Diseño de Sistemas Operativos

Memoria Práctica 1

Marcelino Tena Blanco
NIA: 100383331
Daniel Romero Ureña
NIA: 100383331
Gonzalo Fernández García
NIA: 100383212

Índice

Introducción

El siguiente documento consta de 3 grandes apartados, en las cuales se explicará de forma detallada el diseño realizado, la explicación del código implementado y la batería de pruebas realizada para verificar el correcto funcionamiento del programa, de cada uno de los tres apartados solicitados en la práctica, los cuales son:

- **Round-Robin:** el planificador de hilos se basa en Round-Robin, haciendo que cada hilo sea ejecutado en rodajas de tiempo definidas mediante la variable QUANTUM_TICKS. La ejecución del hilo solo se detendrá cuando el proceso acabe su ejecución o cuando finalice su rodaja de tiempo.
- **Round-Robin/SJF con prioridades:** en este caso se intercalan dos tipos de planificación, haciéndose uso de una planificación Round-Robin para hilos de baja prioridad y SJF (primero el trabajo más corto) para los hilos de prioridad alta. En este caso se añade el hecho de que la ejecución de un hilo con prioridad baja pueda ser expulsado por un hilo de prioridad alta.
- **Round-Robin/SJF con posibles cambios de contexto voluntarios:** este apartado es una ampliación del segundo apartado, añadiendo cambios de contexto debido a la función read_disk, lo que provocará el bloqueo del proceso, añadiéndose este a una cola de bloqueados.

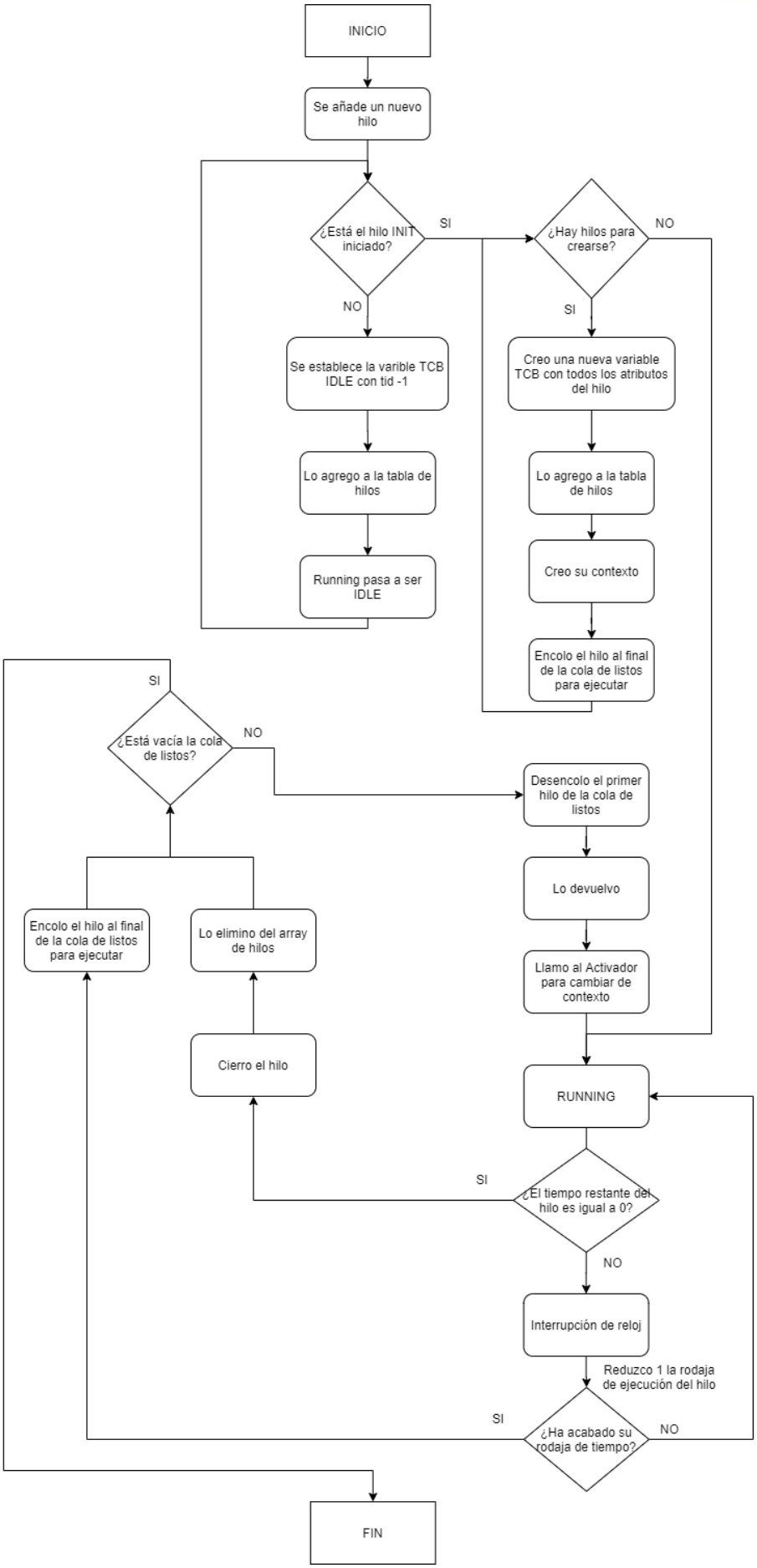
Round-Robin

Descripción del código

El programa comienza con el añadido de un nuevo hilo mediante el método mythread_create, método en el cual se comprueba en primer lugar si el hilo init está iniciado, si este no está iniciado, se procede a iniciarse y agregarse al comienzo del array de hilos, pasando este a estar en ejecución. A continuación, se crea una variable TCB con todos los parámetros del hilo añadido para posteriormente añadirse al array de hilos. Una vez añadido, se creará un contexto para dicho hilo y se encolará al final de la cola de listos.

Una vez no haya más hilos que añadir, comenzará a ejecutar el primer hilo, añadiendo ticks al reloj y realizando una cuenta atrás en la rodaja de tiempo definida. Si se da el caso de que el hilo completa su ejecución dentro del tiempo de la rodaja, dicho hilo se eliminará del array de hilos dando paso al siguiente hilo en el array si lo hubiese. En el caso de que el hilo no completase su ejecución dentro de su rodaja de tiempo, se deberá comprobar si hay más procesos a parte de él en el array de hilos, en caso afirmativo, este hilo es encolado al final de dicho array, se cambia su contexto con el del siguiente hilo, y pasa a ejecutarse el nuevo este último. En caso de no haber más hilos, el hilo actual irá recibiendo rodajas de tiempo seguidamente hasta que finalice su ejecución o se añada un nuevo hilo al array de hilos.

Este modelo de ejecución se seguirá de forma recursiva hasta finalizar la ejecución del último hilo, presente en el array de hilos, terminando de esta manera la ejecución de todos los hilos previamente creados.


INICIO
Se añade un nuevo hilo
¿Está el hilo INIT iniciado?
SI
¿Hay hilos para crearse?
NO
NO
SI
Se establece la varible TCB IDLE con tid -1
Creo una nueva variable TCB con todos los atributos del hilo
Lo agrego a la tabla de hilos
Lo agrego a la tabla de hilos
Running pasa a ser IDLE
Creo su contexto
Encolo el hilo al final de la cola de listos para ejecutar
SI
¿Está vacía la cola de listos?
NO
Desencolo el primer hilo de la cola de listos
Lo devuelvo
Encolo el hilo al final de la cola de listos para ejecutar
Lo elimino del array de hilos
Llamo al Activador para cambiar de contexto
Cierro el hilo
RUNNING
SI
¿El tiempo restante del hilo es igual a 0?
NO
Interrupción de reloj
Reduzco 1 la rodaja de ejecución del hilo
SI
¿Ha acabado su rodaja de tiempo?
NO
FIN

En lo referido a decisiones de diseño implementadas, se ha decidido establecer una nueva cola para almacenar los diferentes procesos creados que se encuentran listos para comenzar su ejecución, la cual nos permitirá gestionar el orden de ejecución de los diferentes hilos organizándolos por el momento de su creación, desencolando dichos hilos cuando les llegue su turno de ejecución. Esta cola se generará al crear el primer hilo, momento en el cual la cola quedará activa e irá encolando los futuros hilos que se vayan creando. Se ha establecido el desencolado del siguiente hilo dentro del planificador (scheduler), donde se obtendrá el siguiente hilo una vez terminada la rodaja de tiempo termine, desencolando de esta forma dicho hilo, de esta forma se optimizará el tiempo de ejecución ya que obtenemos el nuevo hilo a la vez que lo desplazamos fuera de la cola de listos para su inminente ejecución.
Se ha optado por hacer que el hilo idle no termine su ejecución ya que no puede encolarse, consideramos que, de este modo, en caso de presentarse situaciones donde no haya hilos para ejecutar, el programa queda en espera a la creación de nuevos hilos el tiempo que fuese necesario mediante la ejecución del hilo idle.

Batería de pruebas

A continuación, se expone el conjunto de batería de pruebas expuestas para la verificación del correcto funcionamiento del código:

Prueba	Resultado esperado	Resultado obtenido
Uso de hilos con una duración inferior a la rodaja del Round-Robin.	El planificador es capaz de añadir varios hilos en la misma rodaja de tiempo, encajando el tiempo de ejecución de estos dentro de su rodaja de tiempo.	El planificador ejecuta todos los hilos dentro de la misma rodaja de tiempo, finalizando el programa en una única rodaja de tiempo.
Uso de hilos con una duración mayor a la rodaja Round-Robin.	El planificador expulsa a los diferentes hilos tras consumir su rodaja de tiempo, dando paso al siguiente en la cola y devolviendo a la cola de listos al proceso ejecutándose.	El planificador gestiona correctamente la entrada y la salida de los diferentes hilos, una vez finalizan su ejecución dentro de la rodaja de tiempo, acabando la ejecución del hilo de mayor duración al final.
Uso de hilos cuyo tiempo de ejecución es igual a la rodaja de Round-Robin.	El número de rodajas de tiempo al final de la ejecución debe ser igual al número total de procesos creados.	El programa tarda en completar su ejecución una cantidad de rodajas de tiempo cuyo número es igual al número de procesos creados.
Uso de varios hilos cuya suma de tiempo de ejecución sea igual a una rodaja de tiempo de Round-Robin.	Únicamente debe haberse realizado una sola rodaja de tiempo al final de la ejecución.	El planificador es capaz de ejecutar todos los hilos creados dentro de una misma rodaja de tiempo.

Uso de un hilo cuyo tiempo de ejecución sea múltiplo del tiempo de la rodaja de tiempo del Round-Robin, en este caso ser 3 veces superior.	El número de rodajas de tiempo usadas por un determinado hilo debe ser igual a 3.	El número total de rodajas de la ejecución es 3, verificando que se respeta la proporción del tiempo de ejecución con respecto al tiempo de rodaja.
Uso de un único hilo con un tiempo superior a la rodaja de tiempo del Round-Robin.	Al ser el único hilo de la lista, este deberá ejecutarse sin ninguna expulsión hasta la finalización de su ejecución.	Se verifica que al ser el único hilo no hay ningún cambio de contexto por parte del hilo, ejecutándose en todas las rodajas necesarias de forma continua hasta completar su ejecución.
Usar hilos con tiempos de ejecución negativos.	Se notifica un error debido al uso de un tiempo de ejecución no válido.	Comprobamos que al introducir algún hilo con tiempos de ejecución negativos, el mensaje imprime un mensaje de error.

Round-Robin/SJF con prioridades

Descripción del código

El funcionamiento general del código consiste en una variación del apartado anterior, en este caso contaremos con dos colas de listos, de las cuales una almacenará los hilos listos de alta prioridad y otro almacenará los de baja prioridad. En lo referido a las colas de alta prioridad, los hilos serán organizados en función de su tiempo de ejecución, siendo el primero el que posea un menor tiempo, respecto a las colas de baja prioridad, los hilos se ordenarán según su orden de creación.

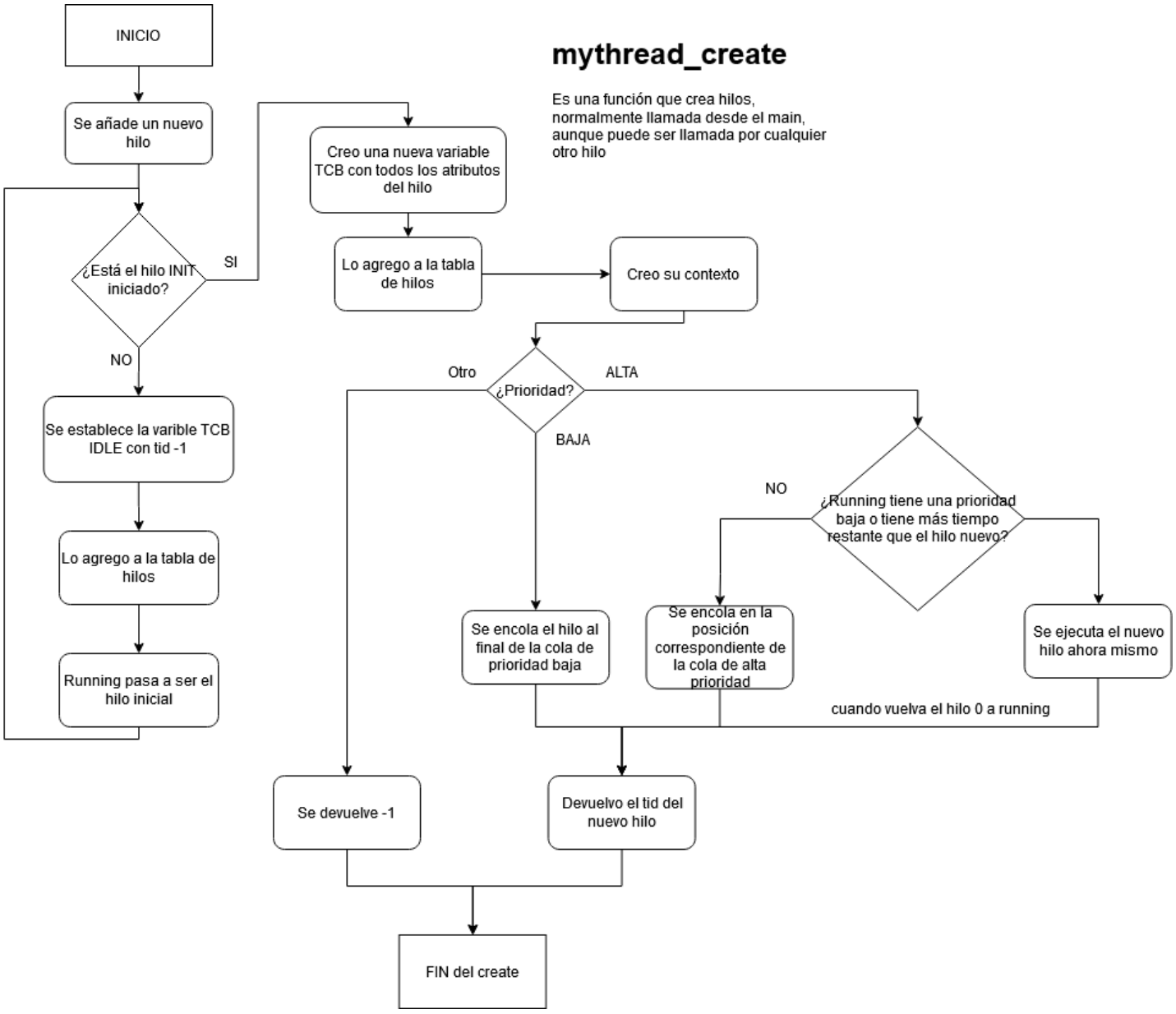
En cuanto a la creación de hilos, hemos implementado la capacidad de que se detecte la creación de nuevos hilos de alta prioridad una vez son creados, todo ello para poder contrastar si el hilo recién creado debe expulsar al hilo en ejecución.

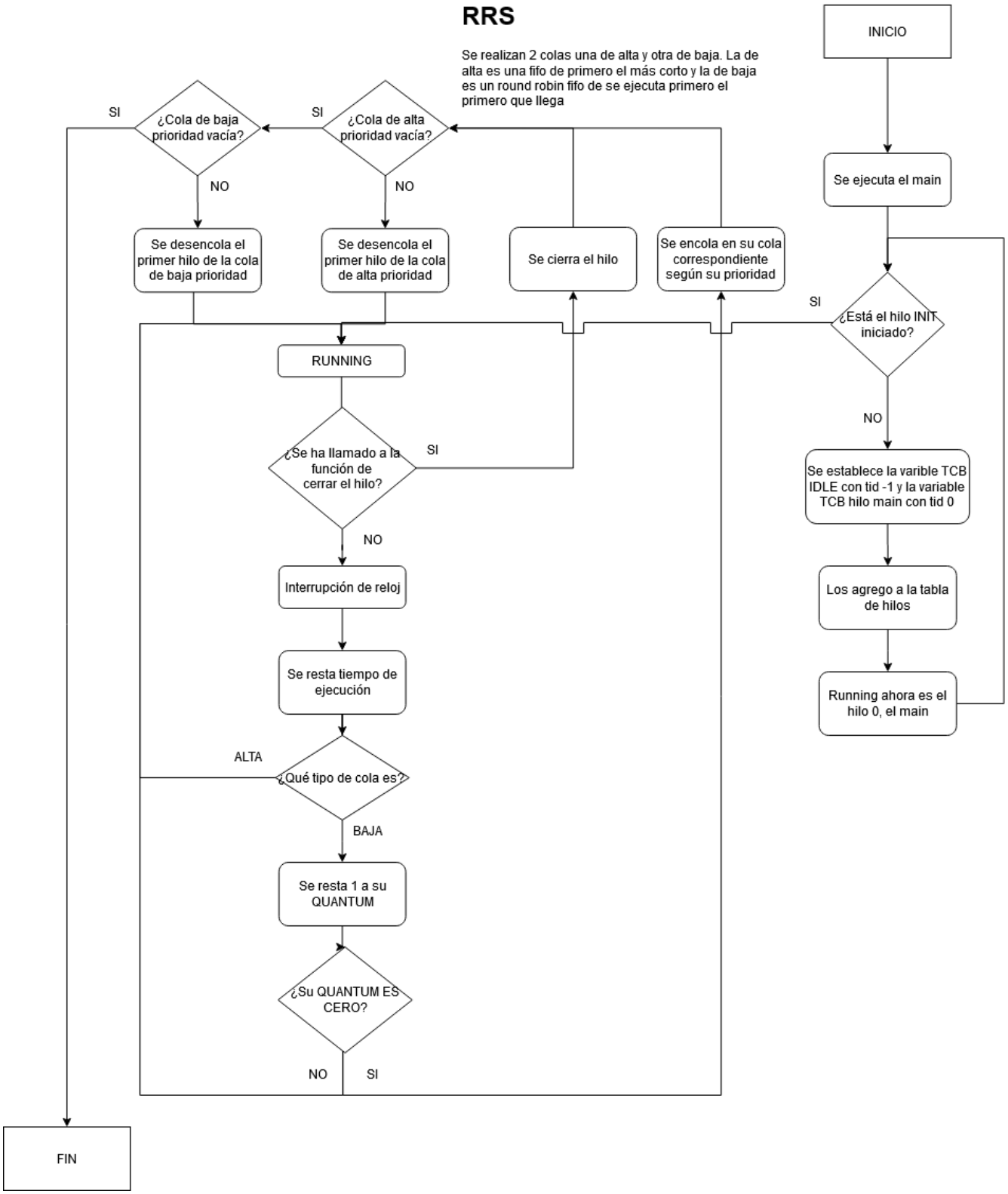
Una vez se crean los hilos y son encolados en su correspondiente cola, se pasa a comprobar si la cola de hilos de alta prioridad está vacía pudiendo darse dos casos:

- **Cola de prioridad alta vacía:** En este caso, se pasaría a desencolar y ejecutar el primer hilo de la cola de prioridad baja si lo hubiese, teniendo como planificador un sistema Round-Robin, implementado del mismo modo que en el apartado anterior. Con el añadido de poder ser expulsado por un proceso de alta prioridad si lo hubiese.
- **Cola de prioridad alta no vacía:** En este caso, comenzaría el desencolado y ejecución del primer proceso de la cola de alta prioridad, el cual utilizaría un sistema SJF manteniendo su ejecución hasta su finalización si no se crease un hilo de prioridad alta cuyos remaining_ticks fueran inferiores a los suyos.

En ambos casos, se podrá parar la ejecución de la ejecución de ambos tipos de hilo con la creación de un nuevo hilo de prioridad alta, ya que cada vez que se crea uno de estos hilos, se procederá con la comparación del hilo actualmente en ejecución, llegando a expulsar a este si se dan las circunstancias por la preferencia en prioridad o por los remaining_ticks de dos hilos de prioridad alta.

Este funcionamiento se desarrolla de forma recursiva hasta terminar la ejecución de todos los hilos creados, pudiendo realizarse expulsiones involuntarias durante este proceso por parte de hilos de prioridad alta a hilos de prioridad baja o hilos de alta prioridad a otros hilos del mismo tipo de prioridad, pero de mayor tiempo de ejecución.





Este funcionamiento se desarrolla de forma recursiva hasta terminar la ejecución de todos los hilos creados, pudiendo realizarse expulsiones involuntarias durante este proceso por parte de hilos de prioridad alta a hilos de prioridad baja o hilos de alta prioridad a otros hilos del mismo tipo de prioridad, pero de mayor tiempo de ejecución.

En lo referido al diseño implementado, como se ha comentado anteriormente, se ha establecido una funcionalidad para comprobar la creación de un hilo de alta prioridad, para poder realizar la expulsión de un hilo de baja o alta prioridad si fuese necesario. El programa tendrá en cuenta la posibilidad de la generación de nuevos hilos de forma constante, de modo que, si se crea un nuevo hilo durante el proceso de ejecución, se tomen las medidas necesarias al instante y se realicen las expulsiones necesarias.
En lo referido a las colas, hemos hecho uso de una para los hilos de prioridad alta y otra para los de prioridad baja, de esta forma, en la cola de prioridad baja los hilos serán ordenados mediante su orden de creación, mientras en la cola de prioridad alta, se ordenarán de menor a mayor tiempo de ejecución (remaining_ticks).

Batería de pruebas

Debido al correcto funcionamiento del primer apartado, del cual se reutiliza código para el desarrollo de la parte actual, obtenemos los mismos resultados de las pruebas anteriormente realizadas, por lo que esta nueva batería de pruebas, se enfocará únicamente en la verificación del funcionamiento de las nuevas funcionalidades añadidas.

Prueba	Resultado esperado	Resultado obtenido
Uso únicamente de varios hilos de prioridad alta.	El orden de ejecución corresponde a la duración de los diferentes, realizando el orden de ejecución de los mismos en función de la duración de su ejecución.	Los hilos se organizan correctamente en la cola de alta prioridad, realizando su ejecución de menor tiempo de ejecución a mayor tiempo de ejecución.
Forzar la entrada de un hilo de alta prioridad cuando un hilo de baja prioridad se está ejecutando.	El hilo de prioridad baja es expulsado y comienza la ejecución del hilo de prioridad alta.	Verificamos que se realiza un swapcontext, expulsando al hilo de baja prioridad en ejecución por el hilo de alta prioridad recién creado, el cual comienza a ejecutarse.
No realizar la creación de ningún hilo.	Se ejecuta el thread del main y tras esto se finaliza el programa por ausencia de hilos.	Verificamos que el programa únicamente ejecuta el hilo main para seguidamente terminar la ejecución del programa.
Uso de hilos de alta y baja prioridad.	El planificador deberá realizar la ejecución de todos los procesos de alta prioridad para posteriormente ejecutar todos los hilos de baja prioridad	El programa gestiona correctamente las expulsiones y la colocación de todos los hilos generados, así como de sus correspondientes colas.
Forzar la entrada de un hilo de baja prioridad durante la ejecución de hilos de baja prioridad.	El hilo creado no expulsa al hilo de ejecución y este es añadido al final de la cola de baja prioridad .	Verificamos que el hilo no expulsa al hilo en ejecución, ordenándose correctamente dentro de la cola y en el orden de ejecución por rodajas de tiempo.

Forzar la entrada de un hilo de baja prioridad cuando un hilo de alta prioridad está en ejecución.	El hilo de alta prioridad debe seguir en ejecución hasta su finalización.	El hilo creado no expulsa al hilo de alta prioridad en ejecución, se encola correctamente al final de la cola de baja prioridad.
Forzar la entrada de un hilo de prioridad alta de menor duración durante la ejecución de un proceso de prioridad alta.	El proceso entrante deberá comenzar a ejecutarse expulsando al proceso actual en ejecución.	El hilo creado tras ser creado es capaz de expulsar al proceso en ejecución, pasando este a estar en espera dentro de la cola.
Uso de prioridades no válidas en los hilos.	Mensaje de error indicando una prioridad no válida y finaliza el programa.	Mensaje de error indicando una prioridad no válida y finaliza el programa.
Forzar la entrada de un hilo de prioridad alta de mayor duración durante la ejecución de un proceso de prioridad alta.	El proceso entrante no expulsará al proceso en ejecución.	Comprobamos que el hilo entrante no expulsa al hilo en ejecución, sino que se coloca en su posición de la cola de alta prioridad a la espera de ser ejecutado.

Round-Robin/SJF con posibles cambios de contexto voluntarios

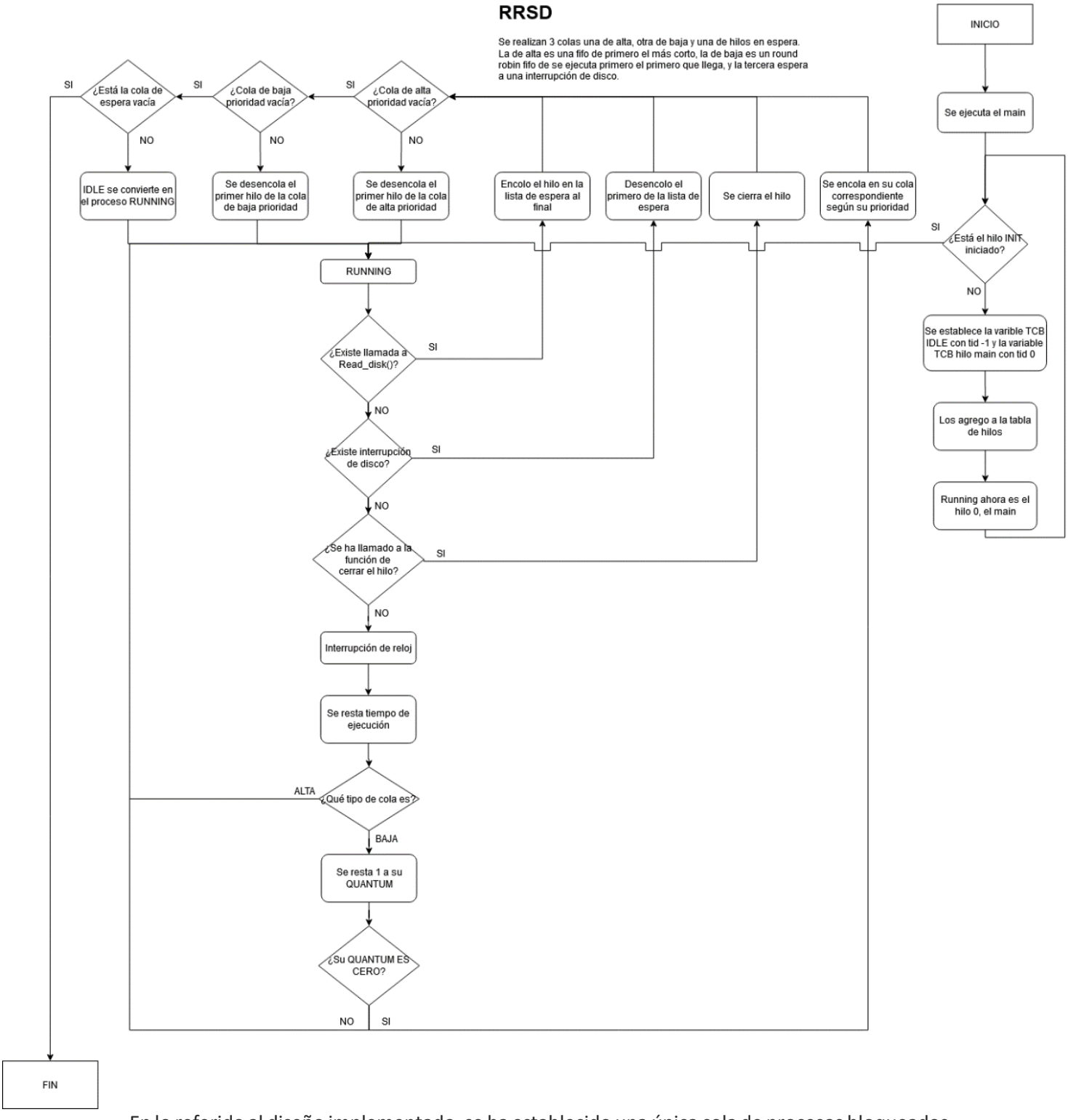
Descripción del código

El funcionamiento de este apartado deriva del apartado anterior, añadiendo la posible lectura de disco, la cual bloqueará los hilos que hagan uso de la llamada read_disk (), quedando estos encolados en una lista de procesos bloqueados. Dentro de esta lista, se seguirán aplicando las preferencias de prioridades ya implementadas, teniendo preferencia para salir de la cola de bloqueados los hilos que primero hayan solicitado la lectura del disco, independientemente de su prioridad. Una vez se produzca el desbloqueo de los hilos bloqueados, se mantendrán las preferencias de prioridad para la ejecución de los hilos disponibles previamente implementadas, pudiendo producirse expulsiones involuntarias en la ejecución tras el desbloqueo de algunos hilos.

Una vez se realice la lectura de disco por parte de los distintos procesos en bloqueados, se producirá una interrupción de disco la cual permitirá el desbloqueo de dicho hilo, la cual ocasionará la salida de dicho hilo de cola de bloqueados pasando su estado de waiting a init.

Mientras se den situaciones donde la totalidad de los hilos se encuentren bloqueados, el hilo IDLE se ejecutará hasta que se solvente esta situación de bloqueo y alguno de los hilos bloqueados deje de estarlo, caso en el cual la ejecución pasará a dicho hilo.

Nota: El programa RRSD hace uso del mimo método thread_create que el programa RRS



En lo referido al diseño implementado, se ha establecido una única cola de procesos bloqueados, donde como ya se ha comentado, se almacenarán todos los procesos que soliciten la lectura disco. Se ha optado por no implementar la preferencia por prioridad dentro de la cola y ordenar según el orden de solicitud de lectura de disco, ya que esto podría provocar el bloqueo permanente de algunos hilos de baja prioridad en la cola de bloqueados, siendo retenidos por hilos de alta prioridad que solicitasen la lectura a disco.

Batería de pruebas

Debido a ser una extensión de las funcionalidades previamente implementadas en los dos anteriores apartados, las pruebas realizadas anteriormente no son incluidas en esta batería de pruebas, ya que han sido correctamente verificadas anteriormente.

Prueba	Resultado esperado	Resultado obtenido
Forzar la llamada a la función read_disk por parte de un determinado hilo de prioridad alta.	Dicho hilo de prioridad alta deberá ser añadido a la cola de hilos en espera y pasará a ejecutarse el siguiente proceso de la cola de listos correspondiente.	Se verifica que el proceso bloquea su ejecución y es añadido a la lista de bloqueados. Tras producirse este proceso vuelve a ejecutarse, expulsando al hilo de prioridad baja en ejecución.
Forzar la llamada a la función read_disk por parte de un determinado hilo de prioridad baja.	Dicho hilo de prioridad alta deberá ser añadido a la cola de hilos en espera y pasará a ejecutarse el siguiente proceso de la cola de listos correspondiente.	Se comprueba que el hilo se añade a la cola de bloqueados y queda en estado bloqueado hasta la interrupción de reloj, momento en el cual, vuelve a estar listo para ejecutar, pero no comienza su ejecución hasta obtener su turno entre el resto de hilos de prioridad baja y no haber ningún hilo en la cola de alta prioridad.
Ejecución del programa con todos los hilos existentes bloqueados.	Se ejecuta el hilo IDLE hasta que se desbloque la situación.	Verificamos que el IDLE se encarga de la ejecución mientras los hilos creados están bloqueados.
Forzar una situación en la que en el momento en el que un hilo de alta prioridad se cree, un hilo de prioridad baja en ejecución lea el disco.	El hilo de alta prioridad interrumpe la ejecución del proceso de baja prioridad, accediendo este a la lectura del disco tras la ejecución del proceso de alta prioridad.	Comprobamos que el planificador obtiene el resultado esperado, aplazando la lectura de disco del proceso de baja prioridad hasta que el proceso de alta prioridad finalice su ejecución.
Forzar la situación en la que un hilo de alta prioridad más corto que otro mientras este en ejecución, lea el disco, pasando el hilo más largo a estar en ejecución y el más corto a bloqueado.	Una vez el hilo bloqueado vuelva a estar listo, expulsará al hilo más largo y pasará a estar en ejecución.	Podemos verificar que una vez el hilo más corto termina su bloqueo, expulsa correctamente al hilo más largo, pasando a estar en ejecución de nuevo.

Conclusiones sobre la práctica

En términos generales, consideramos que esta práctica en términos generales ha sido muy constructiva para asentar los conocimientos sobre planificación de procesos y gestión de interrupciones, ya que en los tres bloques de la práctica se hacen uso de dichos conceptos de manera incremental, aplicando la implementación de estos de una forma práctica y comprobando su funcionamiento mediante pruebas donde podemos observar el comportamiento de una planificación y la gestión de interrupciones durante el tiempo de ejecución.

En cuanto a los problemas encontrados durante la práctica, principalmente hemos tenido problemas al comienzo de la misma, comprendiendo el funcionamiento del código proporcionado, así como la implementación de la estructura principal de nuestro diseño inicial, el cual es la base para los dos apartados siguientes. Además de esto, se nos han presentado diferentes dificultades a la hora de realizar las pruebas del tercer bloque, en el cual hemos obtenido algunos fallos en el desarrollo de las interrupciones que hemos tenido que solventar.