Memoria Productor-Consumidor
Práctica 1

Jesús Manuel García Palma

DESCRIPCIÓN DE LAS VARIABLES Y SEMÁFOROS

En la parte superior del .cpp tenemos los siguientes datos:

- num_items : cantidad de items de datos utilizados
- tam_vector : tamaño del vector_intermedio
- vector_intermedio : vector en el que se almacenarán los item de datos
- sem_t produce,consume : creación de los semáforos para producir y consumir el dato

En producir_dato():

- contador: imprime el dato producido

En productor y consumidor utilizaremos una cola circular o FIFO donde utilizaremos dos variables no negativas, primera_ocupada, que indica el índice en el vector de la primera celda ocupada, y que se incrementa al leer, y primera_libre, que es el índice en el vector de la primera celda libre, y se incrementa al escribir.

En productor:

Inicializamos primera_libre y creamos un bucle for que produzca los num_items. Para ello creamos una variable dato que contiene el dato producido en esa iteración.
Posteriormente llamamos a sem_wait(&produce), que nos indica, según su estado, si podemos producir. Este baja a 0 y y en la posición primera_libre%10 del vector intermedio guardamos la variable dato, incrementamos la variable primera_ocupada y llamamos a sem_post(&consume), desbloqueando y permitiendo que ahora se pueda consumir.

En consumidor:

Inicializamos primera_ocupada y creamos un bucle for que lea los num_items. Para ello creamos una variable dato. Posteriormente, llamamos a sem_wait(&consumir), que nos indica si podemos consumir. Este permite el paso, decrementando su valor, y dato será igual a la posición del vector_intermedio que llenamos en la función productor, aumentamos la variable primera_ocupada, llamamos a consumir dato(dato), para imprimir por pantalla que dato se ha consumido, y por último, llamamos a sem_post(&produce), que desbloquea y permite producir el siguiente dato.

En main, creamos las hebras prod,cons con las que trabajaremos.
Iniciamos los semaforos produce y consume. Inicialmente, se podrá producir y no se podrá leer.
Después, ejecutamos las hebras con pthread_create(&prod,NULL,productor,NULL) y pthread_create(&cons,NULL,consumidor,NULL).
Después llamamos a pthread_join con ambas hebras y finalizar los procesos: pthread_join(prod,NULL) y pthread_join(cons,NULL).
Finalmente destruimos los semáforos com sem_destroy y por último finalizamos el programa con pthread_exit(NULL);

CÓDIGO:

```
#include <iostream>
#include <cassert>
#include <pthread.h>
#include <semaphore.h>

using namespace std ;

// ---------------------------------------------------------------------
// constantes
const unsigned
num_items  = 40 ,
tam_vector = 10 ;

int vector_intermedio[tam_vector];

sem_t produce,consume;

// ---------------------------------------------------------------------
unsigned producir_dato()
{
static int contador = 1 ;
cout << "producido: " << contador << endl << flush ;
return contador++ ;
}
// ---------------------------------------------------------------------
void consumir_dato( int dato )
{
cout << "consumido: " << dato << endl ;
}
// ---------------------------------------------------------------------
void * productor( void * )
{ int primera_libre=0;
for( unsigned i = 0 ; i < num_items ; i++ )
{
int dato=producir_dato();
sem_wait(&produce);
vector_intermedio[primera_libre%10]=dato;
primera_libre++;
sem_post(&consume);
}
cout << "Fin hebra productora\n";
return NULL ;
}
// ---------------------------------------------------------------------
void * consumidor( void * )
{ int primera_ocupada=0;
for( unsigned i = 0 ; i < num_items ; i++ )
{
int dato;
sem_wait(&consume);
dato=vector_intermedio[primera_ocupada%10];
primera_ocupada++;
consumir_dato( dato ) ;
sem_post(&produce);
}
cout << "Fin hebra consumidora\n";
return NULL ;
}
//----------------------------------------------------------------------
```

```
int main()
{
pthread_t prod,cons; //creacion de las dos hebras

sem_init(&produce,0,1); //inicialmente se puede producir
sem_init(&consume,0,0); //inicialmente no se puede consumir

pthread_create(&prod,NULL,productor,NULL); //ejecuta productor
pthread_create(&cons,NULL,consumidor,NULL); //ejecuta consumidor

pthread_join(prod,NULL);
pthread_join(cons,NULL);

sem_destroy(&produce); //destruye semaforo
sem_destroy(&consume); //destruye semaforo

cout << "FIN\n";

pthread_exit(NULL);

return 0 ;
}
```