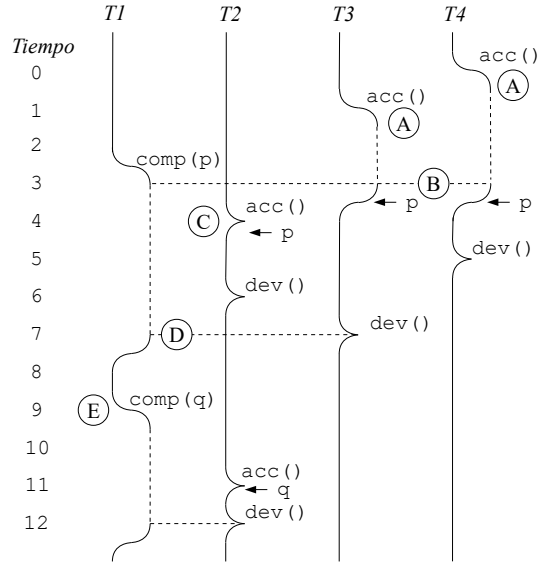
Programe de manera nativa en nThreads las siguientes funciones cuyo fin es permitir que varios threads compartan datos en modo lectura:

- *void nCompartir(void *ptr)*: Ofrece compartir los datos apuntados por *ptr* con los threads (nano threads) que llamen a *nAcceder*. *nCompartir* queda en espera hasta que los threads notifiquen que desocuparon los datos llamando a *nDevolver*.
- *void *nAcceder(int millis)*: Solicita acceso a los datos ofrecidos con *nCompartir*. Si hay una llamada a *nCompartir* en espera, retorna de inmediato el puntero *ptr* suministrado mediante *nCompartir*. Si no, espera hasta la próxima invocación de *nCompartir*. En esta tarea ignore el parámetro *millis*. Se usará en la tarea 5.

void nDevolver(void): Notifica que los datos compartidos ya no se usarán.



El diagrama de arriba explica el funcionamiento pedido. En A *nAcceder* se bloquea hasta que otro thread invoque *nCompartir*. Esto ocurre en B, lo que hace que todos los threads que esperaban en una llamada a *nAcceder* se desbloqueen retornando el puntero a los datos (*p* en este caso). La llamada a *nCompartir* queda en espera hasta que todos los threads que llamaron a *nAcceder* notifiquen que no usarán más los datos invocando *nDevolver*. En C, como hay una llamada a *nCompartir* en espera, *nAcceder* retorna de inmediato los datos compartidos. En D se invoca el último *nDevolver*, y por lo tanto *nCompartir* retorna. Si se invoca *nCompartir* y no hay threads que llamaron a *nAcceder*, *nCompartir* espera hasta que algún thread invoque *nAcceder* (ver E).

Ud. debe programar las funciones *nCompartir*, *nAcceder* y *nDevolver* como herramientas de sincronización nativas de nThreads, es decir usando operaciones como *START_CRITICAL*, *setReady*, *suspend*, *schedule*, *nth_putBack*, etc. Inicialice las variables globales que necesite en la función *nth_compartirInit*. Ud. no puede implementar la API solicitada en términos de otras herramientas de sincronización pre-existentes en nThreads (como semáforos, mutex, condiciones o mensajes). 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/nsem.c*, *nKernel/mutex-cond.c* y *nKernel/nmsgs.c*).

Instrucciones

Descargue *t4.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T4* para recibir instrucciones acerca del archivo en donde debe programar su solución (*T4/nKernel/ncompartir.c*), 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 2 y 3 usando nThreads como implementación de pthreads.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *ncompartir.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.