Sistemas Operativos

Obligatorio 2

Problema del Productor-Consumidor

Considere un proceso que produce información denominado *productor* y un proceso que consume información denominado *consumidor*, siendo diferentes las velocidades de producción y consumo de la información. Dicho desajuste en las velocidades hace necesario que se establezca una sincronización entre los procesos de manera que la información no se pierda, ni se duplique, consumiéndose en el orden en que es producida. Para ello, se considera un *buffer* común, del que el consumidor toma datos y donde el productor los coloca. Se debe mutuo-excluir, porque ambos, productor y consumidor, pueden modificarlo.

- Cuando el productor desea colocar un dato en el buffer y éste está lleno, el productor se quedará bloqueado hasta que exista espacio disponible en el buffer.
- Cuando el consumidor desea tomar un dato en el buffer y éste está vacío, el consumidor se quedará bloqueado hasta que exista algún dato disponible en el buffer.

Se desea construir un programa en C++ que coordine los procesos *productor* y consumidor, utilizando *semáforos* e *hilos*.

El programa debe contener al menos:

- Un procedimiento *productor*, que represente el comportamiento del productor de datos.
- Un procedimiento *consumidor*, que represente el comportamiento de un consumidor de datos.
- Un procedimiento principal (*main*), que realice la inicialización de los *semáforos* y *variables compartidas*, así como la creación de *hilos* para el productor y el consumidor.

Además, el programa deberá imprimir en pantalla:

- Cuando el *productor* produce un dato y luego lo coloca en el buffer.
- Cuando el *consumidor* saca un dato del buffer y luego lo consume.

Considere que el dato es de tipo *char*.

Se pide:

- Construir un programa en C++, denominado *productorconsumidor.cpp*, que implemente la coordinación entre el productor y el consumidor, según lo explicado en la parte anterior. Se pueden utilizar librerías de C++ y estructuras de datos que consideren necesario. Para el manejo de *semáforos* se deberá utilizar la librería *semaphore.h* y para el manejo de *hilos* se deberá utilizar la librería *pthread.h*.

- Crear un breve informe, denominado *informe.pdf*, que especifique los criterios de diseño utilizados para la construcción del programa solicitado. En la primera página del informe se deberá mencionar los datos (*nombre*, *apellido*, *cédula de identidad*) de todos los integrantes del grupo.

Un único integrante de cada grupo deberá subir al *receptor de entregas* del Obligatorio 2, que se encuentra el Campus Virtual[1], un único archivo denominado *oblig2GrX.zip* que contenga todos los archivos solicitados (*archivo fuente* e *informe*) antes del **martes 13 de noviembre a las 23:55 hs.**

No se aceptarán entregas fuera del plazo establecido, ni se aceptarán entregas que se envíen por correo electrónico. No está permitido compartir código entre los grupos, ni enviar código a los foros de consulta.

Además de la entrega, cada grupo deberá defender su obligatorio según el siguiente calendario:

Grupo	Fecha y hora
1	14/11 de 20:00 a 20:45 hs
2	14/11 de 20:45 a 21:30 hs
3	14/11 de 21:30 a 22:15 hs
4	14/11 de 22:15 a 23:00 hs
5	16/11 de 20:00 a 20:45 hs
6	16/11 de 20:45 a 21:30 hs
7	16/11 de 21:30 a 22:15 hs

La asistencia a la defensa es *obligatoria* para todos los integrantes del grupo. Aquellos estudiantes que no asistan a la defensa, no aprobarán el obligatorio.

Las defensas durarán 45 minutos aprox. y consistirán en mostrar la ejecución del programa. Además, se realizarán algunas preguntas a los integrantes del grupo acerca del diseño y construcción del programa.

[1] Campus Virtual de Sistemas Operativos: https://cv.utu.edu.uy/course/view.php?id=1190