Autores, alumnos del grupo 81:

Alberto Gómez Aparicio, 100363805

Carlos García Cañibano, 100363813

Juan José Garzón Luján, 100363861

Planificación de procesos

Diseño de Sistemas Operativos

Contenido

[Introducción 2](#_Toc3915129)

[Descripción del código 2](#_Toc3915130)

[Round-Robin 2](#_Toc3915131)

[Round-Robin y FIFO con prioridades 2](#_Toc3915132)

[Round-Robin y FIFO con posibles cambios de contexto voluntarios 3](#_Toc3915133)

[Batería de pruebas 3](#_Toc3915134)

[Conclusión 5](#_Toc3915135)

# Introducción

En esta memoria documentaremos algunos aspectos sobre la practica que hemos realizado, que consiste en el desarrollo de un planificador de hilos en el espacio de usuario escrito en el lenguaje C.

A grandes rasgos consiste en la creación de diversas políticas de planificación de procesos (pero en esta práctica utilizaremos procesos ligeros o *threads*).

# Descripción del código

Algo fundamental en el planificador son las estructuras de datos necesarias para el funcionamiento y dado que cada apartado de esta práctica se construye utilizando el anterior, todas las estructuras se conservan para el siguiente, añadiendo las necesarias.

Las tablas de control de proceso, **TCB**, contienen toda la información relativa a un proceso, como su id, el estado en el que se encuentra, la prioridad, los ticks de reloj que le quedan para que acabe su rodaja, la función que ejecutan y el contexto.

Además, nos apoyamos sobre el TAD lineal de **cola**, cuyo funcionamiento es el típico de esta estructura; las principales operaciones son encolar (añade un elemento al final) y desencolar (elimina el primer elemento).

Para simular **interrupciones** tipo *hardware* del **reloj**, utilizaremos el código adjunto a la práctica que se basa en señales cada cierto tiempo.

## Round-Robin

La primera política a desarrollar es **Round-Robin**, cada hilo tiene asignado unos ticks de reloj (rodaja de tiempo) y cuando se le acaben, el procesador le expulsará y pasara al final de la cola de hilos preparados. Para implementar esto, utilizaremos una cola de TCB para dar el orden y pasarlo al final cuando sea preciso.

Cuando haya habido suficientes interrupciones de reloj para que en la tabla del proceso su atributo *ticks* haya llegado a cero, puede ocurrir que no haya otro proceso listo para ejecutarse, y solo le resetearemos la rodaja para que siga ejecutándose el actual; o que haya otros, desencolaremos el siguiente y encolaremos al final el que termino su rodaja y cambiaremos el contexto al siguiente (además de resetear).

Terminaremos si todos los hilos han completado su trabajo y la cola de TCB está vacía.

## Round-Robin y FIFO con prioridades

La siguiente política es añadir un mecanismo **FIFO** para una nueva prioridad de procesos. Los hilos de prioridad baja siguen la anterior política, y los que tengan alta prioridad expulsarán a cualquiera de menor prioridad (si hay) y completaran su tarea.

Para dar soporte a este nuevo funcionamiento, nos hace falta una cola para los hilos de baja prioridad (al igual que antes) y otro para los de alta. Cuando se crea un hilo de alta se añade al final de su cola y provoca una llamada al planificador para ejecutarse instantáneamente, salvo si hay otros hilos de alta prioridad que tengan que completarse.

## Round-Robin y FIFO con posibles cambios de contexto voluntarios

Y la última política es dar la posibilidad a los hilos de hacer un **Cambio de Contexto Voluntario** y así continuar con la ejecución de otro que esté listo para ejecutar.

Como tenemos un nuevo dispositivo que provoca esperas, añadiremos otra cola en la que estarán los hilos que se bloqueen por este. Además, hay que desarrollar la interrupción hardware de disco y la función *read\_disk*, que permitirá este **CCV**.

Cuando un hilo quiera cambiar el contexto, cambiará su estado a “Esperando”, se desencolará de la lista en la que este (alta o baja prioridad), se encolará en la de disco y pasará a ejecutarse el siguiente hilo en nuestra política, a no ser que los datos requeridos ya estén en la caché, en este caso no pasará a la espera.

La interrupción de disco simplemente desencolará el hilo de la cola de disco, le encolará a la que le corresponda y dará paso al planificador para que cambie el contexto si es necesario.

# Batería de pruebas

Para conseguir una cobertura completa en las pruebas, enumeraremos a continuación los requisitos de funcionamiento y los mensajes que el programa imprime por pantalla:

Requisitos:

1. Round-Robin:
   1. Cuando termine la rodaja del hilo en ejecución pasar al siguiente.
   2. Si no hay siguiente al acabar la rodaja de un hilo, no cambiar el contexto.
2. Round-Robin para los hilos de prioridad baja y FIFO para los de alta:
   1. Si no hay de alta se ejecutan en Round-Robin.
   2. Cuando uno de alta llega:

Expulsa al actual si es de menor prioridad (pasando al final de los listos de prioridad baja).

Si el actual es de alta va al final de los listos de prioridad alta.

* 1. Los de prioridad alta deben ejecutarse de principio hasta el fin.

1. Round-Robin y FIFO con posibles cambios de contexto voluntarios:
   1. Cuando un hilo ceda la CPU (con la llamada *read\_disk*) es encolado a la lista de espera, si y solo si los datos no están en la cache de páginas.
   2. Si, en efecto, deja la CPU debe ejecutarse el siguiente en la política.
   3. Cuando no haya hilos listos para ejecutar, pero si en espera, se ejecutará la función idle (que no hace computo alguno).

Mensajes:

1. Al cambiar de contexto entre dos threads que aún no han finalizado.

\*\*\* SWAPCONTEXT FROM <origen> TO <destino>

1. Cambio de contexto por finalización de un thread.

\*\*\* THREAD <finalizado> TERMINATED: SETCONTEXT OF <destino>

1. Expulsion de un proceso de prioridad baja para ejecutar un proceso de prioridad alta.

\*\*\* THREAD <expulsado> PREEMTED: SETCONTEXT OF <destino>

1. Al realizar un cambio de contexto voluntario haciendo uso de la funcion read disk (solo si finalmente ocurre el cambio de contexto).

\*\*\* THREAD <current> READ FROM DISK

1. Al mover un thread de la cola de espera a la cola correspondiente.

\*\*\* THREAD <ready> READY

1. Al realizar un cambio de contexto desde el thread idle a un thread listo para ejecutar.

\*\*\* THREAD READY: SET CONTEXT TO <ready>

1. Finalización de un thread.

\*\*\* THREAD <finalizado> FINISHED

1. Finalización del planificador, ya no existen más threads pendientes.

\*\*\* FINISH

Antes de comenzar con las pruebas es necesario comentar lo siguiente: dado que hay tres ficheros de código desarrollado (uno por cada apartado de la práctica) y el *Makefile* solo compila uno de estos, para hacer las pruebas o utilizar cualquier política hay que modificar el archivo *Makefile* para que utilice un archivo u otro; por ejemplo, los requisitos del apartado 1 utilizan el fichero de código “RR.c”.

En la siguiente tabla de pruebas utilizaremos los identificadores de requisitos y mensajes; por ejemplo, el requisito de Round-Robin “Cuando termine la rodaja del hilo en ejecución pasar al siguiente” será **R1a**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador de la prueba | Requisitos y/o mensajes a probar | Descripción de la prueba | Resultados esperados | Resultados obtenidos |
| P1 | R1a, R1b, M1, M2, M7 y M8 | 2 hilos, el 2º dura el doble; R1a y M1 por el intercalamiento que provocan las consumiciones de sus rodajas.  El 2º hilo provoca R1b al no aparecer el M1 (no cambia de contexto) porque el 1º acabará antes, cuando termine el 1º tendremos el M2.  Obviamos el hilo *main* que es el que crea a los demás porque termina muy pronto. | Mensajes de cambios de contexto entre ambos hilos. Cuando el 2º acabe pasará la ejecución al 1º. Y cuando todos terminen acabará el programa. | Los esperados |
| P2 | R2a, R2b, R2c, M3, | 4 hilos, el 1º será de prioridad baja (el *main* que crea los demás), el 2º y el 3º también; cuando llegue el 4º de alta prioridad expulsará al 1º, probando la primera parte de R2b, con el mensaje M3; entonces el 4º acabará sin cambiar el contexto, R2c.  Finalmente, los que quedan tendrán Round-Robin, R2a. | Expulsión del 0 (*main*) por el 4º, cuando termine habrá Round-Robin con los restantes y fin del programa. | Los esperados |
| P3 | R2b | 2 hilos de alta prioridad, el 1º crea al 2º, que quedará después de él en la cola de ejecución de alta prioridad. | El 1º terminará su ejecución y le seguirá el 2º. | Los esperados |
| P4 | R3a, R3b, M5 | 3 hilos, el 1º es de alta prioridad y hace la llamada *read\_disk* al principio de su ejecución (R3a y M4), el 2º y 3º de baja prioridad.  Cuando el 1º espere por la llamada empezarán a ejecutarse en Round-Robin los otros dos, en el momento en que estén los datos disponibles, saltará el mensaje M5 y continuará porque es de prioridad alta, después terminaran los restantes.  Obviamos el hilo *main* que es el que crea a los demás porque termina muy pronto. | El 1º espera por los datos y se bloquea dando paso al Round-Robin de los otros, cuando estén los datos expulsará al que esté, termina y luego acaban los demás. | Los esperados |
| P5 | R3a | Dado que el requisito R3a indica que hay que pasar a la lista de espera solo si los datos no están disponibles, y la simulación de la práctica no permite probar esto de una forma sencilla (porque depende de un numero aleatorio) hemos decidido cambiar la función (solo para esta prueba) que verifica el estado de los datos para provocar que no pase a la lista de espera y probar el buen comportamiento. | El hilo hace la llamada *read\_disk* y no espera porque los datos están disponibles. | Los esperados |
| P6 | R3c y M6 | 1 hilo de alta prioridad que hace la llamada *read\_disk* y pasa a ejecutarse la función *idle*; cuando estén disponibles los datos se para la función idle (M6), continuará el hilo y finaliza el programa. | El hilo hace la espera voluntaria, se ejecuta la función *idle* y cuando estén los datos termina. | Los esperados |

# Conclusión

A la hora de desarrollar la práctica hemos encontrado problemas para entender el código inicialmente. En ocasiones los estudiantes se encuentran con muchas líneas de código que no saben realmente cual es su cometido o funcionalidad. Tuvimos que dedicar un gran esfuerzo inicial para poder conseguirlo. Sin embargo, una vez entendido el funcionamiento básico del primer apartado de Round-Robin, entendimos perfectamente qué había que hacer en los siguientes apartados y cómo había que hacerlo. Quitando esto, la parte más problemática fue identificar los diferentes casos de interacción entre hilos que podían darse. Esto es, manejar los estados y sus prioridades.

Como aspecto positivo recalcar que nos ha servido para entender como funcionaría un planificador de más bajo nivel y pienso que hemos adquirido los conocimientos suficientes para poder implementar (en este contexto simplificado y quizás en otros más complejos…) otras muchas más políticas de planificación.