

Memoria:

Simulador de un planificador a corto plazo

GRUPO 1

José María Cuevas Izaguirre
Paloma Sánchez de la Torre
Cristina Olmedilla López

INDICE

INDICE DE FIGURAS.....	IV
INDICE DE TABLAS.....	V
1. INTRODUCCIÓN.....	1
2. PLANIFICADOR DEL PROCESADOR.....	2
2.1 ¿Qué es un planificador de recursos?.....	2
Objetivos.....	2
2.2 Recursos necesarios para la planificación.....	3
3 TIPOS DE PLANIFICACIÓN.....	3
3.1 La planificación no expulsiva.....	3
3.2 La planificación expulsiva.....	4
4. NIVELES DE PLANIFICACIÓN.....	5
4.1 Planificación a largo plazo.....	5
4.2 Planificación a medio plazo.....	5
4.3 Planificación a corto plazo.....	6
5. PARÁMETROS DE EVALUACIÓN.....	7
¿Qué miden?.....	7
5.1 Parámetros del usuario.....	7
5.2 Parámetros del sistema.....	8
6 ALGORITMOS DE PROGRAMACIÓN.....	9
6.1 Algoritmos de planificación no expulsivos.....	9
6.1.1 Algoritmo Primero en Llegar Primero en Ejecutar (FCFS).....	9
6.1.2 Algoritmo Primero el Trabajo más Corto (SJF).....	10
6.1.3 Algoritmo basado en Prioridades.....	11
6.2 Algoritmos de planificación expulsivos.....	12
6.2.1 Turno Rotatorio (Round Robin).....	13
6.2.2 Algoritmo Shortest Remaining Time First (SRTF).....	13
6.2.3 Algoritmo basado en Prioridades.....	14
6.2.4 Algoritmo Colas Multinivel.....	15
7. CÓDIGO.....	16

7.1. Introducción.....	16
7.2. Descripción general del código.....	16
7.3. Descripción del formato del fichero de entrada.....	17
7.4. Descripción detallada de las estructuras de datos utilizadas.....	18
7.5. Capturas de pantalla con los resultados de la ejecución del programa con los ficheros de entrada del ejercicio propuesto (Ejercicio 4 – Tema 5).....	19
7.6. Descripción del tratamiento de errores realizado sobre el fichero de entrada y capturas de pantalla con la ejecución del programa sobre esos ficheros.....	22
8. CONCLUSIONES.....	24
BIBLIOGRAFIA.....	25

INDICE DE FIGURAS

Figura 1: Ejemplo planificación expansiva.....	4
Figura 2: Planificación a largo plazo.....	5
Figura 3: Planificación a medio plazo.....	6
Figura 4: Planificación a corto plazo.....	6
Figura 5: Transiciones entre niveles.....	7
Figura 6: Primero en llegar, primero en ser servido (FCFS).....	10
Figura 7: Promedio exponencial.....	10
Figura 8: Primero el de duración más corta (SJF).....	11
Figura 9: Cola de usuarios por Prioridad.....	12
Figura 10: Promedio exponencial.....	13
Figura 11: Ejemplo SRTF.....	14
Figura 12: Ejemplo de Prioridades.....	15
Figura 13: Esquema de colas multinivel.....	15
Figura 14: Ejemplo fichero.....	16
Figura 15: Ejemplo fichero entrada ejercicio 4 Figura 16: Ejemplo fichero entrada ejercicio inventado.....	17
Figura 15: Ejemplo fichero entrada ejercicio 4 Figura 16: Ejemplo fichero entrada ejercicio inventado.....	17
Figura 17: Resultados SJF.....	19
Figura 18: Resultados RR-q2.....	20
Figura 19: Resultados RR-q3.....	21
Figura 20: Ejemplo Error 1.....	22
Figura 21: Ejemplo Error 2.....	22
Figura 22: Ejemplo Error 3.....	22
Figura 23: Ejemplo Error 4.....	23
Figura 24: Ejemplo Error 5.....	23

INDICE DE TABLAS

Tabla 1: Ejemplo de FCFS.....	4
Tabla 2: Ejemplo SFJ.....	4

1. INTRODUCCIÓN

A partir de la búsqueda de la eficiencia de los recursos de una computadora, los Sistemas Operativos evolucionaron a lo que se denomina sistema multitarea o multiprogramación. En estos sistemas, la memoria principal contiene varios procesos, que tiene que alternar entre el uso del procesador y esperar que se realice una operación de E/S o que ocurra algún otro evento. Es por eso que el procesador (o los procesadores) se mantienen en gran medida ocupados ejecutando un proceso mientras los demás esperan a que este termine. Estas características hacen necesario la participación de la planificación como clave del éxito de los sistemas de multiprogramación.

En esta memoria abordaremos el concepto de planificación de recursos en todos sus aspectos. Veremos qué diferencias existen entre la planificación expulsiva y la no expulsiva; además observaremos los recursos que son susceptibles de ser planificados, los parámetros de evaluación del mismo, y su descripción desde el punto de vista del usuario y del sistema. También veremos qué objetivos debe lograr dependiendo del sistema, así como los niveles de planificación existentes.

Por otro lado, procederemos a realizar una descripción y a una profundización de los dos tipos de algoritmos mencionados anteriormente, y plantear la aplicación de los mismos dentro de la planificación a corto plazo, el cual nos centraremos en este trabajo.

Finalmente realizaremos una descripción tanto del código realizado como del fichero de entrada y de las estructuras de datos utilizadas, así como de ejemplos gráficos sobre los resultados, sobre la ejecución y sobre el comportamiento de errores.

En este proyecto nos centraremos en crear un simulador en C de un planificador a corto plazo, para poder así realizar un reparto entre varios procesos del tiempo del procesador y así lograr asignar en cada instante un recurso a un determinado usuario.

Esto se realizará en función de dos algoritmos de planificación; el primero de tipo expulsivo (RR, Round Robin) y el segundo no expulsivo (SJF, Shortest Job First).

2. PLANIFICADOR DEL PROCESADOR

2.1 ¿Qué es un planificador de recursos?

Como hemos mencionado en la Introducción, será necesaria la utilización de la planificación de recursos cuando múltiples usuarios necesitan usar de forma exclusiva un determinado recurso.

Objetivos

La planificación va a depender del tipo de sistema al que se va a dirigir, y debe lograr una serie de objetivos. Cabe destacar de forma general: maximizar el número de trabajos por unidad de tiempo, lograr más eficacia mediante la optimización del procesador, que el tiempo de respuesta de uso interactivo sea el mínimo, lograr que la ejecución media de un proceso (thread) sea mínimo, y el que el tiempo de espera sea prácticamente nulo.

Además, hay que mencionar dos tipos de servidores. Por un lado, los servidores interactivos, donde uno de los objetivos principales consiste en la minimización de tiempo de respuesta. Por otro lado, los sistemas por lotes (en inglés batch processing), donde será preciso aumentar por unidad de tiempo el número de trabajos y de la misma forma la eficacia del procesador, mediante su optimización. Es decir, el procesador debe estar ocupado el mayor tiempo posible.

Para poder planificar los recursos vamos a abordar las distintas posibilidades:

Primero hay que tener en cuenta que un recurso puede constar de uno (Recurso simple) o más ejemplares (Recurso múltiple).

Segundo, volvemos a destacar el objetivo principal, la planificación decide en cada instante qué ejemplar de dicho recurso se le asigna a cada uno de los usuarios que soliciten usarlo.

Tercero y último, cuando existe más de un recurso, la planificación lleva a cabo una multiplexación espacial de los recursos y una multiplexación temporal. La multiplexación espacial de los recursos, por definición, es la combinación de dos o más canales de información en un solo medio de transmisión. Por consecuencia, esto consiste en determinar qué recurso se le asigna a cada

usuario. La multiplexación temporal, se lleva a cabo porque también tiene que determinar el instante en el que se le asigna un recurso a un usuario.

2.2 Recursos necesarios para la planificación

Dentro de los recursos que el computador utiliza, podemos destacar aquellos que pueden ser susceptibles a la hora de ser planificados[1]:

El procesador es considerado dentro del computador como el recurso básico. Este recurso es de tipo expropiable, lo cual implica que salva y restaura los registros que se encuentran en el BCP. Además su cola de procesos se identifican con la cola correspondiente a los procesos listos. Por último este recurso muestra la propiedad de la afinidad.

La memoria, ya que en los casos donde los sistemas están provistos de memoria virtual, se origina una multiplexación espacial y temporal de los marcos de página.

El disco, que debe ser planificado debido a que es necesario determinar el orden por el cual los procesos realizarán las peticiones de acceso.

Los mecanismos de sincronización, ya que cuando algunos de estos mecanismos (como el mutex o los cerrojos sobre ficheros) son liberados de un proceso, será necesario planificar el uso de este recurso cuando son designados a nuevos procesos que estuvieran esperando a utilizar este recurso.

3 TIPOS DE PLANIFICACIÓN

Abordamos los tipos de planificación ya que se debe tener en cuenta un aspecto muy importante a la hora de desarrollar un planificador de recursos, ver como se puede acceder a un recurso que está siendo utilizado por otro.

3.1 La planificación no expulsiva

Si nos ponemos en la situación de que un usuario está a la espera y existen recursos libres, se deberá asignar un recurso al usuario, dicho recurso será utilizado por el usuario hasta que termine de utilizarlo. De esta forma, los recursos son *no expropiables*. Algunos ejemplos son:

First-Come First-Served (primero en llegar primero en ser atendido), es una planificación por orden de llegada. En la Tabla 1 podemos observar como se comportaría este algoritmo.

Proceso	Duración
P1	9
P2	4
P3	2

Tabla 1: Ejemplo de FCFS

Shortest Job First (primero la tarea más corta), calcula el tiempo medio de espera, como se puede ver en la Tabla 2:

Proceso	Llegada	Duración	Espera SFJ	Espera SRTF
P1	0	7	0	9
P2	2	4	6	1
P3	4	1	3	0
P4	5	4	7	2

Tabla 2: Ejemplo SFJ

Por prioridad En este caso es recomendable dar prioridad a los procesos que se encuentren en espera.

3.2 La planificación expulsiva

De forma análoga a la planificación no expulsiva mencionada anteriormente, debemos abordar qué no lo es. Poniéndonos en situación, esto es, si existen recursos libres y otra petición de usuario o simplemente él sobrepasa el tiempo máximo de uso. La característica principal es que el recurso es *expropiable*, en el caso de petición de otro usuario, este se quedará con dicho recurso (quitándoselo al usuario anterior - Figura 1).

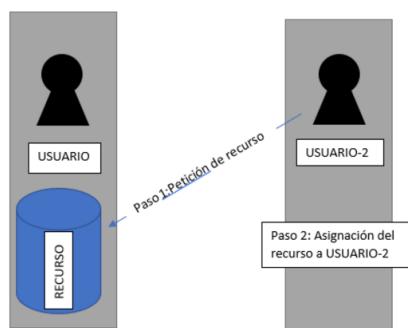


Figura 1: Ejemplo planificación expulsiva

4. NIVELES DE PLANIFICACIÓN

Podemos hablar de tres niveles de planificación, planificación a largo plazo, planificador a medio plazo y planificador a corto plazo.[1][2]

4.1 Planificación a largo plazo

La *planificación a largo plazo* se encarga de los nuevos procesos que llegan al sistema desde la lista de espera, es decir, los siguientes en ser iniciados. Son considerados procesos de tipo batch, mencionado anteriormente. Las ejecuciones no son necesariamente atendidas de inmediato, como se puede observar en la Figura 2, sin embargo se pueden realizar de una manera periódica en el tiempo. Para ejecutar un nuevo proceso, dentro de unos márgenes de espera, hay que considerar qué condiciones han sido declaradas previamente en los procesos y qué condiciones han sido declaradas en el sistema. Actualmente los sistemas son de uso interactivo (el usuario es quien indica qué proceso iniciar) en su mayoría, por lo que ésta planificación se usa para tareas en espera.

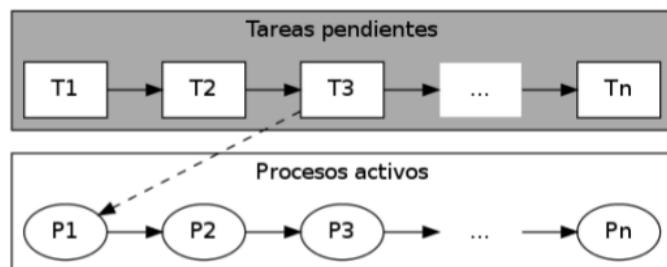


Figura 2: Planificación a largo plazo

4.2 Planificación a medio plazo

La *planificación a medio plazo* (Figura 3) se encarga de la suspensión de procesos. Esto quiere decir que toma las decisiones en cuanto qué procesos son considerados como procesos suspendidos o procesos que dejan de serlo. Esto se puede deber a que un proceso esté esperando a algún evento externo o que esté esperando a que se termine una transferencia de datos. Modifica el grado de multiprogramación en tanto en cuanto excluye o agrega procesos de memoria principal, ya sea porque ésta se encuentre saturada o porque no pueda satisfacer en ese momento la demanda.

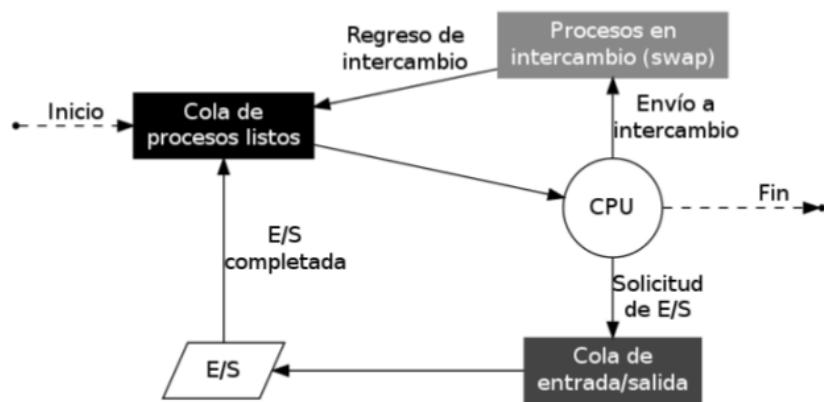


Figura 3: Planificación a medio plazo

4.3 Planificación a corto plazo

La *planificación a corto plazo* tiene por objetivo escoger entre los procesos listos aquel que pase a estado de ejecución, es decir, la que es asignada por el procesador. En la Figura 4 podemos observar como funciona esta planificación. Debido a la asiduidad con la que ésta es activada, es necesario que este planificador se caracterice por su rapidez y por la capacidad de originar poca carga para el procesador. Además su código se debe caracterizar por su simplicidad y eficiencia.

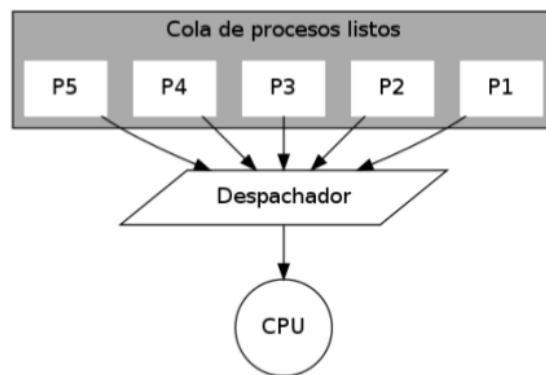


Figura 4: Planificación a corto plazo

En la siguiente figura podemos observar las distintas transiciones entre una planificación y otra (Figura 5). En primer lugar el planificador a largo plazo es el encargado de admitir un nuevo proceso. Después el planificador a mediano plazo se encarga de activar o bloquear un proceso, es decir la zona de intercambio. Por último el planificador a corto plazo dirige qué proceso ejecuta entre los que están preparados, y cuales son los que bloquea.

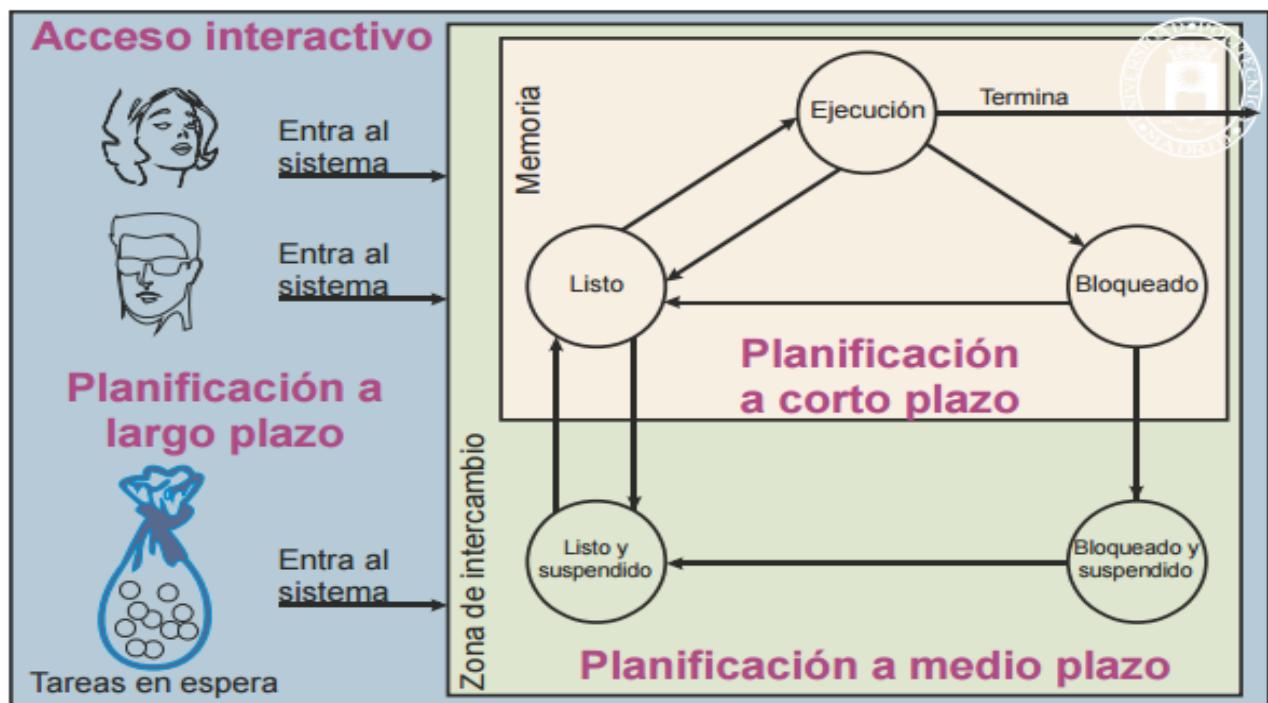


Figura 5: Transiciones entre niveles

5. PARÁMETROS DE EVALUACIÓN

¿Qué miden?

Ya que el planificador tiene como fin optimizar el sistema, tiene que tener en cuenta diversos parámetros de los que hará uso dependiendo de cada situación.

Para definir este comportamiento, nos encontramos con dos tipos de parámetros:

Parámetros de usuario: Estos se refieren al comportamiento del sistema desde el punto de vista de los usuarios o los procesos.

Parámetros de sistema: Son los encargados de conseguir un uso eficiente del procesador.

5.1 Parámetros del usuario

A un proceso le podemos asociar los siguientes parámetros de usuario:

Tiempo de ejecución(Te): Es el tiempo que tarda en ejecutarse un proceso desde que se crea hasta que concluye. Este tiempo abarca todo el tiempo de ejecución de dicho proceso(preparación del proceso, ejecución, y en estado bloqueado).

Tiempos de espera(Tw): Se refiere al tiempo que pasa un proceso en la cola de procesos listos para ejecutar. En el caso de que dicho proceso no se bloquee nunca, sería el tiempo que espera en estado listo para ejecutar antes de que pase al estado de ejecución.

Tiempo de respuesta(Ta): Lo definimos como el tiempo que pasa entre el momento que se crea el proceso y se pone listo para ejecutar y la primera vez que el proceso responde al usuario. Es un parámetro muy importante para los sistemas interactivos.

5.2 Parámetros del sistema

Y los siguientes parámetros de sistema:

Uso del procesador(C): Se trata de un porcentaje que expresa el tiempo útil de uso del procesador partido por el tiempo total (T_u/T).

Tasa de trabajos completados(P): Hace referencia al número de procesos ejecutados completamente por unidad de tiempo, la podemos definir como la inversa del tiempo medio de ejecución.

Otros parámetros que convendría definir son los siguientes:[2]

Tick: Fracción de tiempo en la cual se puede realizar trabajo útil, es decir, poder utilizar el CPU sin interrupciones. Viene determinado por una señal periódica emitida por el temporizador, es variable dependiendo del sistema que se utilice.

Quantum: Con esto nos referimos al tiempo mínimo que se le puede permitir a un proceso para que haga uso del procesador, es variable dependiendo del sistema operativo.

Tiempo núcleo o kernel: Tiempo que ocupa el sistema en el núcleo de este, haciendo funciones tales como la atención a las interrupciones, política de planificación, etc.

Tiempo de sistema: Es el tiempo que utiliza un proceso cuando está usando el núcleo atendiendo el pedido de un proceso (syscall).

Tiempo desocupado: Tiempo en el que la cola de procesos listos está vacía, y por lo tanto no puede realizarse ningún trabajo.

6 ALGORITMOS DE PROGRAMACIÓN

Vamos a describir aquellos algoritmos más importantes y utilizados en la planificación, está utilizará una combinación de ellos en la mayoría de los casos.

6.1 Algoritmos de planificación no expulsivos

Como mencionamos anteriormente, este algoritmo se encuentra dentro de la planificación no expulsiva, que recordamos: solo se utiliza cuando existen usuarios a la espera y dicho recurso está libre. Los usuarios utilizarán exclusivamente los recursos que necesiten hasta que dejen de necesitarlo (no son expropiables).

6.1.1 Algoritmo Primero en Llegar Primero en Ejecutar (FCFS)

El algoritmo FCFS: es un algoritmo muy simple, básicamente se encarga de asignar al usuario que más tiempo lleve en la cola de listos. Como mencionamos anteriormente, en la planificación a corto plazo, escoge entre los procesos listos aquel que pasa a estado de ejecución. Es decir, el procesador ejecuta cada proceso hasta que termina y los procesos que se encuentran en la cola de listos esperarán hasta que llegue su turno (este lo determinará el que más tiempo lleve en espera). La cola utiliza la disciplina FIFO (First In First Out).

Este se caracteriza por:

- Lograr obtener el máximo rendimiento del procesador, al no causar a penas sobrecarga.
- Los tiempos de espera son muy elevados, por ello, no es recomendable en sistemas interactivos, y es recomendable su utilización en aquellos procesos intensivos en procesamiento.

Resumiendo todo esto, veamos un ejemplo (Figura: 6). En esta figura observamos cómo se concede el procesador a aquellos procesos que primero han llegado y más tiempo llevan esperando en la cola.

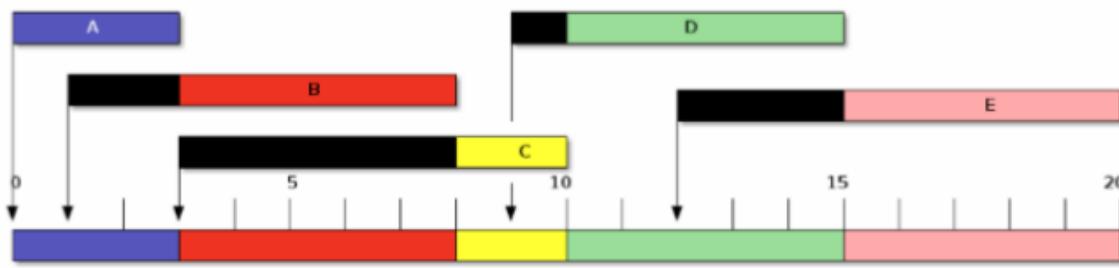


Figura 6: Primero en llegar, primero en ser servido (FCFS)

6.1.2 Algoritmo Primero el Trabajo más Corto (SJF)

Existen dos versiones de este algoritmo, una expansiva denominada SRTF, y la otra denominada SJF que se abordará en este punto.

El algoritmo SJF: de la cola de procesos listos, buscará aquel cuyo tiempo de ejecución sea más corto. Para utilizar este algoritmo será necesario disponer de la duración de cada proceso listo, esto solo será posible en los sistemas por lotes. Por este motivo, para poder utilizarlo se estiman estas duraciones utilizando la información de los ciclos ejecutados anteriormente.

De esta forma, el éxito de dicho algoritmo está directamente relacionado con la estimación realizada.

Para estimar el tiempo que requerirá un proceso, se suele utilizar un promedio exponencial denominado 'ep', este promedio se realiza de la siguiente forma[2]:

1. Se define un factor entre $0 \leq f \leq 1$, que determinará qué tan reactivo será el promedio obtenido a la última duración (Figura 7); este valor suele ser cercano a 0.9.
2. Si el proceso utiliza 'q' quantums durante su última invocación, se puede tomar como semilla para el 'ep' inicial un número elegido arbitrariamente, o uno que ilustra el comportamiento actual del sistema (como el promedio del 'ep' de los procesos actualmente en ejecución).

$$e'_p = fe_p + (1 - f)q$$

Figura 7: Promedio exponencial

Ventaja a destacar:

- Reduce el tiempo de espera medio para los procesos, ya que penaliza los procesos largos.

Inconvenientes:

- Riesgo de inanición de los usuarios de larga duración.
- Sobrecarga del sistema, ya que debe recorrer la cola de procesos listos, para encontrar aquel el que sea más corto. En otro caso la cola debe estar ordenada y además se debe realizar una estimación.

Resumiendo, observamos en la figura 8 el comportamiento general del algoritmo SJF, donde se atienden por prioridad de duración más corta a los procesos.

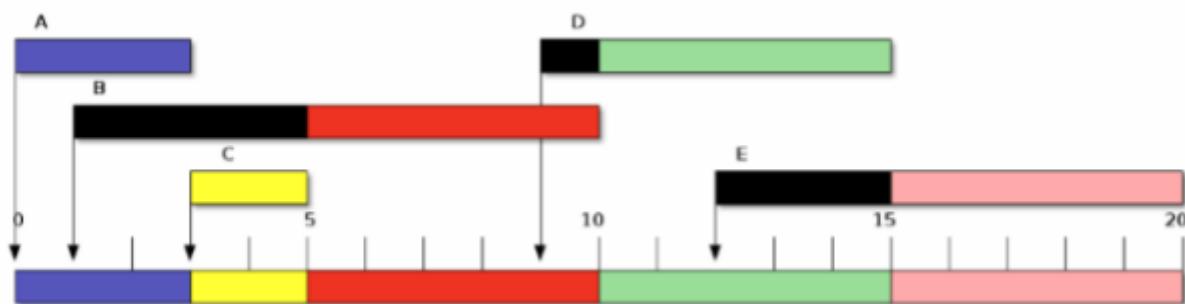


Figura 8: Primero el de duración más corta (SJF)

6.1.3 Algoritmo basado en Prioridades

El planificador selecciona al usuario con mayor prioridad, en el caso de que existan varios usuarios con igual prioridad se hará uso de otros algoritmos, como por ejemplo el FCFS:

-El algoritmo FCFS (First Come First Served) selecciona al usuario que lleva más tiempo esperando en la cola de listos, dada su sencillez es un algoritmo sencillo que casi no sobrecarga el sistema, por lo que se consigue el máximo rendimiento del procesador. Tampoco es expulsivo, por lo que los usuarios mantienen el recurso hasta que dejan de necesitarlo. Este algoritmo beneficia a los procesos intensivos en procesamiento frente a los intensivos de E/S, como contra tenemos que puede producir largos tiempos de espera, por lo tanto no es adecuado para sistemas interactivos.

El planificador tiene la cualidad de proporcionar grados de urgencia, el sistema mantiene una cola de usuarios por prioridad, como podemos observar en la figura 9:

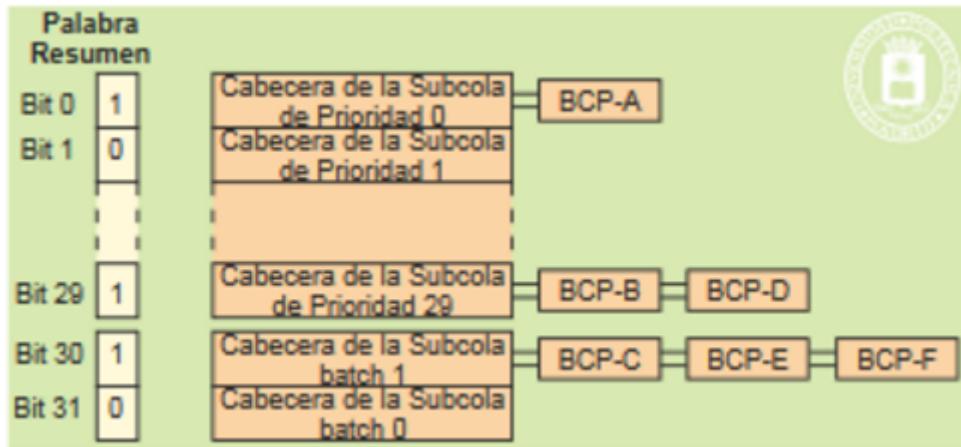


Figura 9: Cola de usuarios por Prioridad

Las colas de procesos organizados por prioridad tienen el inconveniente de que pueden producir inanición en los usuarios de baja prioridad, para solventar esto se hace uso de prioridades dinámicas y mecanismos de envejecimiento, los cuales aumentan la prioridad a los usuarios que lleven más tiempo esperando a ser atendidos. Podemos encontrar una versión expulsiva de este algoritmo, el cual expulsa a un usuario si llega otro con más prioridad que el que se está ejecutando.

6.2 Algoritmos de planificación expulsivos

Tenemos que tener en cuenta que los algoritmos no expulsivos no son los más adecuados a la hora de realizar una planificación en un sistema operativo de propósito general. Entre sus debilidades podemos encontrar que los grados de urgencia de sus algoritmos no están implementados de la forma que un sistema de propósito general requeriría y que estos realizan un reparto deficiente del procesador. En este caso los algoritmos de planificación expulsivos son los indicados para resolver estas deficiencias.

6.2.1 Turno Rotatorio (Round Robin)

Este algoritmo tiene como función repartir de forma equitativa el procesador entre los procesos listos, dando así un tiempo de respuesta predeterminado, tiene como punto positivo que ofrece un reparto equitativo.

Lo podemos definir como una variación del algoritmo FCFS, ya que se asigna el procesador por un tiempo máximo acotado que se denomina rodaja o cuanto. En caso de que se cumpla el cuanto sin que el proceso abandone voluntariamente el procesador, dicho proceso será expulsado y puesto al final de la cola, tal y como podemos observar en la figura 10:

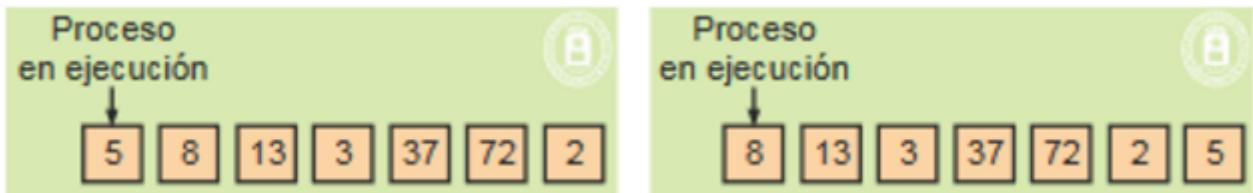


Figura 10: Promedio exponencial

Tenemos que tener en cuenta la dimensión del cuanto, si este es muy grande el algoritmo tenderá a ser igual que el FCFS, y si es muy pequeño se introducirá mucha sobrecarga en el sistema. Podemos encontrar diversos diseños para el cuanto:

- Cuanto igual para todos los procesos.
- Cuanto distinto según el tipo de proceso.
- Cuanto dinámico, su valor variará según sea el comportamiento del proceso o del comportamiento global del sistema.

Aumentando el cuanto conseguiremos una mejora de los tiempos promedio de respuesta, pero si lo aumentamos hasta convertirlo en un FCFS efectivo degeneraría en una penalización a los procesos cortos, pudiendo llegar a la inanición. Según mediciones estadísticas, se dice que el cuanto normalmente debe mantenerse inferior a la duración promedio del 80% de los procesos [2].

6.2.2 Algoritmo Shortest Remaining Time First (SRTF)

Se puede considerar como el algoritmo expulsivo equivalente al SJF (Shortest Job First). Este algoritmo se encarga de escoger un proceso que tenga el menor tiempo de ejecución restante estimado, pero en este caso cada vez que un

proceso cambie su estado a listo se tendrá que verificar que su tiempo de ejecución restante es menor que el restante del proceso. Si esto fuera así, el proceso actual se expulsará del procesador, escogiendo al nuevo. Una vez un proceso termine, se escogerá al siguiente que tenga el menor tiempo estimado de ejecución restante.

Lo que se consigue con este algoritmo es que aunque no exista un reparto igualitario del procesador y en nivel de urgencia que un proceso pueda tener se reducen de manera significativa los tiempos de espera.

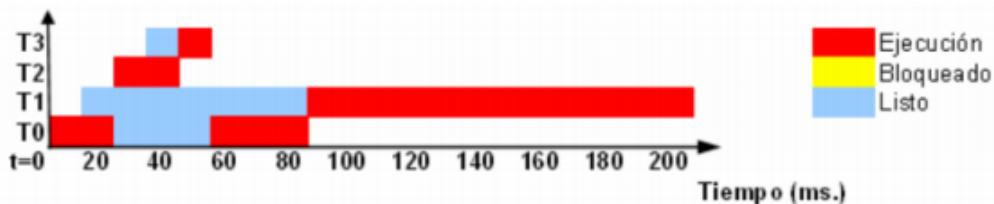


Figura 11: Ejemplo SRTF

En la figura 11 podemos observar como funciona este algoritmo. Aquí se puede observar que T1 se encuentra Listo, pero al ser su tiempo mayor a T0, el primero sigue ejecutándose. Por el contrario cuando T2 se encuentra listo, éste expulsa a T0 del procesador ya que su tiempo de ejecución es menor que T0. Por otro lado T3 tendrá que esperar a que T2 termine ya que cuando este está listo, ambos coinciden en el tiempo de ejecución restante. Por último se ejecutan los procesos listo T0 y T1 respectivamente, ya el primero tiene un menor tiempo de ejecución que el segundo.

6.2.3 Algoritmo basado en Prioridades

Este algoritmo presenta las mismas características que su análogo no expulsivo. Lo que encontramos diferente en este algoritmo es que se ejecutará cuando un proceso se añada a la lista de procesos en el estado listo y la prioridad de este sea mayor. Por otro lado, si un proceso en ejecución disminuyera su prioridad, se tendría que ejecutar de nuevo este algoritmo.

También tendremos que considerar si el núcleo es expulsivo o no. Si no lo fueran, tendremos que esperar a que las rutinas de interrupción y todas las llamadas a sistema anidadas (si estas existen) terminen para poder expulsar un proceso del procesador. Esto puede desencadenar un aumento en el tiempo de respuesta (Las llamadas a sistema suele tomar mas tiempo que las interrupciones) Sin embargo, en el primer caso, sólo será necesario esperar a que las interrupciones anidadas terminen.

Este algoritmo es adecuado cuando queremos conseguir un mejor tiempo de respuesta para los procesos urgentes, aunque perderemos en cuanto a reparto igualitario del procesador y los procesos que hagan uso en gran medida de E/S, no serán favorecidos a menos que su prioridad esté condicionada en este aspecto.

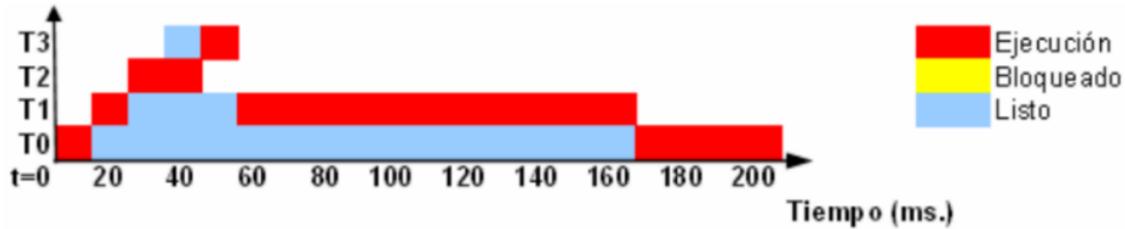


Figura 12: Ejemplo de Prioridades

En la figura 12 podemos ver como funciona el algoritmo: en primer lugar observamos que T0 T1 y T2 tienen una prioridad progresiva. T0 será expulsado por T1, que a su vez será expulsado por T2. Sin embargo, T3 no tiene una prioridad mayor a T2, por lo que tendrá que esperar a que éste último termine su ejecución para realizar la suya propia.

6.2.4 Algoritmo Colas Multinivel

Este algoritmo se puede considerar distinto al resto de los que hemos estudiado con anterioridad. Aunque corresponde a los algoritmos de tipo expulsivo, éste es el único que trata los procesos dentro de distintos niveles de procesos. Esta distinción hace que se diferencie del resto de los algoritmos de planificación en tanto en cuanto los primero tratan a todos los procesos como iguales.

Esta diferenciación por niveles dará la posibilidad de utilizar una estructura de planificación específica para cada clase (Figura 13)

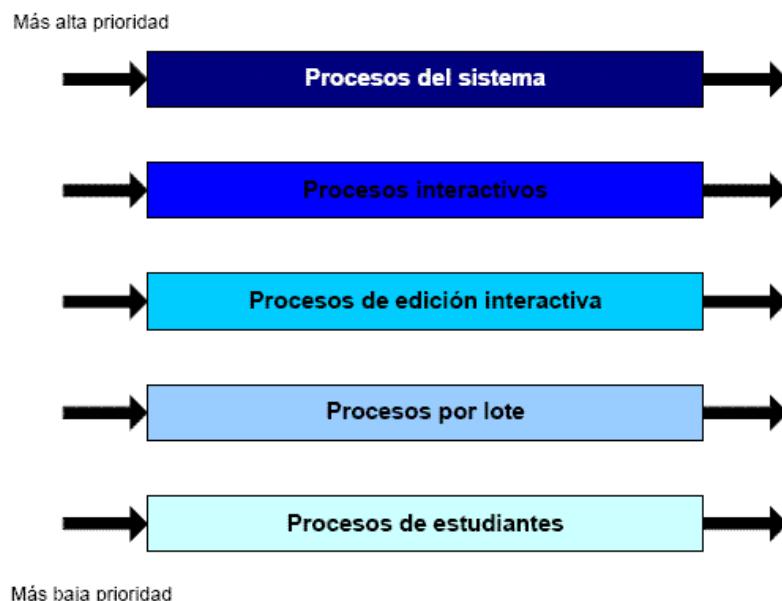


Figura 13: Esquema de colas multinivel

7. CÓDIGO

7.1 Introducción

La siguiente sección tratará el código utilizado para la realización del objetivo final, crear un simulador en C de un planificador a corto plazo, para poder así realizar un reparto entre varios procesos del tiempo del procesador y de esta forma lograr asignar en cada instante un recurso a un determinado usuario.

Esto se realizará en función de dos algoritmos de planificación: el primero de tipo expulsivo (RR, Round Robin) y el segundo no expulsivo (SJF, Shortest Job First).

7.2. Descripción general del código

De forma general se utiliza una estructura concreta en los ficheros pasados como parámetro, dichos ficheros utilizarán la extensión .txt y en ellos se incluirán los datos necesarios para la aplicación de los distintos algoritmos (Fig. 14):

- La primera línea contendrá el número de procesos con la sintaxis: p='número_de_procesos'.
- La segunda línea contendrá el número de quantum con la sintaxis: q='número_de_quantum'.
- La tercera línea y en adelante contendrá: 'nombre del proceso', 'tiempo de llegada' y 'tiempo de ejecución' (acabando cada línea con salto de línea y entre cada atributo un espacio en blanco).

```

p='número_de_procesos'
q='quantum'
A 'Tiempo de llegada' 'Tiempo de ejecución'
B 'Tiempo de llegada' 'Tiempo de ejecución'
C 'Tiempo de llegada' 'Tiempo de ejecución'
.
.
.
```

Figura 14: Ejemplo fichero

En un primer instante nuestro código comprobará los distintos errores: no se ha introducido el número de parámetros adecuados, no se ha introducido un parámetro válido (porque no existe dicho fichero), el fichero no contiene más de un proceso, la primera línea del fichero no contiene el número de procesos, la segunda línea del fichero no contiene el número de quantum empleado en el algoritmo Round Robin y el número de procesos no es igual a los procesos existentes en dicho fichero.

Después se pasará a ejecutar el código sin errores, donde se utilizarán los dos algoritmos planteados: SJF y Round Robin. Tras dicha ejecución se crearán dos ficheros con extensión .txt una para cada uno de ellos y contendrá el reparto temporal correcto de la CPU entre todos los procesos, el porcentaje de utilización de la CPU y para cada proceso la información relativa a su tiempo de permanencia, tiempo de respuesta y tiempo de espera.

7.3. Descripción del formato del fichero de entrada

El formato que utilizaremos para el fichero de entrada será el mencionado anteriormente: ficheroEntrada.txt, el cual pasaremos como parámetro por la ruta correcta. Por ejemplo:

"C:\Users\USUARIO\Documents\NetBeansProjects\SIMULADOR\ficheroEntrada.txt"

De forma general deberá contener en una primera línea el número de procesos que se van a tratar, en la segunda línea deberá incluir el número de quantum destinado a utilizarse en el algoritmo Round Robin, y en el resto de líneas estarán los procesos con sus datos correspondientes (tiempo de llegada y tiempo de ejecución).

En la siguientes figuras observamos ejemplos de ficheros de entrada con extensión .txt para los ejercicios planteados (Fig 15 y 16):

p=5
q=2
A 3 9
B 5 3
C 7 2
D 8 5
E 11 1

p=7
q=2
A 0 5
B 1 4
C 3 1
D 9 7
E 1 7
F 5 4
G 2 8

Figura 15: Ejemplo fichero entrada ejercicio 4

Figura 16: Ejemplo fichero entrada ejercicio inventado

7.4. Descripción detallada de las estructuras de datos utilizadas

- Inicialización de estructura para definir atributos de un proceso:

char* nombre:	Nombre del proceso.
int llegada:	Ciclo de llegada.
int copiaLlegada:	Copia para uso en RR.
int vacio:	Comprueba si se ha sido visitado.
int tiempoEjec:	Tiempo de ejecución de un proceso (En ciclos).
int ciclosEj:	Ciclos ejecutados por el proceso.
int tPermanencia:	Tiempo de permanencia (en estado listo).
int tRespuesta:	Tiempo en el que comienza a ejecutar.
int tEspera:	Tiempo de espera.

- Inicialización de estructura para definir atributos de la CPU:

int ciclosTotal:	Ciclos totales.
int ciclosCPU:	Ciclos en los que la CPU está trabajando.
char* liTiempo:	Línea temporal donde se observan los procesos ejecutados en cada ciclo

- Inicialización de estructura de enumeración para SJF y FIFO (los dos algoritmos utilizados para las colas).
- Otras variables dentro de cada función.

7.5. Capturas de pantalla con los resultados de la ejecución del programa con los ficheros de entrada del ejercicio propuesto (Ejercicio 4 – Tema 5)

- Utilizando el algoritmo de SJF (Shortest Job First). Fig.17:

Datos de la CPU

Línea de tiempo:

- nop - nop - nop - A - A - A - A - A - A - A - A -
 E - C - C - B - B - B - D - D - D - D - D -
 Uso de la CPU en tanto por ciento: 86.956519%

PROCESOS

Nombre de proceso A

Acaba en el ciclo 12

Tiempo Permanencia = 9

Tiempo de Respuesta = 0

Tiempo de Espera = 0

Nombre de proceso E

Acaba en el ciclo 13

Tiempo Permanencia = 2

Tiempo de Respuesta = 1

Tiempo de Espera = 1

Nombre de proceso C

Acaba en el ciclo 15

Tiempo Permanencia = 8

Tiempo de Respuesta = 6

Tiempo de Espera = 6

Nombre de proceso B

Acaba en el ciclo 18

Tiempo Permanencia = 13

Tiempo de Respuesta = 10

Tiempo de Espera = 10

Nombre de proceso D

Acaba en el ciclo 23

Tiempo Permanencia = 15

Tiempo de Respuesta = 10

Tiempo de Espera = 10

Figura 17: Resultados SJF

- Utilizando el algoritmo de RR (Round Robin) con quantum igual a 2.
- Fig.18:

Datos de la CPU

Línea de tiempo:

- nop - nop - nop - A - A - A - A - B - B - A - A - C -
C - D - D - B - A - A - E - D - D - A - D -

Uso de la CPU en tanto por ciento: 86.956519%

PROCESOS

Nombre de proceso C

Acaba en el ciclo 13

Tiempo Permanencia = 6

Tiempo de Respuesta = 4

Tiempo de Espera = 4

Nombre de proceso B

Acaba en el ciclo 16

Tiempo Permanencia = 11

Tiempo de Respuesta = 2

Tiempo de Espera = 8

Nombre de proceso E

Acaba en el ciclo 19

Tiempo Permanencia = 8

Tiempo de Respuesta = 7

Tiempo de Espera = 7

Nombre de proceso A

Acaba en el ciclo 22

Tiempo Permanencia = 19

Tiempo de Respuesta = 0

Tiempo de Espera = 10

Nombre de proceso D

Acaba en el ciclo 23

Tiempo Permanencia = 15

Tiempo de Respuesta = 5

Tiempo de Espera = 10

Figura 18: Resultados RR-q2

- Utilizando el algoritmo de RR (Round Robin) con quantum igual a 3.
- Fig.19:

Datos de la CPU

Línea de tiempo:

- nop - nop - nop - A - A - A - B - B - B - A - A - A -

C - C - D - D - D - E - A - A - A - D - D -

Uso de la CPU en tanto por ciento: 86.956519%

PROCESOS

Nombre de proceso B

Acaba en el ciclo 9

Tiempo Permanencia = 4

Tiempo de Respuesta = 1

Tiempo de Espera = 1

Nombre de proceso C

Acaba en el ciclo 14

Tiempo Permanencia = 7

Tiempo de Respuesta = 5

Tiempo de Espera = 5

Nombre de proceso E

Acaba en el ciclo 18

Tiempo Permanencia = 7

Tiempo de Respuesta = 6

Tiempo de Espera = 6

Nombre de proceso A

Acaba en el ciclo 21

Tiempo Permanencia = 18

Tiempo de Respuesta = 0

Tiempo de Espera = 9

Nombre de proceso D

Acaba en el ciclo 23

Tiempo Permanencia = 15

Tiempo de Respuesta = 6

Tiempo de Espera = 10

Figura 19: Resultados RR-q3

7.6. Descripción del tratamiento de errores realizado sobre el fichero de entrada y capturas de pantalla con la ejecución del programa sobre esos ficheros.

ERROR NUMERO 1: Número de parámetros.

Este error aparecerá en aquellos casos donde no se hayan introducido los parámetros adecuados (Fig 20). En nuestro caso será necesario la inserción de la ruta correspondiente al archivo que queremos utilizar.

```
falta el parametro de llamada.
Sintaxis Correcta: Planificador_S0.exe "C:\ficheroEntrada.txt"
Press [Enter] to close the terminal ...
```

Figura 20: Ejemplo Error 1

ERROR NUMERO 2: Existencia de parámetro

Si la ruta facilitada no existe o está mal escrita, aparecerá el siguiente mensaje(Fig 21):

```
fopen() failed in file main.cpp at line # 96
ERROR: el fichero pasado como parametro no existe.
Press [Enter] to close the terminal ...
```

Figura 21: Ejemplo Error 2

ERROR NUMERO 3: Introducción de número de procesos

Este error ocurrirá en aquellos casos donde la primera linea de un archivo no contenga la sintaxis adecuada, como en la Fig. 22:

```
El fichero de entrada no contiene en la primera linea: p='numero'
Press [Enter] to close the terminal ...
```

Figura 22: Ejemplo Error 3

ERROR NUMERO 4: Introducción de quantum

Este error ocurrirá en aquellos casos donde la segunda linea de un archivo no contenga la sintaxis adecuada, como en la Fig. 23:

```
El fichero de entrada no contiene en la segunda linea: q='numero'  
Press [Enter] to close the terminal ...
```

Figura 23: Ejemplo Error 4

ERROR NUMERO 5: Datos sobre los procesos

Si por algún motivo algunos de los procesos no tuviera alguno de sus parámetros (Tiempo de llegada o Tiempo de ejecución), este no podría ser procesador, con el consiguiente mensaje de error (Fig. 24):

```
Error en el fichero de entrada.  
Formato: <Nombre> <TLlegada> <TEjecucion>.  
Press [Enter] to close the terminal ...
```

Figura 24: Ejemplo Error 5

8. CONCLUSIONES

Tras todo lo visto en este documento, podemos afirmar que la planificación es un elemento de vital importancia en los sistemas operativos, en concreto en los sistemas multiprogramación, ya que el concepto de planificación de recursos surgió a partir de estos sistemas.

La planificación permite ejecutar un mayor número de procesos en menos tiempo y permite optimizar el procesador ya que el tiempo de espera es prácticamente nulo, ejecuta los procesos en poco tiempo y, además hace que el tiempo de respuesta entre el usuario y la máquina sea mínimo.

Podemos definir la planificación como la serie de comportamientos y mecanismos que utiliza el sistema operativo para determinar cuándo se ejecutarán los programas en el procesador.

Empezamos viendo cuáles eran los recursos que susceptibles a ser planificados y los parámetros de evaluación que se utilizan para asignarlos. Luego de conocer los tipos de planificación (Expulsiva y no expulsiva) vimos los niveles de planificación (corto, medio y largo plazo), para finalmente ahondar en los algoritmos de planificación más importantes.

Ya que nos hemos centrado en este trabajo en la planificación a corto plazo, haciendo hincapié en su funcionamiento, cuyo objetivo es el de escoger entre los procesos listos aquel que pase a estado de ejecución, procedimos a realizar un código el cual consistiría en el diseño de un planificador a corto plazo. Su código se debía caracterizar por su simplicidad y eficiencia.

Respecto a la elaboración del código del planificador en C, hemos sacado en conclusión que es un trabajo muy interesante, a la vez que laborioso e incluso tedioso. También podemos destacar que ha sido un punto de unión entre las clases de teoría y las prácticas, lo que ha hecho asentar completamente los conocimientos sobre la planificación de procesos, y también ha sido un recordatorio de programación en C.

A medida que se ve aumentado el número de procesos, se hace un uso más eficiente del procesador, observamos en nuestro programa como el ejercicio cuatro hace un uso de la CPU de un 86.956519% con cinco procesos, mientras que si aumentamos a siete procesos el uso de la CPU es de un 100%.

BIBLIOGRAFIA

- [1]: Pérez, J. C., Carballeira, F. G, de Miguel Anasagasti, P., & Costoya, F. P. (2001) Sistemas operativos. McGraw-Hill Interamericana
- [2]: Wolf, G., Ruiz, E., Bergeo, F., Meza, E. (2015) Fundamentos de Sistemas Operativos. Universidad Nacional Autónoma de México