Universidad de Carabobo

Facultad de Ciencia y Tecnologia

Sistemas Operativos

Algoritmos de Planificación del Procesador

Victor Tortolero, 24.569.609

25 de marzo de 2016

1. First Come First Serve (FCFS)

Con este algoritmo los procesos se van ingresando en una cola FIFO(First In, First Out), al llegar el proceso se inserta su PCB en la cola, luego si el CPU esta libre toma el primer elemento de la cola y se elimina este de la cola. Como es no apropiativo, la CPU solo toma otro proceso de la cola al terminar el actual. Para implantar este algoritmo solo se requiere una cola FIFO.

– **Ventajas**

- Es fácil de entender e implementar.

– **Desventajas**

- Aunque normalmente es justo en como dedica tiempo de CPU a los procesos, los procesos largos hacen esperar a los cortos.
- El tiempo de espera es alto por lo que carece de rendimiento.

Supongamos que tenemos 3 procesos:

Proceso	Tiempo de Ráfaga
P_0	20
P_1	6
P_2	9

Si los procesos llegan en el siguiente orden, $P_0 \rightarrow P_1 \rightarrow P_2$, tendríamos el diagrama de gantt 1, y el tiempo de espera promedio seria $\frac{0+20+26}{3} = 15{,}333ms$.
En cambio si los procesos llegaran en el orden $P_1 \rightarrow P_2 \rightarrow P_0$, tendríamos el diagrama de gantt 2, y el tiempo de espera promedio seria $\frac{0+6+15}{3} = 8ms$.
En este algoritmo es importante el orden en el que llegan los procesos, y si el tiempo entre ellos varia de gran manera, entonces tendremos un tiempo de espera alto.

Figura 1: FCFS, Ejemplo 1

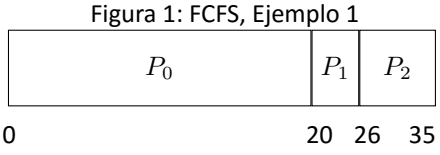
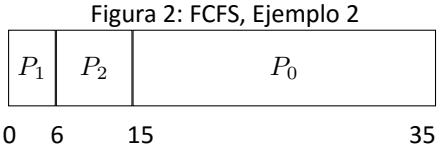


Figura 2: FCFS, Ejemplo 2



2. Shortest Job First (SJF)

Este algoritmo selecciona el proceso con el próximo tiempo de ejecución mas corto(El que tenga la próxima ráfaga de CPU mas corta). Asocia con cada proceso la duración de la siguiente ráfaga de CPU del proceso. Cuando el CPU esta libre se le asigna el proceso que tiene la siguiente ráfaga de CPU mas corta. Si varios procesos tienen la misma duración de ráfaga, se usa el algoritmo FCFS para elegir el proceso.

Para implantar este algoritmo se requiere una cola, y una método para saber la duración de la siguiente ráfaga de CPU para los procesos en la cola, normalmente se usan métodos para hacer aproximaciones o predecir la siguiente rafaga mediante las ráfagas ya conocidas. Este algoritmo pudiera ser el mas rápido, pero es difícil conocer la duración de la siguiente ráfaga.

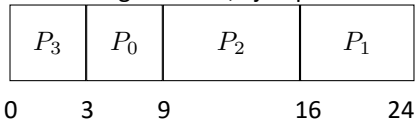
- **Ventajas**
  - Tiempo promedio de espera reducido.
- **Desventajas**
  - Difícil de implementar ya que es complicado conocer la duración de la siguiente ráfaga de CPU para un proceso.

Supongamos que tenemos 4 procesos y que todos llegan al mismo tiempo:

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	6	0
P_1	8	0
P_2	7	0
P_3	3	0

En este caso tenemos un tiempo promedio de espera $\frac{0+3+9+16}{4} = 7ms$.

Figura 3: SJF, Ejemplo



3. Shortest Remaining Time First (SRTF)

Este algoritmo selecciona el proceso que esta en ejecución con otro que exija menor tiempo total de ejecución. Consigue una buena eficiencia, ya que logra que la lista de procesos preparados sea lo más corta posible.

Para implantar este algoritmo necesitamos una cola, y conocer el tiempo total de cada proceso.

– **Ventajas**

- Es eficiente.
- Presenta un buen tiempo promedio de servicio.
- Logra que la lista de procesos preparados sea lo más corta posible.

– **Desventajas**

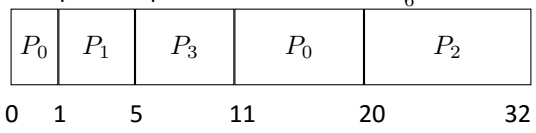
- Los procesos largos no se ejecutaran mientras existan procesos cortos en la cola.

Supongamos que tenemos 4 procesos:

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	20	3
P_1	6	2
P_2	24	0
P_3	8	1

Primero el CPU trataría el P_2 ya que es el primero en entrar a la cola, luego cuando llega el P_3 el algoritmo verifica y tenemos que el tiempo que le queda a P_2(23ms) es mayor al tiempo que le queda a P_3(8ms), por lo que el CPU trata a P_3. Siguiendo estas reglas tendríamos el diagrama de gantt 4.

Figura 4: Tiempo de espera Promedio: $\frac{0+1+2+8+15+35}{6} = 10{,}166ms$



4. Round Robin (RR)

Consiste en conceder a cada proceso en ejecución un determinado período de tiempo, denominado quantum (Q), transcurrido el cual, si el proceso no ha terminado, se le devuelve al final de la cola de procesos preparados y se pasa al proceso que este de primero en la cola. Esta interrupción periódica continúa hasta que el proceso termine su ejecución, formando una rueda de procesos que serán ejecutados cíclicamente hasta que terminen.

Para su implantación requiere una cola (circular) y el valor optimo de Q, el cual se determina según el tipo de sistema y el numero de procesos.

– **Ventajas**

- Es justo con todos los procesos.

– **Desventajas**

- Si el valor de Q es mayor que el tiempo requerido por el proceso mas largo, se convierte en FCFS.
- Si el valor de Q es muy pequeño se producen muchos cambios de contexto lo que es ineficiente.

Supongamos que tenemos 3 procesos:

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	5	0
P_1	3	1
P_2	8	2

Para Q = 4, y Q = 8 tenemos:

Figura 5: Q = 4

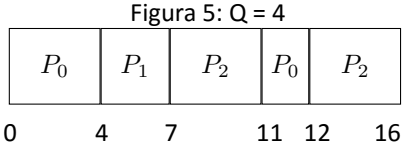
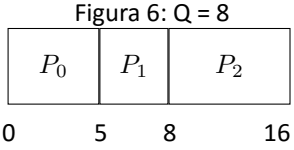


Figura 6: Q = 8



5. Prioridades

Se asocia a cada proceso un numero de prioridad (mientras menor sea el número, mas alta la prioridad), y se ejecuta el proceso con la prioridad mas alta. Si hay varios procesos con la misma prioridad, se resuelve con FCFS. Las prioridades pueden ser definidas interna o externamente. En el primer caso, el sistema operativo se basa en una serie de informaciones medibles para el cálculo y asignación de dichas prioridades (tiempo necesitado de procesador, necesidad de memoria, etc.). Los factores externos son asignados por otro programa o el usuario. La prioridad de un proceso puede cambiar dependiendo si se usa envejecimiento o no. El mecanismo de envejecimiento aumenta la prioridad de un proceso cada vez que pase un tiempo T.

Para su implantación se necesita una cola de prioridades, indicar si las prioridades se definirán de manera interna o externa y la razón de incremento si se usa envejecimiento.

– **Ventajas**

  - Puede ser apropiativo o no apropiativo.

– **Desventajas**

  - Si no se usa envejecimiento, un proceso con muy baja prioridad puede llegar a no ejecutarse nunca.

Supongamos que tenemos 3 procesos:

Proceso	T. de Ráfaga(ms)	T. de llegada(ms)	Prioridad
P_0	5	0	3
P_1	3	1	1
P_2	8	2	2

Tenemos el primer ejemplo sin envejecimiento, y el segundo ejemplo con envejecimiento T = 2 (Cada 2ms la prioridad disminuye en 1).

Figura 7: Prioridades, Sin envejecimiento

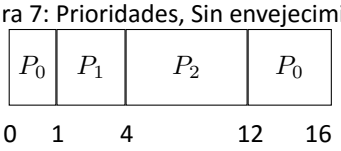
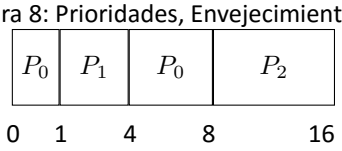


Figura 8: Prioridades, Envejecimiento T=2



6. Colas Multinivel (MLQ)

Con este algoritmo los procesos se agrupan por clasificación y se asignan a diferentes colas, cada cola puede tener su propio algoritmo de planificación. Para elegir que cola usar, se puede usar un algoritmo de prioridades sin envejecimiento, o asignar un porcentaje de tiempo para cada cola. Cada proceso se inserta en una cola y permanece en ella hasta que termine su ejecución.

Para su implantación se debe definir la clasificación para las colas y la prioridad para cada una, y el algoritmo para elegir la cola a utilizar.

– **Ventajas**

- Es muy adaptable a las necesidades del sistema, ya que cada cola puede ser gestionada de forma diferente.

– **Desventajas**

- Las colas requieren un constante monitoreo, y esto es una actividad costosa.

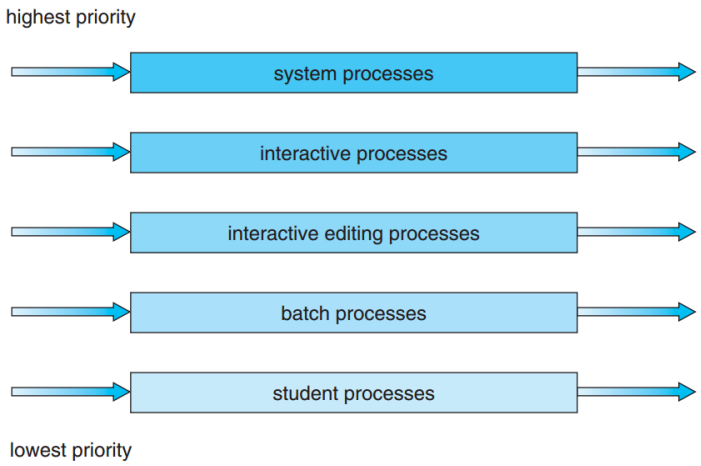


Figura 9: Colas y sus prioridades.

7. Colas Multinivel con Retroalimentacion (MLFQ)

Es parecido al MLQ, pero aqui los procesos no estan restringidos a quedarse en la cola en la que entraron, pueden cambiar de una cola a otra. Este algoritmo agrupa los procesos según las características de sus ráfagas de CPU, si un proceso usa el CPU en mayor nivel, se pasa a una cola de prioridad mas baja. Esto deja a los procesos limitados por E/S y los procesos interactivos en las colas con mayor prioridad. Y los procesos que lleven mucho tiempo esperando en una cola de baja prioridad se pasa a una cola de mayor prioridad (envejecimiento).

Para su implantación necesita el numero de colas, el algoritmo de planificación de cada cola, el método usado para determinar cuando pasar un proceso a una cola de prioridad mas alta, el método usado para determinar cuando pasar un proceso a una cola de prioridad mas baja y el método usado para determinar en que cola se introducirá un proceso cuando haya que darle servicio.

- **Ventajas**
  - Es general y adaptable a cualquier sistema.
- **Desventajas**
  - Es el algoritmo de planificacion mas complejo, requiere de algun mecanismo para definir el valor de todos los parametros.

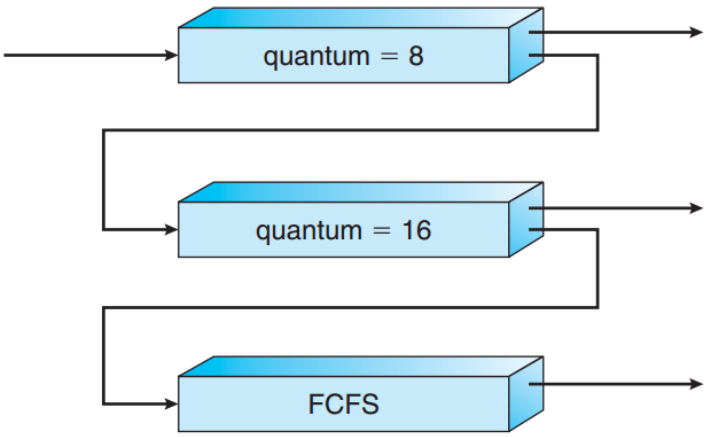


Figura 10: Colas con diferentes algoritmos.

Sistemas de Tiempo Real

Es un sistema en el que se requiere no solo que los resultados calculados sean correctos, sino que también se produzcan dentro de un periodo de tiempo especifico. Los procesos deben terminar su ejecución en un periodo de tiempo.

Esta los Soft Real-Time Systems, que no aseguran cuando un proceso sera ejecutado, pero garantiza que se la dará preferencia (mas prioridad) sobre procesos no críticos. Y los Hard Real-Time Systems, estos requieren que un proceso termine su ejecución en un determinado tiempo, y si no ocurre así el proceso es abortado.

Tenemos los siguientes métodos de planificación:

- **Planificación de tablas estáticas**: Se conocen con anticipación los procesos a realizar, y se tratan como periódicos. Se planifican de tal modo que terminaran su ejecución antes del tiempo determinado incluso en las peores condiciones.
- **Planificación apropiativa con prioridad estática**: Se asignan prioridades a cada proceso relacionadas con el tiempo en que deban ser completados.
- **Planificación dinámica**: Se basa en realizar una prueba de factibilidad de manera dinámica (En tiempo de ejecución). Garantiza que un proceso terminara en el tiempo que debe realizando un plan de ejecución tomando en cuenta el tiempo que necesita para su peor caso y los recursos que necesita.
- **Dynamic best-effort**: Este es parecido a la planificación dinamica pero no realiza una prueba de factibilidad. El sistema hace lo que puede por cumplir los tiempos propuestos. Al no existir garantías, una tarea podría llegar a ser abortada durante su ejecución.

Planificación para Linux

Linux provee soporte para sistemas SMP(Symmetric Multi-Processing o Multiprocesamiento Simétrico, es un tipo de arquitectura de computadores en la que dos o más unidades de procesamiento comparten una única memoria central.), y un algoritmo de planificacion que corre en tiempo $O(1)$ sin importar el numero de procesos. Linux asigna quantum de tiempo mas grande para procesos con mayor prioridad. El algoritmo es basado en prioridades y apropiativo. Linux asigna mas tiempo dedicado a las tareas de prioridad mas alta mas tiempo, y menos tiempo dedicado a las tareas de prioridad mas baja.

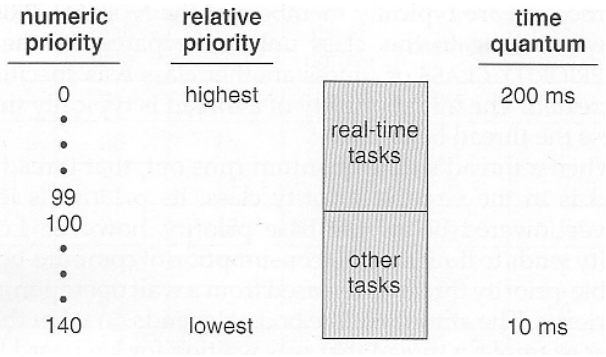


Figura 11: Relación entre prioridades y el tiempo dedicado que poseen.

El kernel mantiene una lista de todas las tareas ejecutables en una estructura de datos denominada cola de ejecución. Debido al soporta para sistemas SMP, cada procesador mantiene su propia cola de ejecución y la planifica de forma independiente. Cada cola de ejecución contiene dos matrices de prioridades, denominadas matrices activa y caducada. Cada una de estas matrices de prioridades contiene una lista de tareas indexada en función de la prioridad. Cuando todas las tareas han caducado sus tiempos totales (cuando la matriz activa está vacía), las dos matrices se intercambian.

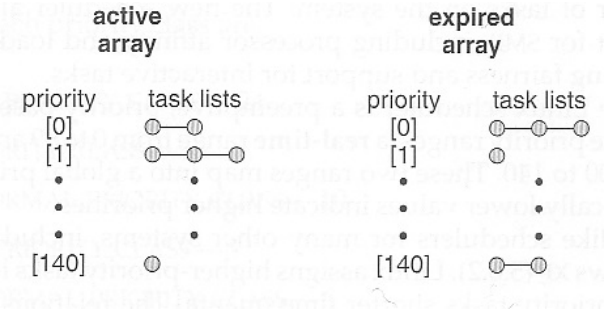


Figura 12: Lista de tareas indexadas por prioridad.

Planificación para Windows

Windows usa un algoritmo apropiativo basado en prioridades. El planificador se asegura que siempre se ejecute el hilo con la prioridad mas alta. Existe un hilo especial “Idle” que se ejecuta cuando ningún otro hilo esta listo. La parte del kernel de Windows que maneja la planificacion es llamada el despachador.

Existe una relación entre las prioridades del kernel de Windows y el API de Windows. El API de Windows identifica 7 clases de prioridades (las filas de la tabla 1) y 6 prioridades relativas dentro de cada clase (las columnas de la tabla 1).

	Real-time	High	Above normal	Normal	Below normal	Idle priority
Time-critical	31	15	15	15	15	15
Highest	26	15	12	10	8	6
Above normal	25	14	11	9	7	5
Normal	24	13	10	8	6	4
Below normal	23	12	9	7	5	3
Lowest	22	11	8	6	4	2
Idle	16	1	1	1	1	1

Cuadro 1: Clases de Prioridades de Windows

A cada proceso se le da una prioridad base dentro de su clase de prioridad. Cuando un proceso consume todo su tiempo disponible, su nivel de prioridad baja, pero nunca menos de su prioridad base. A los procesos en primer plano se les multiplica su quantum de tiempo planificado (el tiempo total dedicado) por 3, para que tengan mejor respuesta.

Planificación en sistemas Multiprocesadores

Cuando se tienen varios procesadores disponibles, la planificación se complica porque ahora hay mas de un CPU al que se debe tener ocupado y en uso efectivo de manera constante. Compartir la carga ayuda ya que balancea el trabajo entre múltiples procesadores. Cuando se planifica para varios procesadores hay que tomar en cuenta si son distintos, o si son idénticos.

- **Distintos (Sistema Heterogéneo)**: Cada procesador tiene su propia cola y algoritmo de planificación.
- **Idénticos (sistema homogéneo)**: Una solución podría ser el multiprocesamiento asimétrico, en el cual un procesador es el master (o líder), controla todas las actividades y corre todo el código del Kernel, mientras que los otros procesadores solo corren código de usuario. Esta solución es relativamente simple ya que no hay necesidad de compartir información critica del sistema.

  Por otro lado también esta el multiprocesamiento simétrico, en el cual cada procesador planifica sus propias tareas, ya sea de una cola en común o de colas separadas para cada procesador. Todos los sistemas operativos modernos soportan el multiprocesamiento simétrico.

Planificación en sistemas Multicore

En el caso de sistemas Multicore, tenemos sistemas con múltiples procesadores pero en un solo chip físico. A pesar de esta diferencia en hardware, el sistema operativo lo percibe de la misma manera, como muchos procesadores a los que debe tener ocupado y en uso efectivo. Se ven mejoras en el consumo de energía.

Planificacion de hilos

En el caso de los hilos, el sistema operativos planifica los hilos a nivel de kernel, los hilos a nivel de usuario son gestionados por una librería de hilos y el kernel no es consciente de ellos. Muchos sistemas consideran una estructura intermedia entre el hilo de kernel y el hilo de usuario. Esta estructura es llamada Light Weight Process o LWP (proceso ligero) y es un medio para lograr multitarea, se ejecuta en el espacio del usuario en la parte superior de un solo hilo de kernel, es decir, por cada hilo de kernel le corresponde un LWP y este es el que planifica los hilos de usuario.

Para ejecutarse en el CPU los hilos de usuarios deben ser asignados a un hilo de nivel de kernel asociado.

La biblioteca de hilos planifica los hilos de usuario para que se ejecuten sobre proceso LWP disponible, lo cual es un esquema conocido con el nombre de ámbito de contienda del proceso(PCS, process contention scope),

dado que la competición por el CPU tiene lugar entre hilos que pertenecen al mismo proceso. Para decidir que hilo del kernel planificar en un CPU, el kernel usa el ámbito de contienda del sistema (SCS), la competición por el CPU con la planificacion SCS toma lugar entre todos los hilos del sistema.

Criterios para evaluar los algoritmos

El primer problema es definir el criterio utilizado para seleccionar un algoritmo. El criterio es definido comúnmente por la utilización del CPU, tiempo de repuesta o el rendimiento. Para seleccionar un algoritmo primero debemos definir la importancia de estos elementos. Una vez definido el criterio a utilizar, podemos usar uno de los siguientes métodos:

- **Modelado determinista**: es un tipo de evaluación analítica. Este método toma una carga de trabajo predeterminada y define el rendimiento de cada algoritmo para esa carga de trabajo. El modelado determinista es simple y rápido, nos da números exactos lo que nos permite comparar los algoritmos.

- **Modelado de colas**: En muchos sistemas los procesos que son ejecutados varían de día a día, así que no existen un conjunto fijo de procesos para usar con modelado determinista. Sin embargo se puede determinar la distribución de las ráfagas de CPU y de E/S. El resultado es una formula matemática que describe la probabilidad de una ráfaga de CPU en particular. También se puede describir al distribución de los tiempos en que los procesos llegan al sistema. Con estas 2 distribuciones es posible calcular el rendimiento, utilización y tiempo de espera promedio para la mayoría de los algoritmos.

  El sistema informático se describe, dentro del modelado de colas, como una red de servidores. Cada servidor dispone de una cola de procesos en espera. El CPU es un servidor con su cola de procesos separados, al igual que el sistema de E/S con sus colas de dispositivos. Conociendo las tasas de llegada y el tiempo de servicio, podemos calcular la utilización, la longitud media ed las colas, el tiempo medio de espera, etc.

  La formula de Little (Little's Formula) dice que para una cola de longitud n, con un tiempo de espera promedio W y una tasa media de llegada de nuevos procesos λ, la relación entre estos valores viene dada por:

$$n = \lambda \times W \tag{1}$$

  Podemos emplear esta formula para calcular una de las 3 variables si conocemos 2 de ellas. El análisis de colas puede resultar útil para comparar los distintos algoritmos de planificacion, aunque también tiene sus limitaciones. Por el momento, los algoritmos que pueden incluirse en el análisis son bastante limitados. Debido a esto y a otras dificultades, a menudo los modelos de cola sólo representan una aproximación a los sistemas reales y la precisión de los resultados obtenidos puede ser cuestionable.

- **Simulaciones**: Las simulaciones nos permiten obtener una evaluación mas precisa de los algoritmos de pla-

nificación. Ejecutar estas requiere programar un modelo del sistema informático. Los componentes principales del sistema se representan mediante estructuras de datos. El simulador tiene una variable que representa una señal del reloj, y cuando el valor de esta variable se incrementa, el simulador modifica el estado del sistema para reflejar las actividades de los dispositivos, los procesos y del planificador. A medida que se ejecuta la simulación las estadísticas que indican el rendimiento del algoritmo se recopilan y se envían a alguna salida. La información para ejecutar la simulación puede ser generada en diversas maneras, el método mas común usa un generador de números aleatorios que esta programado para generar procesos, tiempos de ráfaga de CPU, tiempos de llegada, tiempos de salida de acuerdo con una serie de distribuciones de probabilidad.

Las simulaciones pueden ser costosas, en muchos casos requieren bastante tiempo de computo. Una simulación mas detallada provee resultados mas precisos, pero también requiere mas tiempo.

- **Implementación**: La manera mas precisa de evaluar un algoritmo de planificacion, es programarlo y usarlo en un sistema operativo y observar como funciona. Este acercamiento nos permite observar el algoritmo bajo un sistema operativo en condiciones reales. La principal dificultad es su alto coste, que no solo viene de programar el algoritmo y modificar el sistema operativo para que lo soporte, sino también la reacción de los usuarios a un sistema operativo en constante cambio. La mayoría de los usuarios no están interesados en crear un sistema operativo mejor, simplemente quieren ejecutar sus procesos y utilizar los resultados obtenidos.

  Otra dificultad es que el entorno en el que se use el algoritmo esta sujeto a cambios. El entorno no cambiará unicamente de manera usual, a medida que se escriban nuevos programas y los tipos de problemas cambien, sino también como consecuencia del propio rendimiento del planificador. Si se da prioridad a los procesos cortos, entonces los usuarios pueden dividir los procesos largos en conjuntos de procesos mas cortos. Si se asigna una mayor prioridad a los procesos interactivos que a los no interactivos, entonces los usuarios pueden decidir utilizar procesos interactivos.

Referencias

[1] Krithi Ramamrithan & John A. Stankovic. Scheduling algorithms and operating systems support for real-time systems. Disponible en http://sweet.ua.pt/lda/str/ramamritham94scheduling.pdf.

[2] Justo Sáez Arenas. Instituto nacional de tecnologias educativas y de formacion del profesorado (intef, españa). Disponible en http://mimosa.pntic.mec.es/~jsaez9/Clases/simr/B-Sist_Operativos/T7-Gestion%20de%20Procesos.pdf.

[3] Mercedes Fernández Redondo. Universitat jaume i. Disponible en http://www3.uji.es/~redondo/so/capitulo2_IS11.pdf.

[4] Abraham Silberschatz & Peter Baer Galvin & Greg Gagne. *Fundamentos de Sistemas Operativos, 7ma Edición*. McGraw Hill, 2005.

[5] John T. Bell. University of illinois. Disponible en https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html.

[6] Nada Yahya & Saphia Osama & Arwa Salem. Multilevel queue scheduling. Disponible en https://prezi.com/kzs2ycuj-c65/multilevel-queue-scheduling/.

[7] Planificacion del procesador. Disponible en http://pachel.tripod.com/materias/material3sis.htm.