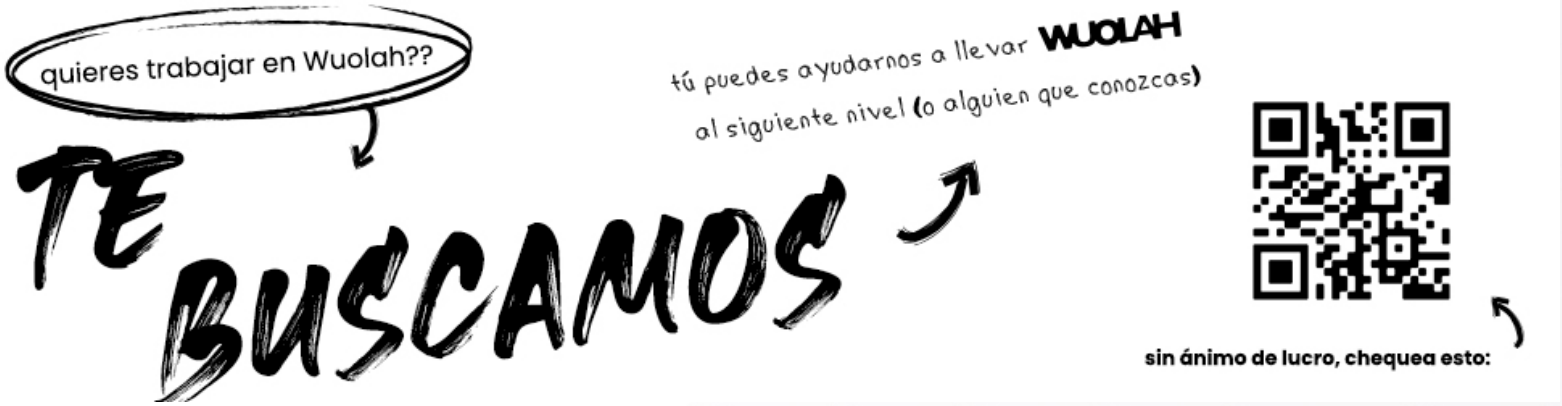
examenanamaria.pdf

pr0gramming_312823

Sistemas Concurrentes y Distribuidos

2º Grado en Ingeniería Informática

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



Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

quieres trabajar en Wuolah??
TE BUSCAMOS

Enunciados del examen SCD P1 y P2

1- Modificar el problema de los fumadores de la práctica 1 (semáforos) de la siguiente manera (el archivo se debe llamar "**fuma_ex.cpp**"):

Habrá un **nuevo estanquero** en el problema (con lo que ahora hay **estanquero1 y estanquero2**). **Se van turnando** entre ellos de la siguiente manera:

- **Cada vez que el estanquero1 genera dos veces seguidas el mismo ingrediente**, le toca **descansar** y **la labor del estanquero pasa a ser desempeñada por el estanquero2**.

- **Cada vez que el estanquero2 genera cuatro ingredientes**, le toca **descansar** y la **labor del estanquero pasa a ser desempeñada por el estanquero1**.

- El **estanquero1 es el que comienza poniendo ingredientes**.

2- Modificar el problema de los lectores-escritores de la práctica 2 (monitores) de la siguiente manera (el archivo se debe llamar "**lec_esc_ex.cpp**"):

- En el **procedimiento "fin_escritura", la prioridad debe ser de los escritores**. El **número de procesos lectores será cinco y el de escritores, siete**.

- Se debe crear una **nueva hebra "fumigadora"**, que será un **ciclo infinito** y cuyo código será solamente un **nuevo procedimiento del monitor que se llama "fumigar**", seguido de un **retardo aleatorio que simula la fumigación**. La hebra "**fumigadora" debe estar bloqueada siempre que no haya que hacer la fumigación**.

- Hay que hacer una **fumigación cada vez que haya cuatro accesos completados de escritura seguidos** (es decir, sin ningún acceso de lectura entre ellos)

- **Cada vez que haya una fumigación se debe indicar con un mensaje**, cuál es el **identificador del cuarto proceso escritor que ha accedido al recurso**.

3- Un programa concurrente está formado por pacientes (ciclos infinitos) que acuden a una clínica dental. Los clientes acuden por dos motivos (limpieza bucal o empaste de muela). Inicialmente, los pacientes deciden de forma aleatoria el motivo de la visita a la clínica. En la clínica hay dos tipos de profesionales trabajando, cuatro higienistas dentales (que solo realizan limpiezas bucales) y dos dentistas (que solo realizan empastes), en ambos casos se consideran recursos del problema (no hebras) que solo pueden ser accedidos por un paciente en cada momento. En los pacientes, el tiempo de consulta y la parte de código fuera de la clínica se simulan con retardos aleatorios. Los límites temporales para el retardo aleatorio de los empastes deberán ser el doble que el de los límites para el retardo de las limpiezas bucales. Habrá catorce hebras paciente. Implementar dicho programa (el archivo se deberá llamar "ejercicio3.cpp") con un monitor de acuerdo al siguiente esquema. Incluir mensajes que permitan seguir la traza del programa.

sin ánimo de lucro, chequea esto:

tú puedes ayudarnos a llevar WUOLAH al siguiente nivel (o alguien que conozcas)

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

```
 1: #include <iostream>
 2: #include <iomanip>
 3: #include <cassert>
 4: #include <thread>
 5: #include <mutex>
 6: #include <condition_variable>
 7: #include <chrono> // duraciones (duration), unidades de tiempo
 8: #include <random> // dispositivos, generadores y distribuciones aleatorias
 9: #include "HoareMonitor.h"
10:
11: using namespace std;
12: using namespace HM;
13:
14: const int num_pacientes = 14;
15: const int num_higienistas = 4;
16: const int num_dentistas = 2;
17: const int num_profesional = num_higienistas+num_dentistas;
18:
19: mutex mtx;
20:
21: //**************************************************************************
22: // plantilla de funciÃ³n para generar un entero aleatorio uniformemente
23: // distribuido entre dos valores enteros, ambos incluidos
24: // (ambos tienen que ser dos constantes, conocidas en tiempo de compilaciÃ³n)
25: //--------------------------------------------------------------------------
26:
27: template< int min, int max > int aleatorio()
28: {
29:   static default_random_engine generador( (random_device())() );
30:   static uniform_int_distribution<int> distribucion_uniforme( min, max ) ;
31:   return distribucion_uniforme( generador );
32: }
33:
34: void profesionales_espera(int TC){
35:     chrono::milliseconds dur_escritura(aleatorio<20,200>());
36:
37:     //  Espera bloqueada de dur_escritura milisegundos
38:     if(TC == 1){ // Si es empaste
39:         this_thread::sleep_for(dur_escritura*2);
40:         mtx.lock();
41:         cout << "Empastando.. " << endl;
42:         mtx.unlock();
43:     }else
44:     {
45:         this_thread::sleep_for(dur_escritura);
46:         mtx.lock();
47:         cout << "Limpiando boca.. " << endl;
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

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.

```
48:         mtx.unlock();
49:     }
50:
51: }
52:
53: void Espera(){
54:     chrono::milliseconds tiempo(aleatorio<20,200>());
55:     this_thread::sleep_for(tiempo);
56: }
57:
58: class Clinica: public HoareMonitor{
59:     private:
60:         int num_dent, num_hig, num_pac;
61:
62:         CondVar pacientes, profesionales;
63:
64:     public:
65:         Clinica(){
66:             num_dent = 2;
67:             num_hig = 4;
68:             num_pac = 0;
69:             pacientes = newCondVar();
70:             profesionales = newCondVar();
71:         }
72:         void acceso_consulta(int TC){
73:             num_pac++;
74:             if(TC == 0){ //  Si el motivo es limpieza
75:                 cout << "Entra paciente con limpieza" << endl;
76:                 if(num_hig == 0){
77:                     pacientes.wait(); // Si estan ocupados se espera
78:                 }
79:                 else{
80:                     num_hig--;
81:                     profesionales.signal(); // Se lanza una hebra profesional y se atiende al paciente
82:                 }
83:             }
84:             else if(TC == 1){    // Si el motivo es empaste
85:                 cout << "Entra paciente con empaste" << endl;
86:                 if(num_dent == 0) //mismos comentarios que arriba
87:                     pacientes.wait();
88:                 else
89:                 {
90:                     num_dent--;
91:                     profesionales.signal();
92:                 }
93:             }
94:         }
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

```
 95:         void fin_consulta(int TC){
 96:             num_pac--;
 97:             if(TC == 0){
 98:                 num_hig++;  // Se libera un higienista
 99:                 mtx.lock();
100:                 cout << "Se ha terminado de limpiar la boca" << endl;
101:                 mtx.unlock();
102:                 pacientes.signal();
103:             }
104:             else if(TC == 1){
105:                 num_dent++;
106:                 mtx.lock();
107:                 cout << "Se ha terminado de empastar" << endl;
108:                 mtx.unlock();
109:                 pacientes.signal();
110:             }
111:             profesionales.signal();
112:         }
113: };
114:
115: void funcion_hebra_pacientes(MRef<Clinica> monitor, int numPaciente){
116:     int motivo;
117:     while(true){
118:         motivo = aleatorio<0,1>();
119:         monitor->acceso_consulta(motivo);
120:         profesionales_espera(motivo);
121:         monitor->fin_consulta(motivo);
122:         Espera();
123:     }
124: }
125:
126:
127: int main(){
128:     cout << "------------------------------------------------------" << endl <<
129:             "- Problema de la clinica. Monitor SU. --" << endl <<
130:             "------------------------------------------------------" << endl << flush;
131:     MRef<Clinica> monitor = Create<Clinica>();
132:
133:     thread hebras_paciente[num_pacientes], hebras_profesional[num_profesional];
134:
135:     for(int i = 0; i < num_pacientes; i++)
136:         hebras_paciente[i] = thread(funcion_hebra_pacientes, monitor, i);
137:
138:     for(int i = 0; i < num_pacientes; i++)
139:         hebras_paciente[i].join();
140:
141:
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

LOS JUEGOS DEL CUATRI
te imaginas no pagar ni primera ni segunda matrícula??
WUOLAH


```
142:
143:  }
```

matrícula GRATIS

PARTICIPA

Si consigues subir más apuntes que tus compañeros te regalamos una mátrícula valorada en 1000€

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

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

using namespace std;
using namespace HM;

const int num_lectores = 5;
const int num_escritores = 7;

mutex mtx;

//**********************************************************************
// plantilla de funciÃ³n para generar un entero aleatorio uniformemente
// distribuido entre dos valores enteros, ambos incluidos
// (ambos tienen que ser dos constantes, conocidas en tiempo de compilaciÃ³n)
//----------------------------------------------------------------------

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 );
}

void escribir(int escritor){
    chrono::milliseconds dur_escritura(aleatorio<20,200>());

    //  Se empieza a escribir
    mtx.lock();
    cout << "El escritor " << escritor << " comienza a escribir ( " << dur_escritura.count() << " milisegundos)" << end
    mtx.unlock();

    //  Espera bloqueada de dur_escritura milisegundos
    this_thread::sleep_for(dur_escritura);

    //  Informa que ha terminado de escribir
    mtx.lock();
    cout << "El escritor " << escritor << " ha terminado de escribir" << endl;
    mtx.unlock();

```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

```
48: }
49:
50: void Espera(){
51:     chrono::milliseconds tiempo(aleatorio<20,200>());
52:     this_thread::sleep_for(tiempo);
53: }
54:
55: void leer(int lector){
56:     chrono::milliseconds dur_lectura(aleatorio<20,200>());
57:
58:     //  Se empieza a leer
59:     mtx.lock();
60:     cout << "El lector " << lector << " comienza a leer ( " << dur_lectura.count() << " milisegundos)" << endl;
61:     mtx.unlock();
62:
63:     //  Espera bloqueada de dur_lectura milisegundos
64:     this_thread::sleep_for(dur_lectura);
65:
66:     //  Informa que ha terminado de leer
67:     mtx.lock();
68:     cout << "El lector " << lector << " ha terminado de leer" << endl;
69:     mtx.unlock();
70:
71: }
72:
73: class LectoresEscritores: public HoareMonitor{
74:     private:
75:         int num_lec, num_esc;
76:         bool escribiendo, afumigar;
77:
78:         CondVar lectura, escritura, fumigadora;
79:
80:     public:
81:         LectoresEscritores(){
82:             num_lec = 0;
83:             num_esc = 0;
84:             escribiendo = false;
85:             afumigar = false;
86:             lectura = newCondVar();
87:             escritura = newCondVar();
88:             fumigadora = newCondVar();
89:         }
90:         void iniLec(){
91:             if(escribiendo || num_esc == 0 )
92:                 lectura.wait();
93:             num_lec++;
94:             num_esc = 0;
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

```
 95:             lectura.signal();
 96:         }
 97:         void finLec(){
 98:             num_lec--;
 99:             if(num_lec == 0)
100:                 escritura.signal();
101:         }
102:         void iniEsc(){
103:             if(num_lec > 0 || escribiendo)
104:                 escritura.wait();
105:             escribiendo = true;
106:         }
107:         void finEsc(){
108:             num_esc++;
109:             escribiendo = false;
110:
111:             if(num_esc == 4){
112:                 afumigar = true;
113:                 fumigadora.signal();
114:                 cout << "ESCRITOR " << escritura.get_nwt() << " LANZA FUMIGADORA" <<endl;
115:                 num_esc = 0;
116:             }
117:
118:             if(!lectura.empty())
119:                 lectura.signal();
120:             // else
121:                 escritura.signal();
122:         }
123:         void fumigar(){
124:             if(afumigar){
125:             afumigar = false;
126:             cout << "FUMIGANDO..." << endl;
127:             chrono::milliseconds dur_lectura(aleatorio<20,200>());
128:             this_thread::sleep_for(dur_lectura);
129:             }
130:             else{
131:                 fumigadora.wait();
132:             }
133:         }
134: };
135:
136: void funcion_hebra_fumigadora(MRef<LectoresEscritores> monitor){
137:     while(true){
138:         Espera();
139:         monitor->fumigar();
140:         cout << "Se ha terminado de fumigar" << endl;
141:     }
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

LOS JUEGOS DEL CUATRI
quieres la play quinta?? (no digo el númerito porque ya nos conocemos, don comedia)
WUOLAH


```
142: }
143: void funcion_hebra_lector(MRef<LectoresEscritores> monitor, int numLectores){
144:     while(true){
145:         Espera();
146:         monitor->iniLec();
147:         leer(numLectores);
148:         monitor->finLec();
149:     }
150: }
151:
152: void funcion_hebra_escritor(MRef<LectoresEscritores> monitor, int numEscritores){
153:     while(true){
154:         Espera();
155:         monitor->iniEsc();
156:         escribir(numEscritores);
157:         monitor->finEsc();
158:     }
159: }
160:
161:
162: int main(){
163:     cout << "-------------------------------------------------------" << endl <<
164:             "- Problema de los lectores y escritores con fumigacion. Monitor SU. --" << endl <<
165:             "-------------------------------------------------------" << endl << flush;
166:     MRef<LectoresEscritores> monitor = Create<LectoresEscritores>();
167:
168:     thread hebras_lectoras[num_lectores], hebras_escritoras[num_escritores];
169:     thread hebra_fumigadora(funcion_hebra_fumigadora, monitor);
170:
171:     for(int i = 0; i < num_lectores; i++)
172:         hebras_lectoras[i] = thread(funcion_hebra_lector, monitor, i);
173:
174:     for(int i = 0; i < num_escritores; i++)
175:         hebras_escritoras[i] = thread(funcion_hebra_escritor, monitor, i);
176:
177:
178:     for(int i = 0; i < num_lectores; i++)
179:         hebras_lectoras[i].join();
180:
181:     for(int i = 0; i < num_escritores; i++)
182:         hebras_escritoras[i].join();
183:     hebra_fumigadora.join();
184:
185:  }
```

PARTICIPA

Será sorteada entre todos los usuarios estudiantes que el día de la finalización del concurso estén en el top de su comunidad

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH