En esta tarea Ud. deberá implementar una nueva herramienta de sincronización de nThreads para organizar eficientemente los accesos a disco. Debe garantizar la exclusión mutua en el acceso a disco e implementar la estrategia C-SCAN para reducir el movimiento del cabezal del disco. Esta estrategia atiende los accesos pendientes barriendo repetidamente con el cabezal del disco desde las pistas internas hacia las pistas externas. Cuidado porque la tarea 5 será una continuación de esta tarea. Las funciones que se requiere programar son:

void nRequestDisk(int track, int delay);	Solicita acceso al disco indicando la pista. El parámetro *delay* será siempre -1 en esta tarea.
void nReleaseDisk( );	Notificación de término de uso del disco
void iniDisk( );	Función de inicialización
void cleanDisk( );	Función de limpieza

El siguiente es un ejemplo de uso de estas funciones:

```
nRequestDisk(100, -1);
... se accede a la pista 100 del disco ...
nReleaseDisk( );
```

Un thread invoca *nRequestDisk* para solicitar acceso exclusivo al disco. Debe esperar si el disco está ocupado. La identificación numérica de pista, *track*, es linealmente proporcional a la distancia de la pista al centro del disco. La pista 0 es la más cercana al centro. El significado del parámetro *delay* se explicará en la tarea 5. El thread accederá a la pista *track* después del retorno de *nRequestDisk*. Luego invocará *nReleaseDisk* notificando el término del uso del disco. Si en ese momento hay varias solicitudes de acceso en espera, y se acaba de acceder a la pista *t*, entre todos los *nRequestDisk(t')* pendientes Ud. debe autorizar el acceso que requiera el cabezal en la pista *t'* más cercana a *t* sujeto a que $t' \geq t$. Autorice haciendo que ese *nRequestDisk(t')* retorne. Si no hay ninguna solicitud con esas características, autorice la solicitud que lleve el cabezal a la pista más cercana al centro del disco. Por ejemplo si el cabezal está en la pista 4 y hay solicitudes en espera para las pistas 2, 2, 3, 4, 6 y 10, el orden de autorización debe ser 4, 6, 10, 2, 2 y 3 (si no se hicieron otras solicitudes).

Requerimientos

Ud. debe programar las funciones pedidas en el archivo *disk.c* como herramientas de sincronización nativas de nThreads, es decir usando operaciones como *START_CRITICAL*, setReady, *suspend*, *schedule*, etc. Ud. no puede implementar la API solicitada en términos de otras herramientas de sincronización pre-existentes en nThreads (como semáforos, mutex o condiciones).

Ayuda: Use 2 instancias de la cola de prioridades que está programada en *pss.h* y *pss.c* con operaciones *makePriQueue*, *priBest*, *priPeek*, *priGet*, *priPut*, *emptyPriQueue* y *destroyPriQueue*. En *nRequestDisk*, si el disco está desocupado, retorne de inmediato para atender la solicitud. Siempre guarde en una variable global la última pista *U* atendida. Si el disco está ocupado suspenda el thread. Además si $track \geq U$, agregue la solicitud a la primera cola con prioridad *track* y si no a la segunda cola de prioridad. En *nReleaseDisk*, si la primera cola no está vacía, extraiga el primer thread y despiértelo. Si no, intercambie la primera con la segunda cola y repita. Si las 2 colas están vacías solo retorne. Recuerde actualizar *U*. Puede serle útil el campo *ptr* de tipo *void* * declarado en el descriptor de thread para almacenar ahí cualquier dirección que necesite.

Ejemplos de la solución que se espera de Ud. son la implementación de los semáforos, mutex, condiciones y mensajes de nThreads (en los archivos nKernel/sem.c, nKernel/mutex-cond.c y nKernel/nmsgs.c).

Instrucciones

Baje *t4.zip* de U-cursos y descomprímalo. El directorio *T4* contiene los archivos *test-disk.c*, *Makefile*, *disk.h* (con los encabezados de las funciones requeridas) y otros archivos. Ejecute el comando make sin parámetros en el directorio T4 para recibir instrucciones adicionales sobre cómo compilar y probar su solución, los requisitos que debe cumplir para aprobar la tarea y cómo entregar su tarea por U-cursos. Además se explica cómo puede probar sus tareas 1, 2 y 3 usando nThreads como implementación de pthreads.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *disk.zip* generado por *make zip*. Recuerde descargar el archivo que subió, descargar nuevamente los archivos adjuntos y volver a probar la tarea tal cual como la subió a U-cursos. Solo así estará seguro de no haber entregado archivos incorrectos. Se descuenta medio punto por día de atraso. No se consideran los días de receso, sábados, domingos o festivos.