ejercicio1.pdf

Clarasdfgh

Sistemas Concurrentes y Distribuidos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Examen prácticas 1, 2 -- SCD 2020

Ejercicio 1:

Modificar la solución al problema de los fumadores planteado en la Práctica 1 tal como se muestra a continuación, adjuntando el archivo .cpp resultante con nombre ejercicio1.cpp:

En lugar de tener tres fumadores, se tendrán cuatro fumadores: fumador[0], fumador[1],...,fumador[3] y ahora se necesitan cuatro ingredientes para fumar.

Se dispondrá de una única función para crear las cuatro hebras fumadoras (el comportamiento de una hebra fumadora concreta dependerá del argumento que se le pase a la función).

La única diferencia sustancial con respecto al problema trabajado en el guión de prácticas, además de existir un ingrediente y un fumador más (que ya supone algunas adiciones en el código existente), es la siguiente:

Los fumadores registrarán el número de cigarros totales fumados por todos los fumadores. Cuando un fumador ve que su ingrediente está en el mostrador, si el número de cigarrillos es par, dicho fumador solo avisará al estanquero (de que puede poner más ingredientes) cuando haya terminado de fumar y si es impar avisará al estanquero antes de fumar. Se ha de tener en cuenta que el número total de cigarros fumados en un instante (valor global para todos los fumadores) solo se incrementa justo después de que un fumador fuma.

```
#include <iostream>
#include <cassert>
#include <thread>
#include <mutex>
#include <random> // dispositivos, generadores y distribuciones aleatorias
#include <chrono> // duraciones (duration), unidades de tiempo
#include "Semaphore.h"

using namespace std ;
using namespace SEM ;

//**********************************************************************
// variables compartidas

const int num_items = 4;
int       contador  = 0;

Semaphore mostrador = 1;
std::vector<Semaphore> fumador;


mutex       mtx;


//**********************************************************************
// plantilla de función para generar un entero aleatorio uniformemente
// distribuido entre dos valores enteros, ambos incluidos
// (ambos tienen que ser dos constantes, conocidas en tiempo de compilación)
//----------------------------------------------------------------------
```

```
template< int min, int max > int aleatorio()
{
  static default_random_engine generador( (random_device())() );
  static uniform_int_distribution<int> distribucion_uniforme( min, max ) ;
  return distribucion_uniforme( generador );
}

//----------------------------------------------------------------------
// Función que simula la acción de producir un ingrediente, como un retardo
// aleatorio de la hebra (devuelve número de ingrediente producido)

int producir_ingrediente()
{
   // calcular milisegundos aleatorios de duración de la acción de fumar)
   chrono::milliseconds duracion_produ( aleatorio<10,100>() );

   // informa de que comienza a producir
   mtx.lock();
   cout << "Estanquero : empieza a producir ingrediente (" <<
duracion_produ.count() << " milisegundos)" << endl;
   mtx.unlock();

   // espera bloqueada un tiempo igual a ''duracion_produ' milisegundos
   this_thread::sleep_for( duracion_produ );

   const int num_ingrediente = aleatorio<0,num_items-1>() ;

   // informa de que ha terminado de producir
   mtx.lock();
   cout << "Estanquero : termina de producir ingrediente " << num_ingrediente <<
endl;
   mtx.unlock();

   return num_ingrediente ;
}

//----------------------------------------------------------------------
// función que ejecuta la hebra del estanquero

void funcion_hebra_estanquero(  )
{
    while( true ){
      sem_wait( mostrador );
        int i = producir_ingrediente();

        mtx.lock();
        cout << "Estanquero : ingrediente " << i << " puesto en el mostrador" <<
endl;
        mtx.unlock();

      sem_signal( fumador[i] );
    }

}

//----------------------------------------------------------------------
// Función que simula la acción de fumar, como un retardo aleatoria de la hebra
```

Cuando disfrutas de tu gente y de la cerveza,
con cabeza, disfrutas el doble.

Cerveceros de España
100 años más

UNA GRAN CERVEZA.
UNA GRAN RESPONSABILIDAD.

```
void fumar( int num_fumador )
{

   // calcular milisegundos aleatorios de duración de la acción de fumar)
   chrono::milliseconds duracion_fumar( aleatorio<20,200>() );

   // informa de que comienza a fumar
   mtx.lock();
    cout << "Fumador " << num_fumador << "  :"
         << " empieza a fumar (" << duracion_fumar.count() << " milisegundos)"
<< endl;
   mtx.unlock();
   // espera bloqueada un tiempo igual a ''duracion_fumar' milisegundos
   this_thread::sleep_for( duracion_fumar );

   // informa de que ha terminado de fumar
    mtx.lock();
    cout << "Fumador " << num_fumador << "  : termina de fumar, comienza espera
de ingrediente." << endl;
    mtx.unlock();

    contador++;

}

//----------------------------------------------------------------------
// función que ejecuta la hebra del fumador
void  funcion_hebra_fumador( int num_fumador )
{
   while( true )
   {
     sem_wait( fumador[num_fumador] );

     mtx.lock();
     cout << "Fumador " << num_fumador << "  : toma ingrediente del mostrador" <<
endl;
     mtx.unlock();

     if(contador % 2 == 0){
       fumar(num_fumador);

       sem_signal( mostrador );

     } else{
       sem_signal( mostrador );

       fumar(num_fumador);

     }

   }
}

//----------------------------------------------------------------------


//----------------------------------------------------------------------
```

```
int main()
{
  cout << "-------------------------------------------------" << endl
       << " Problema de los fumadores -- examen, ejercicio 1 " << endl
       << "-------------------------------------------------" << endl
       << flush ;

       for (int i = 0; i < num_items; i++){
         fumador.push_back(0);
       }

       thread estanquero;
       thread fumadores[num_items];

       for (int i = 0; i < num_items; i++){
         fumadores[i] = thread (funcion_hebra_fumador, i);
       }

         estanquero = thread (funcion_hebra_estanquero);


       for (int i = 0; i < num_items; i++){
         fumadores[i].join();
       }

         estanquero.join();

       cout << endl << endl;

}
```