elver_g4larg4
www.wuolah.com/student/elver_g4larg4

543

P2-2SCD.pdf

Fumadores RESUELTO Practica 2

2° Sistemas Concurrentes y Distribuidos

Grado en Ingeniería Informática

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

Ejercicio fumadores
p2 SCD
Resuelto

Para el **otro ejercicio o los cpp comprimidos** descargar el extraible p2_resuelta de mi perfil

by: elver_g4larg4

```
#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 ;

const int num_fumadores = 3;
Semaphore mostr_vacio = 1;
Semaphore ingr_disp[num_fumadores]  = {0,0,0};
//**************************************************************************
// 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
   cout << "Estanquero : empieza a producir ingrediente (" << duracion_produ.count() << "
milisegundos)" << endl;

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

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

   // informa de que ha terminado de producir
   cout << "Estanquero : termina de producir ingrediente " << num_ingrediente << endl;

   return num_ingrediente ;
}

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

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

void funcion_hebra_estanquero(  )
{
        int ingr;
        while (true){
                ingr = producir_ingrediente();
                sem_wait(mostr_vacio);
                cout << "Se ha puesto el ingrediente numero " << ingr << endl;
                sem_signal(ingr_disp[ingr]);
        }
}

//----------------------------------------------------------------------
// 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 << "Fumador " << 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 << "Fumador " << 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 << "Se ha retirado el ingrediente numero " << num_fumador << endl;
                sem_signal(mostr_vacio);
                fumar(num_fumador);
  }
}

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

```
int main()
{

  thread hebra_estanquero(funcion_hebra_estanquero);
  thread hebras_fumadores[num_fumadores];

  for(int i  =0; i < num_fumadores;i++)
        hebras_fumadores[i] = thread(funcion_hebra_fumador,i);

  hebra_estanquero.join();
}
```