CC4302 Sistemas Operativos - Tareas 4 y 5 – Semestre Otoño 2025
Profs.: Mateu, Torrealba, Arenas

En las siguientes 2 tareas Ud. debe implementar nuevas herramientas utilizando nThreads para resolver la implementación de un sistema de subastas multi-threaded.

A continuación los encabezados de las funciones que debe implementar

```
enum { FALSE= 0, TRUE= 1 };
typedef struct subasta *nSubasta;

nSubasta *nNuevaSubasta(int n);
void nDestruirSubasta(nSubasta s);
double nAdjudicar( nSubasta s, int *punid   );
int nOfrecer(nSubasta s, double oferta,
             int timeout);
```

La función **nNuevaSubasta** inicializa la subasta por un producto del cual se subastarán *n* items. Esta función es invocada por el propietario de la subasta.

La función **nDestruirSubasta** se utiliza para liberar los recursos solicitados para realizar la subasta.

La subasta se da por cerrada cuando el propietario decide cerrarla invocando **nAdjudicar.**

Cuando el propietario cierra la subasta mediante la invocación de la función **nAdjudicar**, la cual retorna el monto total recaudado y en la variable *punid* las unidades que no fueron vendidas ya que llegaron menos oferentes que los n items ofrecidos.

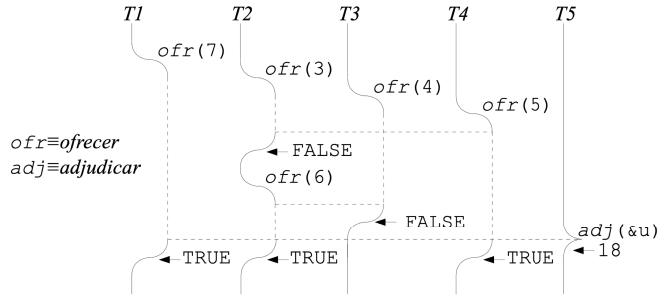
Múltiples oferentes interesados en la compra del producto, realizan una *oferta* en dinero por un item del producto mediante el llamado de la función **nOfrecer** y esperan una respuesta por tiempo *timeout.*

La función **nOfrecer** queda a la espera hasta que:

i) La subasta se cierra, retornando *TRUE* cuando se adjudica un producto y *FALSE* cuando se presentaron *n* oferentes con un precio mayor a la oferta realizada.

ii) Si se produce el timeout el *nthread* retorna FALSE.

El siguiente diagrama muestra el funcionamiento con n=3



Observe que al momento que T4 oferta un precio de 5, existen 4 oferentes a la espera, siendo la oferta de T2(3) la peor por lo que pierde y **nOferta** retorna *FALSE.* T2 realiza una nueva oferta lo que produce que a T3(4) se le retorna *FALSE.*

Cuando T5 invoca **nAdjudicar** se le retorna 18 y u=0, pues no quedan item sin vender. Los ganadores adjudicados fueron T1(7), T2(6) y T4(5).

Requerimientos:

- Ud. debe programar la funciones pedidas y la estructura del tipo *subasta* en el archivo *subasta.c, utilizando las funciones nativas de nThreads,* es decir utilizando operaciones como *START_CRITICAL, setReady, suspend, schedule, etc.*

- *Ud. no puede utilizar las herramientas de sincronización pre-existentes a nThreads* (semáforos, mutex, condiciones o mensajes).

TIP:

- Ud puede encontrar ejemplos de sincronización en nThreads en el directorio nKernel, en los archivos *sem.c, mutex-cond.c y nmsgs.c.*

- El siguiente mensaje de advertencia no es un problema, ==4952==WARNING: ASan doesn't fully support makecontext/swapcontext functions and may produce false positives in some cases!

T4: Implemente las funciones requeridas considerando el parametro *timeout = -1*, es decir tiempo infinito.

T5: Implemente las funciones requeridas considerando que existe un timeout y el *nthread* puede realizar una nueva oferta.

Instrucciones

Descarque t4.zip de u-cursos y descomprimirlo. Ejecute el comando *make* sin parámetros en el directorio T4 para recibir instrucciones acerca del archivo en donde debe programar su solución (T4/subasta.c), como compilar y probar su solución, los requisitos que debe cumplir para aprobar la tarea y cómo entregar su tarea por u-cursos.

Entrega

Ud. solo debe entregar por medio de u-cursos el archivo subasta.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.