CC3301 Programación de Software de Sistemas – Tarea 6
Semestre Primavera 2020 – Prof.: Luis Mateu

Un conjunto de threads comparte un recurso único. El recurso no puede ser usado simultáneamente por múltiples threads. Para evitar el uso simultáneo, antes de comenzar a utilizar el recurso un thread solicita el recurso invocando la función *ocupar( )* y concluido su uso, invoca la función *desocupar( )*. La actual implementación de *ocupar* y *desocupar* emplea un simple *mutex* para evitar el uso simultáneo, pero no respeta el orden de llegada a la hora de otorgar el recurso.

Reprograme las funciones ocupar y desocupar del archivo *t6.c*, de modo que se garantice la exclusión mutua al ocupar el recurso y además se otorgue por orden de llegada.

Metodología obligatoria

Use un protocolo similar al que usan farmacias, isapres, bancos, etc. Al llegar, un cliente toma un número. Un visor anuncia el número del cliente que se está atendiendo. Cuando se termina de atender a ese cliente, el visor se incrementa en 1 para que pase el siguiente cliente. Un cliente sabe que es su turno de atención porque el número del visor coincide con su propio número. Aplicado a esta tarea, significa que en la función *ocupar*, un thread sabe que se le otorgó el recurso cuando su número de cliente coincide con el número en el visor. En la función *desocupar* debe incrementar el visor para que el recurso sea ocupado por otro thread: el que lleva más tiempo esperando.

Instrucciones

Baje *t6.zip* de U-cursos y descomprímalo. El directorio *T6* contiene los archivos *test-t6.c*, *Makefile, t6.h* (con los encabezados requeridos) y *t6.c* con la implementación actual que no respeta el orden de llegada. Ejecute la implementación actual en un computador con al menos 2 cores con el comando *make test-g*. No pasará el test. Ud. debe modificar el archivo *t6.c* reprogramando ahí las funciones *ocupar* y *desocupar*. Necesitará definir nuevas variables globales.

Pruebe su tarea bajo Debian 10 (con soporte para programas de 32 y 64 bits) con los comandos:

- *make test-g*
- *make test-O*
- *make test-O-m32*
- *make test-sanitize*

Copie la salida de todos estos comandos y péguela en el archivo *resultados.txt*. En la última prueba gcc usa la opción *-fsanitize=thread* y *valgrind* para generar un ejecutable que reporta potenciales *dataraces*. Su tarea será rechazada si esta prueba reporta *dataraces*.

Entrega

Ud. solo debe entregar los archivos *t6.c* y *resultados.txt* en el formato *.zip* por medio de U-cursos. Se descontará medio punto por día de atraso. No se consideran los días de vacaciones, sábado, domingo o festivos.