y asigna el procesador a ese proceso. De esta forma el ciclo se repite hasta el final.

El rendimiento del algoritmo depende mucho del tamaño del quantum. Si es muy grande este algoritmo se traduce al algoritmo FCFS y los tiempos de respuesta se ven desmejorados, y si es muy pequeño la razón entre el tiempo de utilización efectiva de CPU y cambio de contexto es muy baja. El compromiso que se debe tomar, en este algoritmo es un quantum, que sea grande respecto al tiempo de contexto, pero no demasiado para no caer en el algoritmo FCFS. Lo razonable es que el quantum se fije de manera tal, que el cambio de contexto esté cercano al 5% del quantum. Habitualmente el cambio de contexto toma aproximadamente entre 1 y 5 [mseg[4]], por lo que un tiempo razonable para el

[4] Unidad de tiempo, que significa una milésima de segundo.

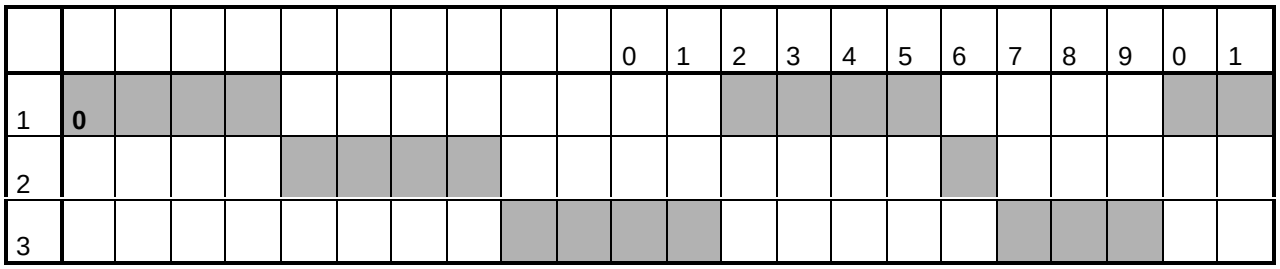
quantum está entre 50 y 100 [mseg]. Este tiempo alcanza para realizar 5.000.000 de instrucciones en un Pentium de 100 MHz[5].

A continuación se presenta un ejemplo de como opera este algoritmo. Considere un quantum = 4 [mseg] y el tiempo de cambio de contexto considérelo despreciable.

Proceso	Tiempo ejecución
P1	10
P2	5
P3	7

Según la tabla presentada, el tiempo de espera por proceso es el siguiente:

Tpo espera P1 = 12
Tpo espera P2 = 12
Tpo espera P3 = 13
Tpo espera Prom = 12,3



Como se ve cada proceso recibe como máximo 4 instantes que es el valor definido como quantum, si es que le falta, tiene que esperar hasta que todos reciban esa cantidad para que le toque nuevamente. Si es que necesitaba menos, como ocurre con el proceso 2 en el instante 16, ocupa lo que le falta y luego libera el procesador en forma voluntaria.

Ejercicio:

Haga el mismo ejercicio considerando un tiempo en cambio de contexto de 1 [mseg].

Algoritmo por Prioridades

Este algoritmo consiste en asociarle a cada proceso una prioridad, de modo tal que los procesos que poseen mayor prioridad se ejecutarán primero. Si dos o más procesos poseen la misma prioridad entonces se aplica política FCFS.

[5] Unidad que mide la velocidad de procesamiento de instrucciones de la CPU

No se ha llegado a un acuerdo en la representación con números grandes o pequeños. Aquí consideraremos los números pequeños como aquellos de mayor prioridad (el proceso que tenga asociado un 1 tendrá mayor prioridad que el que tenga asociado un 2, y así sucesivamente).

El algoritmo de planificación por prioridades puede ser apropiativo o No apropiativo. En el caso de un algoritmo apropiativo significa lo siguiente: Si un proceso, que llega a la cola de procesos listos, posee mayor prioridad que el proceso que se esta ejecutando, entonces se le quitará la CPU a ese proceso y se le otorgará al proceso que llega.

En cambio en un algoritmo No apropiativo, el procesador no es quitado al proceso que lo está utilizando, sino, lo que solamente ocurre es que el proceso que llega se coloca al inicio de la cola de procesos listos.

Un problema que surge en este tipo de algoritmos es que puede suceder que los procesos de prioridades muy baja no obtengan CPU. Esto se puede dar en sistemas que tienen mucha demanda por procesamiento, en donde muchos procesos son de alta prioridad, esto provocará que continuamente se incorporan en la lista de procesos listos procesos de alta prioridad que van desplazando la posibilidad de ejecución de los procesos de prioridad baja.

Un mecanismo que se utiliza para evitar el aplazamiento indefinido de los procesos de baja prioridad es el envejecimiento o añejamiento, el cual consiste en aumentar gradualmente en el tiempo la prioridad de los procesos que esperan durante mucho tiempo en el sistema.

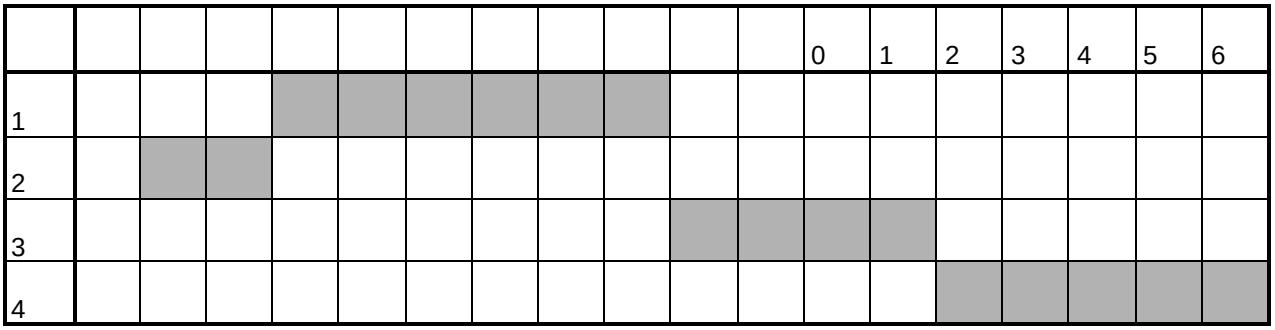
Un ejemplo de como funciona este algoritmo, es el siguiente:

Proceso	Tiempo ejecución	Instante de llegada	Prioridad
P1	6	0	3
P2	2	0	2
P3	4	4	1
P4	5	7	2

Caso No apropiativo:

Tiempo espera P1 = 2
Tiempo espera P2 = 0
Tiempo espera P3 = 4
Tiempo espera P4 = 5

Tiempo espera prom = 2.75



Obsérvese que en este gráfico agregamos una columna más a la izquierda con el valor de la prioridad de cada proceso, para que a la hora de tener que elegir cual de los procesos presentes tiene la mejor prioridad sea más fácil.

Ejercicio

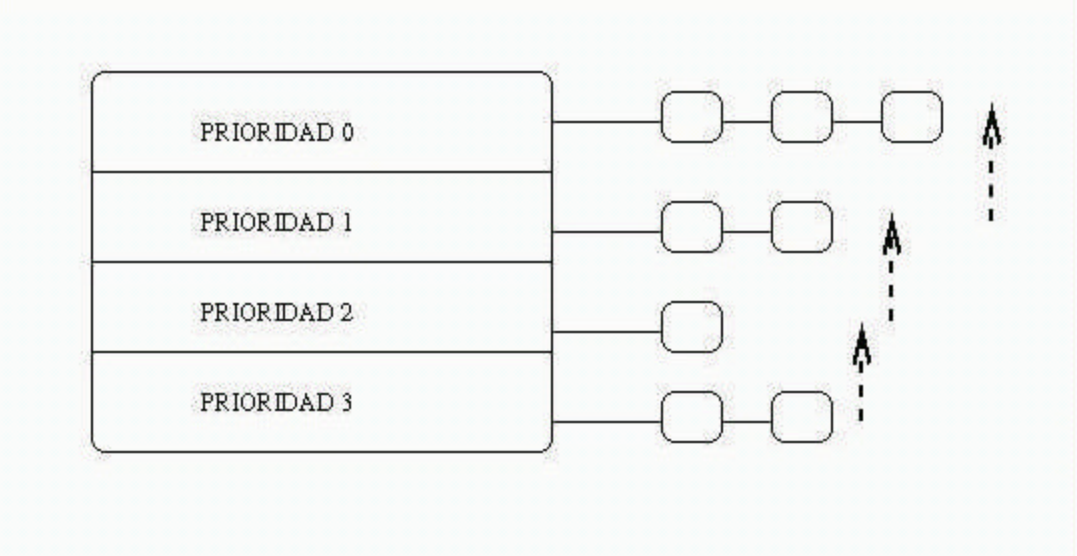
Como serían los tiempos de espera para el caso Apropiativo?...

Algoritmo de colas de Prioridades múltiples

Este algoritmo surge para tratar de dar un mejor servicio a los usuarios interactivos, y está basado en una mezcla entre algoritmos ya vistos y el algoritmo por prioridades; lo usual es que se mezcle el algoritmo RR y el por prioridades. Este algoritmo contempla distintas colas de procesos en donde a cada una de ellas se le asigna una determinada prioridad; y se utiliza el algoritmo RR para la planificación de procesos pertenecientes a una misma cola.

La operación del algoritmo es como sigue: la CPU es asignada a los procesos pertenecientes a las colas de mayor prioridad, cuando la cola de prioridad alta queda vacía, se pasa a ejecutar los procesos asociados a la cola de siguiente prioridad y así, sucesivamente.

.



Algoritmo de colas de múltiples niveles con realimentación

Este algoritmo básicamente es igual al anterior, solo que los procesos pueden moverse de una cola de menor a una de mayor prioridad y viceversa. La idea en este algoritmo es diferenciar los tipos de procesos de acuerdo a sus requerimientos en tiempos de ejecución; si un proceso utiliza demasiado tiempo de CPU, se pasa a una cola de menor prioridad. De la misma manera, si un proceso espera demasiado tiempo en una cola de menor prioridad se puede mover a una cola de mayor prioridad, esta es una manera de evitar el bloqueo indefinido, producto del envejecimiento.

Este esquema permite dejar los procesos interactivos en las colas de mayor prioridad. Los procesos interactivos requieren tiempos de respuesta pequeños y necesitan ser atendidos con mayor frecuencia, los procesos por lotes en cambio no requieren interactuar con el sistema y básicamente lo que les interesa es ejecutar su tarea completamente en un tiempo de retorno razonable.

De esta manera, lo que sucede en este algoritmo es que cada una de las colas posee una prioridad asociada y la planificación de los procesos que le pertenecen se realiza mediante RR. Sin embargo, el quantum no es el mismo en ambas colas, siendo más pequeño para los procesos de mayor prioridad que corresponde a los procesos interactivos, y mayores para los prioridades mas bajas.

Planificación de dos niveles

Los algoritmos de planificación vistos hasta ahora, consideran que la cantidad de memoria disponible es ilimitada. Sin embargo, en los sistemas reales el recurso *Memoria* es limitado y por lo tanto debe ser utilizado de la mejor manera posible.

Como ya sabemos para que un proceso se pueda ejecutar las instrucciones que definen su acción deben estar almacenadas en memoria, por lo que, si no se dispone de suficiente memoria es necesario que algunos de los procesos o

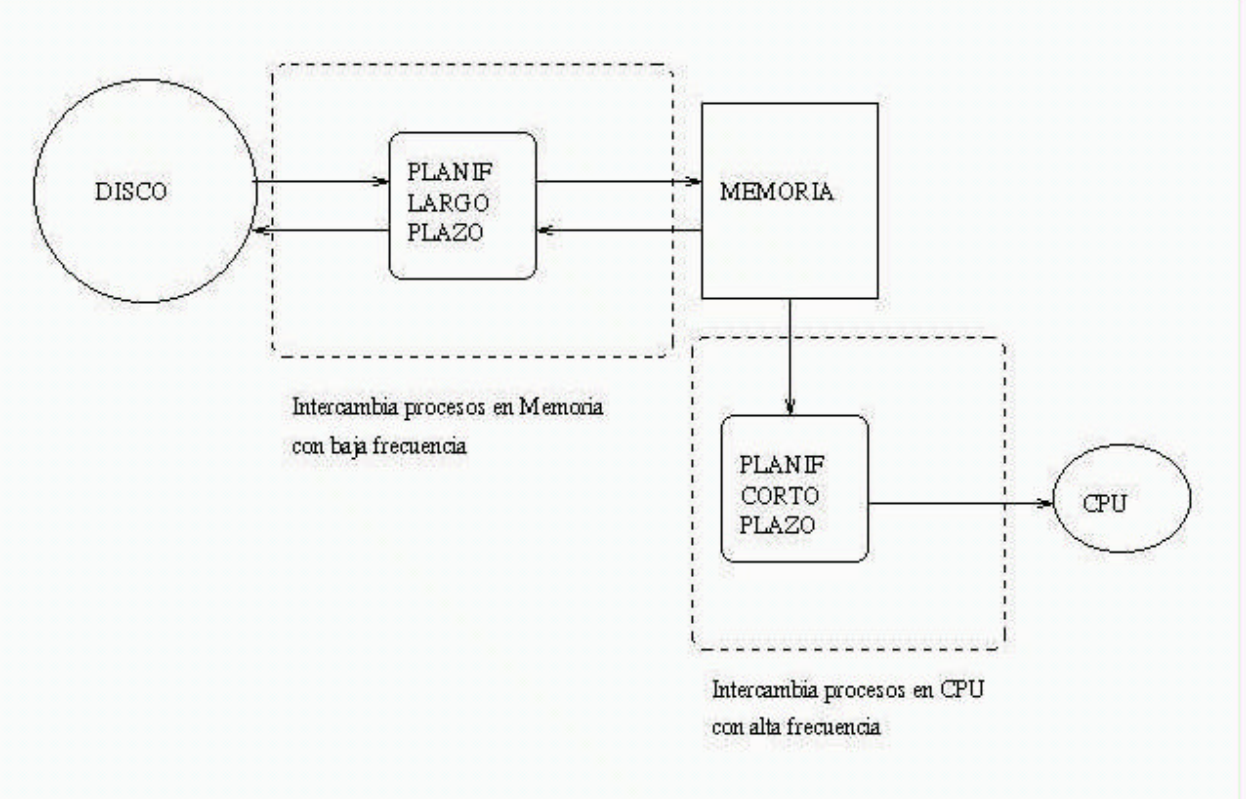
porciones de ellos se encuentren en memoria secundaria. Desde el punto de vista de planificación de procesos ésta es una situación que debe ser considerada, ya que el tiempo involucrado para ejecutar un proceso que está en disco es mucho mayor que el tiempo requerido para ejecutar un proceso que está en memoria, puesto que necesariamente hay que traspasar el proceso de disco a memoria antes de ejecutarlo.

La planificación de dos niveles se realiza mediante, un *planificador de largo plazo* o bien *planificador de nivel alto,* que es el que selecciona los procesos de disco para cargarlos en memoria y viceversa; y el *planificador de corto plazo* o bien *planificador de nivel bajo*, que es el que selecciona un proceso listo y le asigna CPU.

La principal diferencia entre estos dos planificadores es la frecuencia de su ejecución. El planificador de corto plazo debe seleccionar con mucha frecuencia un nuevo proceso para asignación de CPU. Como este planificador debe asegurar una rápida alternancia entre procesos, la selección de cada proceso y la asignación de la CPU propiamente tal debe realizarla muy rápidamente.

Por otro lado, el planificador de largo plazo se ejecuta con mucha menos frecuencia que el de corto plazo. Éste planificador es el encargado de controlar el *grado de multiprogramación* (el número de procesos en memoria). Si el grado de programación se mantiene mas o menos estable, entonces, el planificador debe velar porque la cantidad de procesos que entran a memoria sea mas o menos igual a la cantidad de procesos que salen de memoria.

El algoritmo funciona de la siguiente manera. Primero se cargan en memoria un conjunto de procesos, el planificador de bajo nivel se encarga entonces, por un período de tiempo, de planificar dichos procesos en la CPU. Luego, periódicamente se llama al planificador de alto nivel para que saque de memoria el conjunto de procesos de memoria e intercambie de disco a memoria un nuevo conjunto de procesos. Posteriormente se repite este ciclo. El siguiente esquema refleja el funcionamiento.



Algunos de los parámetros que se analizan para decidir qué procesos se intercambian de disco a memoria y viceversa son:

- Tiempo que el proceso ha permanecido en memoria o disco.
- Tiempo que el proceso ha utilizado CPU.
- Prioridad del proceso
- Tamaño del proceso

EJERCICIOS PLANTEADOS

Calcular el tiempo de espera y graficar el orden de ejecución de cada una de las series de procesos siguientes según el algoritmo requerido.

Algoritmo FCFS

a)

Procesos	Duración
P1	4
P2	6
P3	2
P4	3

b)

Procesos	Duración
P1	1

P2	2
P3	4

P4	10

Algoritmo SJF

a)

Procesos	Llegada	Duración
P1	0	6
P2	0	3
P3	2	5
P4	4	3

b)

Procesos	Llegada	Duración
P1	0	3
P2	0	7
P3	2	2
P4	7	4
P5	8	2

Algoritmo RR

a)

Procesos	Duración
P1	7
P2	3
P3	5
P4	2

Quantum = 4

b)

Procesos	Duración
P1	6
P2	3
P3	4
P4	1

Quantum = 2

Algoritmo de prioridades

a)

Procesos	Duración	Llegada	Prioridad
P1	6	0	2
P2	2	0	1
P3	4	3	4
P4	1	10	2

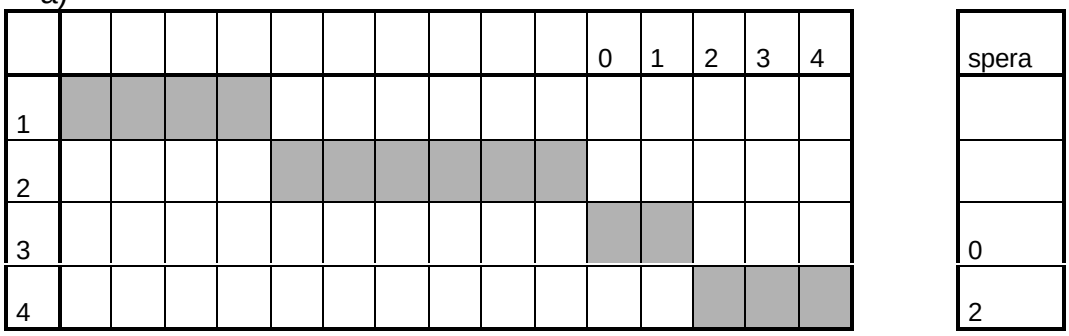
Procesos	Duración	Llegada	Prioridad
P1	3	0	2
P2	7	0	1
P3	2	3	3
P4	2	8	1

b)

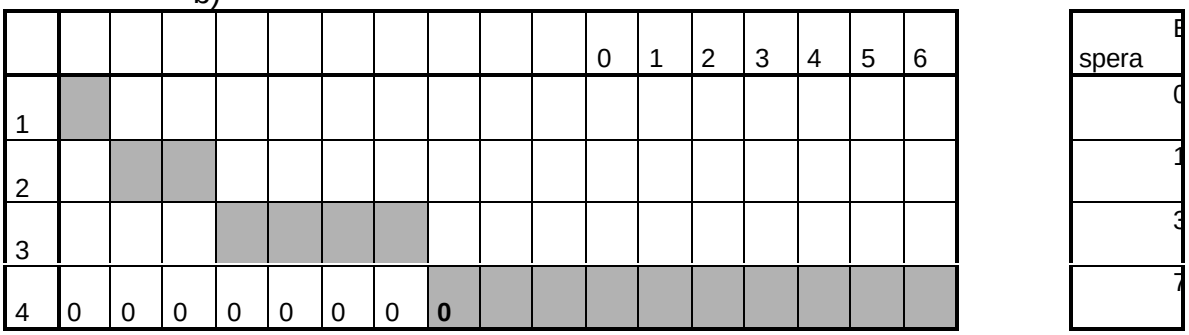
EJERCICIO RESUELTOS

FCFS

a)

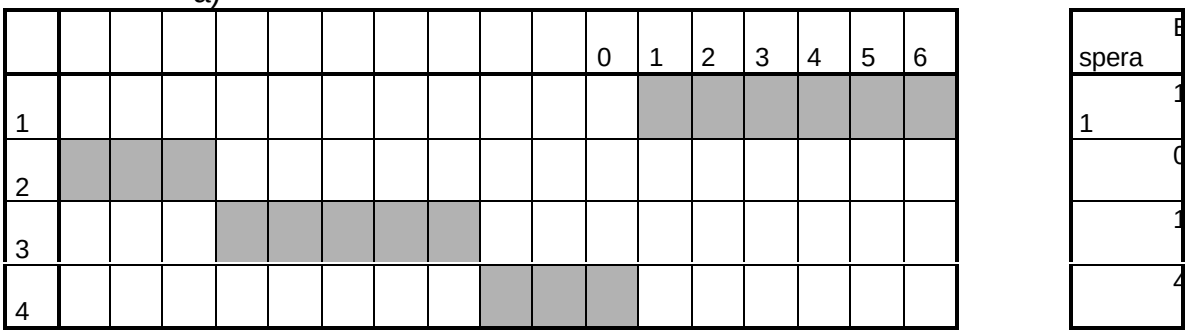


b)

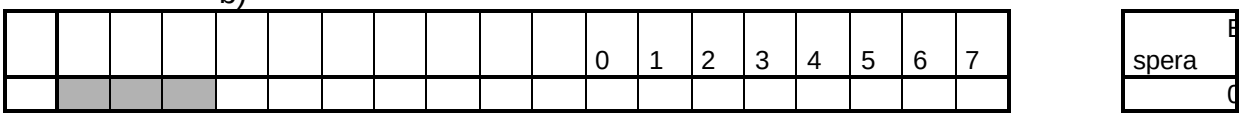


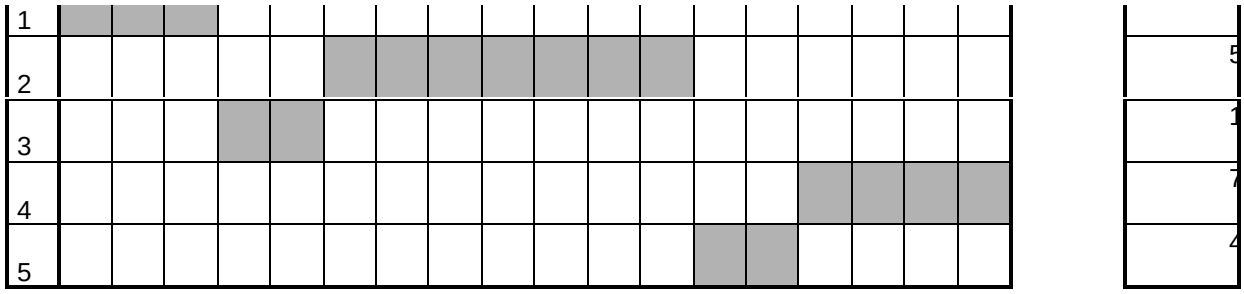
SJF no Apropiativo

a)



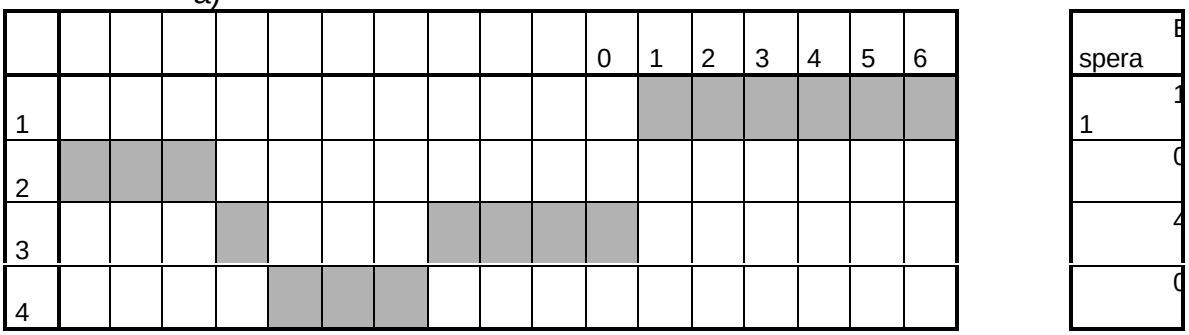
b)



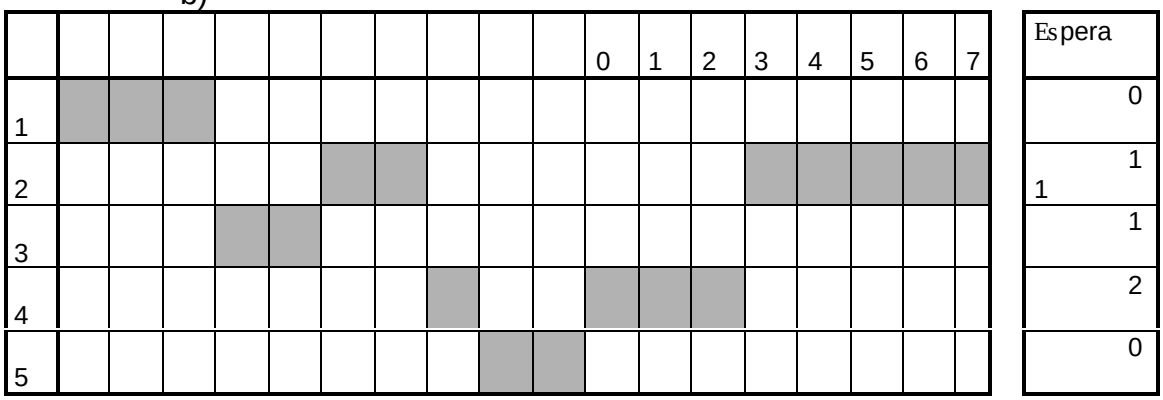


SJF Apropiativo

a)



b)

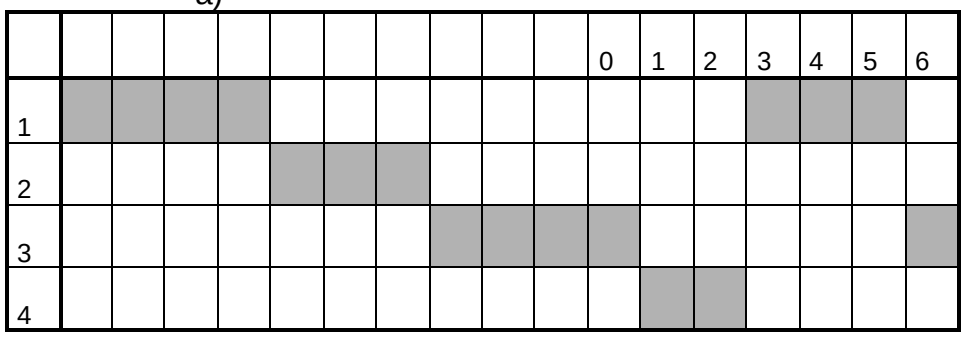


	0	1	2	3	4	5	6	7
1								
2								
3								
4								
5								

Espera
0
11
1
2
0

Algoritmo RR

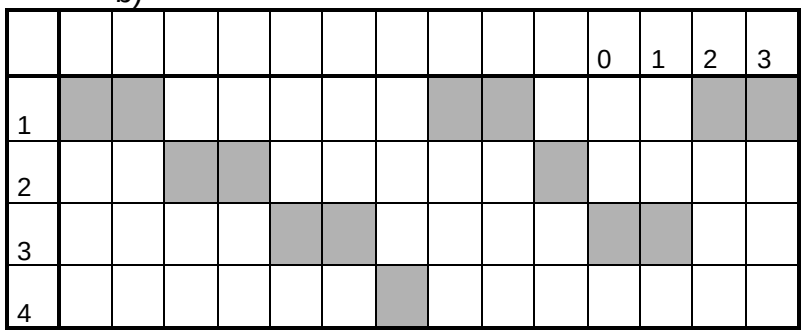
a)



	0	1	2	3	4	5	6
1							
2							
3							
4							

Espera
9
4
12
11

b)

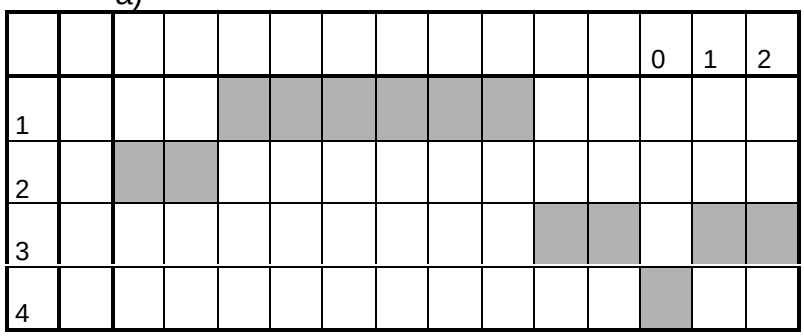


	0	1	2	3
1				
2				
3				
4				

Espera
8
7
8
6

Algoritmo de prioridades apropiativo

a)



	0	1	2
1			
2			
3			
4			

Espera
2
0
6
0

a)

												0	1	2	3
1															
2															
3															
4															

Espera
9
0
9
0

DIRECCIONES DE INTERNET

Si desean ampliar sus conocimientos, le recomendamos consultar las siguientes páginas de Internet:

http://dcc.ing.puc.cl/~jnavarro/iic2332/apuntes/apuntes.html

http://dcc.ing.puc.cl/~jnavarro/iic2332/apuntes/apuntes_12.html

MATERIA: **SISTEMAS OPERATIVOS**
MÓDULO: **GUÍA Nº 2 – ADMINISTRACIÓN DE PROCESOS**

Apellido y Nombre: .....

Lugar: ..... Fecha: .....

Desarrollar

Además de los ejercicios requerido en las explicaciones de SJF, RR y Prioridades desarrolle los siguientes puntos. Una ves desarrollados, acérquelos a su delegación por duplicado para ser corregidos.

1. Diferenciar un proceso de un programa
2. ¿En qué consiste el procesamiento concurrente?
3. ¿Cuáles son los objetivos de un planificador de CPU?
4. Diferenciar la planificación apropiativa de la no apropiativa
5. Destacar los puntos fuertes y puntos débiles de los algoritmos de planificación de CPU FCFS, SJF (en sus dos versiones) RR y PRIORIDADES
6. Según las siguientes tablas de datos desarrollar el algoritmo indicado en cada caso y calcular el tiempo de espera de cada proceso.

a) algoritmo SJF (desarrollarlo como apropiativo y como no apropiativo, con la misma serie de procesos)

Procesos	Llegada	Duración
P1	0	2
P2	2	9
P3	3	2
P4	5	5
P5	10	3

b) Algoritmo Round Robin con cuanta de 4 instantes

Procesos	Duración
P1	3
P2	10
P3	5
P4	8

c) Algoritmo de Prioridades apropiativo

Procesos	Prioridad	Llegada	Duración
P1	2	0	4
P2	1	2	2
P3	3	3	4
P4	4	5	3
P5	1	6	2
P6	2	9	2