Esta tarea es una continuación de la tarea 4. Ahora deberá implementar el parámetro timeout para *nEnterWrite*. Por simplicidad **no necesita implementar timeout para nEnterRead**.

Instrucciones

Descargue *t5.zip* de U-cursos y descomprímalo. Copie su archivo *rwlock.c* con la solución de su tarea 4 a este mismo directorio. Modifique *rwlock.c* implementando los timeouts de *nEnterWrite*. Ejecute el comando *make* sin parámetros en el directorio *T5* para recibir instrucciones 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.

Ayuda

Como en la tarea 4, use 2 colas de tipo NthQueue: una para los nthreads lectores y la otra para los nthreads escritores.

Necesita distinguir entre nthreads que invocaron *nEnterWrite* con timeout (timeout>0) y las que lo hicieron sin timeout (timeout<0). Para los primeros use el estado WAIT_RWLOCK_TIMEOUT. Para los segundos use el estado WAIT_RWLOCK. Es importante porque cuando se invoca *nExitWrite* o nExitRead, un nthread en espera en un *nEnterWrite* con timeout está configurado para despertarse con *nth_programTimer*. Si su estado continúa en WAIT_RWLOCK_TIMEOUT su timeout debe cancelarse con *nth_cancelThread* (vea la clase auxiliar de mensajes con timeout).

Observe que *nth_programTimer* recibe el timeout en nanosegundos, mientras que *nEnterWrite* recibe el timeout en milisegundos. Para convertir timeout en milisegundos a nanosegundos, use la expresión timeout*1000000**LL** para que el resultado sea de tipo long long, de otro modo el tipo sería int y habría desborde. ¡Use el sufico **LL**!

Cuando el timeout expira para un nthread que invocó *nEnterWrite*, ese nthread todavía está en la NthQueue de los escritores en espera. Tendrá que eliminarlo de esa cola. Para lograrlo, siga cuidadosamente las siguientes instrucciones. Necesitará leer varias veces. Un error le costará mucho tiempo en depuración. Cuando un nthread escritor necesite esperar con timeout, coloque la dirección de la NthQueue de escritores en el campo *ptr* del descriptor de ese nthread. Al programar el timeout con *nth_programTimer*, suministre en el segundo parámetro una función *f* que hará la eliminación. Esa función f recibe como parámetro el descriptor del nthread cuyo timeout expiró. En el campo *ptr* estará la NthQueue en donde se encuentra en espera. Elimine el nthread de esa cola con la función *nth_delQueue* y deje el campo *ptr* en NULL. Si después de llamar a *schedule*, el campo *ptr* es NULL, quiere decir que el timeout expiró y por lo tanto *nEnterWrite* debe retornar 0. Si no es NULL, *nEnterWrite* debe retornar 1 porque sí obtuvo el recurso compartido.

Tenga cuidado porque la función *f* se invoca dentro de una rutina de atención de señales. Haga cosas simples en esa función.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *rwlock.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, sábado, domingo o festivos.