ejercicio2.pdf

cerola

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 2:

Modifica tu solución al problema de los Lectores-Escritores de la práctica 2, tal como se indica a continuación, adjuntando el archivo .cpp resultante con nombre ejercicio2.cpp:

Se lanzarán 5 hebras lectoras y 3 hebras escritoras.

Un escritor requiere acceso exclusivo al recurso, pero los escritores acceden al recurso por parejas de forma que entra uno inmediatamente después de que salga el primero. Esto implica que, cuando logra entrar un escritor, la siguiente hebra que entraría a la estructura de datos cuando este escritor sale debería ser obligatoriamente otro escritor. Cuando sale este segundo escritor, ya se podría dar preferencia a los lectores para entrar.

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

using namespace std ;
using namespace HM;


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

const int num_lectores   = 5,
          num_escritores = 3;

mutex   mtx;  //Mutex para la salida por pantalla


//**************************************************************************
// 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 );
}


//**************************************************************************
// Monitor hoare lect_escr
//--------------------------------------------------------------------------
```

```
class Lect_escr : public HoareMonitor
{
private:
int     n_lec;
bool    escrib;
CondVar lectura,
        escritura;
int     contador;

public:
  Lect_escr( );
  void ini_lectura( );
  void fin_lectura( );
  void ini_escritura( );
  void fin_escritura( );
};

// ----------------------------------------------------------------------------
Lect_escr::Lect_escr( ){
  n_lec = 0;
  escrib = false;

  lectura = newCondVar();
  escritura = newCondVar();
}

// ----------------------------------------------------------------------------
void Lect_escr::ini_lectura( ){
  if (escrib){
    lectura.wait();
  }

  n_lec++;
  lectura.signal();

}

// ----------------------------------------------------------------------------
void Lect_escr::fin_lectura( ){
  n_lec--;

  if (n_lec == 0){
    escritura.signal();
  }
}

// ----------------------------------------------------------------------------
void Lect_escr::ini_escritura( ){
  if (escrib || n_lec > 0){
    escritura.wait();
  }

  escrib = true;
  if(contador == 0){
    contador++;
  } else{
    contador = 0;
```

AP Academia

Cursos
Semi Intensivos

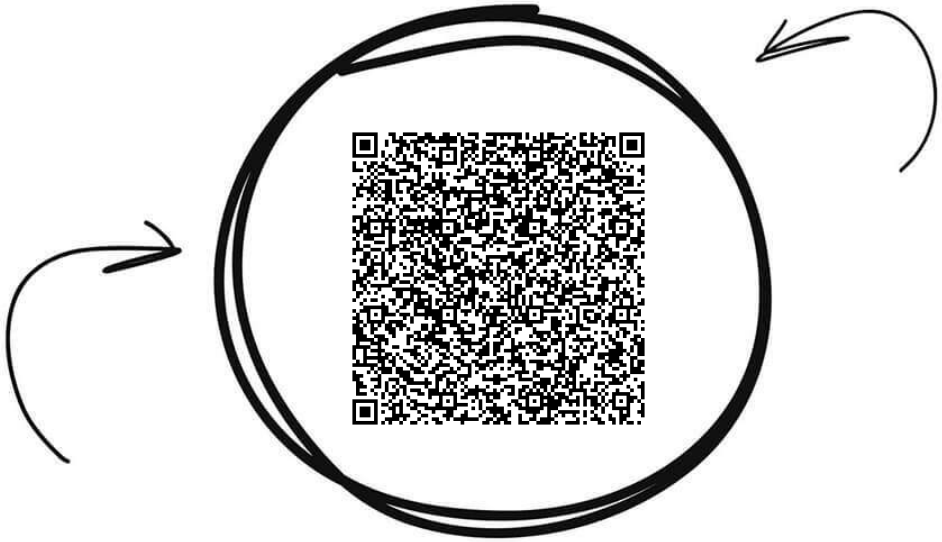
Prepárate para el examen **Aptis**

- **8 semanas** de duración
- **4h** semanales
- **2h** cada clase

Obtén más información

*También **Cursos Beginners** para que no temas al inglés

Sistemas Concurrentes y Dist...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
  }
}

// -----------------------------------------------------------------------------
void Lect_escr::fin_escritura( ){
  escrib = false;

  if (contador > 0){
    escritura.signal();
  } else{
    lectura.signal();
  }

  /*if(escritura.get_nwt() > 0 && n_lec >= 3){
    if(n_lec > 3){
      escritura.signal();
    } else{
      lectura.signal();
    }
  } else{
    lectura.signal();
  }*/

  /*if (lectura.get_nwt() != 0) //Función de CondVar que comprueba cuántas hebras
hay esperando en ella
  {
    lectura.signal();
  } else{
    escritura.signal();
  }*/
}


//-----------------------------------------------------------------------------
// Función que simula la acción de escribir, como un retardo
// aleatorio de la hebra

void escribir( int n_escritor )
{
   // calcular milisegundos aleatorios de duración de la acción de escribir
   chrono::milliseconds duracion_escr( aleatorio<10,100>() );

   // informa de que comienza a escribir
   mtx.lock();
   cout << "Escritor " << n_escritor << " : empieza a escribir (" <<
duracion_escr.count() << " milisegundos)" << endl;
   mtx.unlock();

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

   // informa de que ha terminado de escribir
   mtx.lock();
   cout << "Escritor " << n_escritor << " : termina de escribir " << endl;
   mtx.unlock();
}

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


el podcast para entender que la vida da mas vueltas que la silla de un peluquero


```
// Función que ejecuta la hebra del escritor

void funcion_hebra_escritor( MRef<Lect_escr> monitor, int n_escritor )
{
    while( true ){
      chrono::milliseconds espera( aleatorio<10,100>() );
      this_thread::sleep_for( espera );

      monitor->ini_escritura();
      escribir(n_escritor);
      monitor->fin_escritura();
    }

}

//----------------------------------------------------------------------------
// Función que simula la acción de leer, como un retardo
// aleatorio de la hebra

void leer( int n_lector )
{
   // calcular milisegundos aleatorios de duración de la acción de leer
   chrono::milliseconds duracion_lect( aleatorio<10,100>() );

   // informa de que comienza a leer
   mtx.lock();
   cout << "Lector   " << n_lector << " : empieza a leer (" <<
duracion_lect.count() << " milisegundos)" << endl;
   mtx.unlock();

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

   // informa de que ha terminado de leer
   mtx.lock();
   cout << "Lector   " << n_lector << " : termina de leer " << endl;
   mtx.unlock();
}


//----------------------------------------------------------------------------
// Función que ejecuta la hebra del lector

void  funcion_hebra_lector( MRef<Lect_escr> monitor, int n_lector )
{
   while( true )
   {
     chrono::milliseconds espera( aleatorio<100,120>() );
     this_thread::sleep_for( espera );

     monitor->ini_lectura();
     leer(n_lector);
     monitor->fin_lectura();

   }
}

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

```
int main()
{
  cout << "------------------------------------------------------------------------
-----" << endl
       << " Problema de los lectores-escritores con monitor SU -- examen,
ejercicio 2 " << endl
       << "------------------------------------------------------------------------
-----" << endl
       << flush ;

  MRef<Lect_escr> monitor = Create<Lect_escr>( );
  thread lectores[num_lectores];
  thread escritores[num_escritores];


  for (int i = 0; i < num_lectores; i++){
    lectores[i] = thread (funcion_hebra_lector, monitor, i);
  }

  for (int i = 0; i < num_escritores; i++){
    escritores[i] = thread (funcion_hebra_escritor, monitor, i);
  }


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

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

  cout << endl << endl;
}
```