Alberto Rodríguez Santana 48950011-P Grupo D1

PRACTICA 1

Fumadores

Variables

Como variables compartidas únicamente tenemos declarados los semáforos para la sincronización de las hebras.

En la hebra principal (main) está declarada la variable num_hebras para el control de los bucles.

```
int main()
{

    const int num_hebras = 3;

    thread hebra_fumador[3];

    thread hebra_estanquero (funcion_hebra_estanquero);

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

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

   hebra_estanquero.join();


}
```

```
Semaphore ingr_disp[3] = {0, 0, 0};
Semaphore mostr_vacio = 1;
```

Semáforos

- Ingr_disp -> Vector de 3 semáforos para cada ingrediente, inicialmente a 0 cada uno de ellos ya que se debe permitir colocar los ingredientes en el mostrador. Valdrá 1 si el ingrediente i esta disponible en el mostrador y 0 si no. Sobre este semáforo se realiza sem_wait una vez puesto el ingrediente y sem_signal antes de retirar el ingrediente.
- mostr_vacio -> Inicializado a 1. Tomará valor 0 si hay un ingrediente sobre el mostrador y 1 si el mostrador está vacío. Sobre este semáforo hay que hace un sem_wait antes de indicar que ingrediente se ha puesto sobre el mostrador y un sem_signal una vez algún fumador haya retirado el ingrediente.

```
Semaphore ingr_disp[3] = {0, 0, 0};
Semaphore mostr_vacio = 1;
```

```
void funcion_hebra_estanquero(  )
{
    int i;

    while(true){
        i = aleatorio<0,2>(); //Produce entero aleatorio con retraso;

        sem_wait(mostr_vacio);

        cout<<"\nPuesto ingrediente: " << i << endl;

        sem_signal(ingr_disp[i]);
    }

}
```

```
// función que ejecuta la hebra del fumador
void  funcion_hebra_fumador( int num_fumador )
{
   while( true ){

        sem_wait(ingr_disp[num_fumador]);

        cout<<"\nRetirado ingrediente: " << num_fumador << "\n";

        sem_signal(mostr_vacio);

        fumar(num_fumador);
   }
}
```

Código Fuente

```cpp
#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 ;

Semaphore ingr_disp[3] = {0, 0, 0};
Semaphore mostr_vacio = 1;

//**********************************************************************
// 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 ejecuta la hebra del estanquero
void funcion_hebra_estanquero(  )
{
    int i;
    while(true){
        i = aleatorio<0,2>(); //Produce entero aleatorio con retraso;

        sem_wait(mostr_vacio);

        cout<<"\nPuesto ingrediente: " << i << endl;

        sem_signal(ingr_disp[i]);
    }
}
```

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

    cout << "\nFumador " << num_fumador << ":"
         << " empieza a fumar (" << duracion_fumar.count() << " milisegundos)" << endl;

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

   // informa de que ha terminado de fumar

    cout << "\nFumador " << num_fumador << ": termina de fumar, comienza espera de ingrediente." << endl;

}
//----------------------------------------------------------------------
// función que ejecuta la hebra del fumador
void  funcion_hebra_fumador( int num_fumador )
{
   while( true ){

        sem_wait(ingr_disp[num_fumador]);

        cout<<"\nRetirado ingrediente: " << num_fumador << "\n";

        sem_signal(mostr_vacio);

        fumar(num_fumador);
   }
}
```

```
int main()
{

    const int num_hebras = 3;

    thread hebra_fumador[3];

    thread hebra_estanquero (funcion_hebra_estanquero);

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

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

   hebra_estanquero.join();


}
```