Practica 3

Sistemas Operativos

Grupo: 2214

Cristian Tatu Pareja: 3 Sara Sanz

*Semana 1*

**Memoria Compartida**

**Ejercicio 3:**

Para resolver el problema del productor consumidor nos harán falta 3 semáforos. La implementación por la que hemos optado ha sido crear cada semáforo con una longitud igual al numero de elementos del buffer. Así podemos tener mas control sobre cada posición del array y quien lee y escribe en cada una.

El primer semáforo implementado es el que nos garantiza la exclusión mutua, *mutex,* garantizándonos que solo uno de los dos procesos accederá a la posición de memoria especifica en cada momento.

Dado que el consumidor no podrá leer nada a menos que el productor no haya colocado nada en la posición correspondiente y que el productor no pueda escribir nada a menos que el consumidor haya leído alguna posición, nos requiere el uso de otros 2 semáforos que nos indica que posición del array se puede leer y cual se puede escribir.

Si el consumidor es el primero que empieza hará un *down* en el semáforo *sePuedeLeer* en el índice correspondiente que solamente estará a 1 si el productor pasó antes por ahí para escribir algo. Así garantizamos que el consumidor tenga que esperar al productor si empieza antes.

Al igual pasa si es el productor el que empieza primero, aunque realmente el único problema que habría sería que el productor vaya mucho más rápido que el consumidor y que llegue a alcanzarle. Entonces el productor tendría que esperarle antes de poder escribir nada. Esto lo logra haciendo un *down* en el semáforo *sePuedeEscribir* que inicialmente esta inicializado a 1 para todas las posiciones del array, por tanto, cuando el consumidor lea ese elemento volverá a poner esa posición a 1 para que el productor pueda escribir otra vez.

Para poder contemplar los dos casos hemos añadido un argumento más al ejecutable para distinguir cual de los dos procesos va a empezar primero.

Observamos en la *Figura1.2* que, al haber empezado el productor primero, ha llegado a alcanzar al consumidor y por tanto sus índices están muy cerca uno del otro pero lo que es mas importante es que el índice del productor está por debajo del consumidor como esperábamos.

En la *Figura2* observamos lo contrario. Al haber empezado el consumidor llega al alcanzar al productor y por tanto sus índices están juntos pero el del consumidor es más bajo.

*Semana 2*

**Colas de Mensajes**

**Ejercicio 5:**

En contraste con las decisiones de diseño tomadas, otra solución al problema implementaría también colas de mensajes, pero solamente una. Ya que podemos darles a los mensajes un identificador distinto y así saber que mensajes pertenecen a que parte de la cadena de montaje.

Por ejemplo, los mensajes que se envían desde A a B podrían tener ID = 1 y los que se envían de B a C tener ID = 2. Así B solo leería los mensajes de tipo 1 y C los de tipo 2 sin tener que crear 2 colas de mensajes. Pero del enunciado hemos entendido que cada par de procesos que se van a comunicar deben tener una cola diferente.

Para la prueba del programa se ha generado antes un fichero (*random.txt*) que contiene solamente caracteres alfabéticos minúsculas y mayúsculas con el recurso de Linux *urandom*.

**ANEXO:**

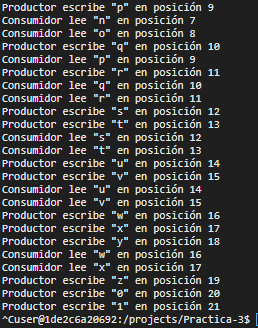
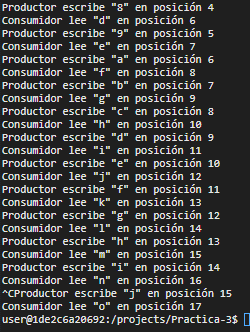


Ilustración 1[El Productor Empieza]

Ilustración 1.2[El Consumidor Empieza]