

WUOLAH



ElonMusk

www.wuolah.com/student/ElonMusk



6467

FumadoresMonitor.pdf

Fumadores Monitor Resuelto



2º Sistemas Concurrentes y Distribuidos



Grado en Ingeniería Informática



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



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.





**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

Fumadores. Enunciado:

En este ejercicio consideraremos de nuevo el mismo problema de los fumadores y el estanco cuyo solución con semáforos ya vimos en la práctica 1. Queremos diseñar e implementar un monitor SU (monitor Hoare) que cumpla los requerimientos:

Se mantienen las tres hebras de fumadores y la hebra de estanco.

Se mantienen exactamente igual todas las condiciones de sincronización entre esas hebras.

El diseño de la solución incluye un monitor (de nombre **Estanco**) y las variables condición necesarias.

Hay que tener en cuenta que ahora no disponemos de los valores de los semáforos para conseguir la sincronización. A continuación haremos un diseño del monitor, y se deja como actividad la implementación de dicho diseño.

Los fumadores, en cada iteración de su bucle infinito:

Llaman al procedimiento del monitor **obtenerIngrediente(i)**, donde *i* es el número de fumador (o el número del ingrediente que esperan). En este procedimiento el fumador espera bloqueado a que su ingrediente esté disponible, y luego lo retira del mostrador.

Fuman, esto es una llamada a la función Fumar, que es una espera aleatoria.

El estanco, en cada iteración de su bucle infinito:

Produce un ingrediente aleatorio (llama a una función **ProducirIngrediente()**, que hace una espera de duración aleatoria y devuelve un número de ingrediente aleatorio).

Llama al procedimiento del monitor **ponerIngrediente(i)**, (se pone el ingrediente *i* en el mostrador) y después a **esperarRecogidaIngrediente()** (espera bloqueado hasta que el mostrador está libre).

Las variables permanentes necesarias se deducen de las esperas que deben hacer las hebras:

Cada fumador debe esperar a que el mostrador tenga un ingrediente y que ese ingrediente coincida con su número de ingrediente o fumador.

El estanco debe esperar a que el mostrador esté vacío (no tenga ningún ingrediente)

Como actividad, debes de escribir (en tu portafolio):

Variable o variables permanentes: para cada una describe el tipo, nombre, valores posibles y significado de la variable.

Cola o colas condición: para cada una, escribe el nombre y la condición de espera asociada (una expresión lógica de las variables permanentes).

Pseudo-código de los tres procedimientos del monitor.

Solución:

NOTA: AL HACER UN MONITOR SU (SEÑALAR Y ESPERA URGENTE) NECESITAMOS UNA CLASE YA HECHA (HOAREMONITOR) QUE TENDRÁS QUE AÑADIR EN LA INSTRUCCIÓN DE COMPILACIÓN

NOTA 2: LOS EJERCICIOS ESTÁN REALIZADOS EN BUCLES INFINITOS, PARA PARAR LA EJECUCIÓN DEL PROGRAMA Y COMPROBAR QUE TODO VA BIEN PULSA CTRL+D

```
#include <iostream>

#include <cassert>

#include <thread>

#include <mutex>

#include <random> // dispositivos, generadores y distribuciones aleatorias
#include <chrono> // duraciones (duration), unidades de tiempo

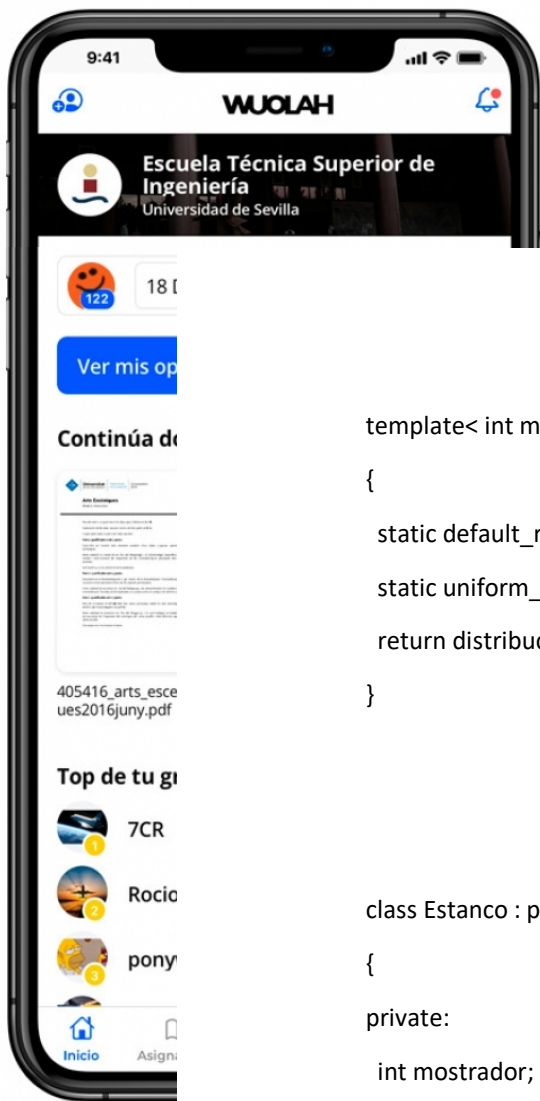
#include "HoareMonitor.h"

using namespace std ;
using namespace HM ;

//*****
*

// 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)
//-----

// número de ingredientes y de fumadores del problema
const int ingredientes = 3;
```



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



```
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 );
}

class Estanco : public HoareMonitor
{
private:
    int mostrador;

    CondVar mostr_vacio;          // indica si el mostrador está vacío
    CondVar ingr_disp[ingredientes]; // indica la disponibilidad de cada ingrediente

public:
    Estanco();
    void PonerIngrediente(const int i);
    void esperarRecogidaIngrediente();
    void obtenerIngrediente(const int i);
};

Estanco::Estanco( )
{
    mostrador = -1;
    mostr_vacio = newCondVar();

    for (int i = 0; i < ingredientes; ++i)
```

```

    ingr_disp[i] = newCondVar();
}

void Estanco :: PonerIngrediente (const int i)
{
    esperarRecogidaIngrediente();
    mostrador = i;
    cout << "Se ha puesto el ingrediente número " << i << endl;

    ingr_disp[i].signal();
}

void Estanco :: esperarRecogidaIngrediente ()
{
    if (mostrador != -1)
        mostr_vacio.wait();
}

void Estanco :: obtenerIngrediente (const int i)
{
    if (mostrador != i)
        ingr_disp[i].wait();

    cout << "Se ha retirado el ingrediente número " << i << endl;
    mostrador = -1; // mostrador vacío

    mostr_vacio.signal();
}

```

```
}
```

```
int producir_ingrediente()
```

```
{
```

```
    chrono::milliseconds duracion_produ( aleatorio<10,100>() );
```

```
    cout << "Estanquero : empieza a producir ingrediente (" << duracion_produ.count()  
<< " milisegundos)" << endl;
```

```
    this_thread::sleep_for( duracion_produ );
```

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

```
    cout << "Estanquero : termina de producir ingrediente " << num_ingrediente << endl;
```

```
    return num_ingrediente ;
```

```
}
```

```
void funcion_hebra_estanquero(MRef<Estanco> Estanco)
```

```
{
```

```
    while ( true ) {
```

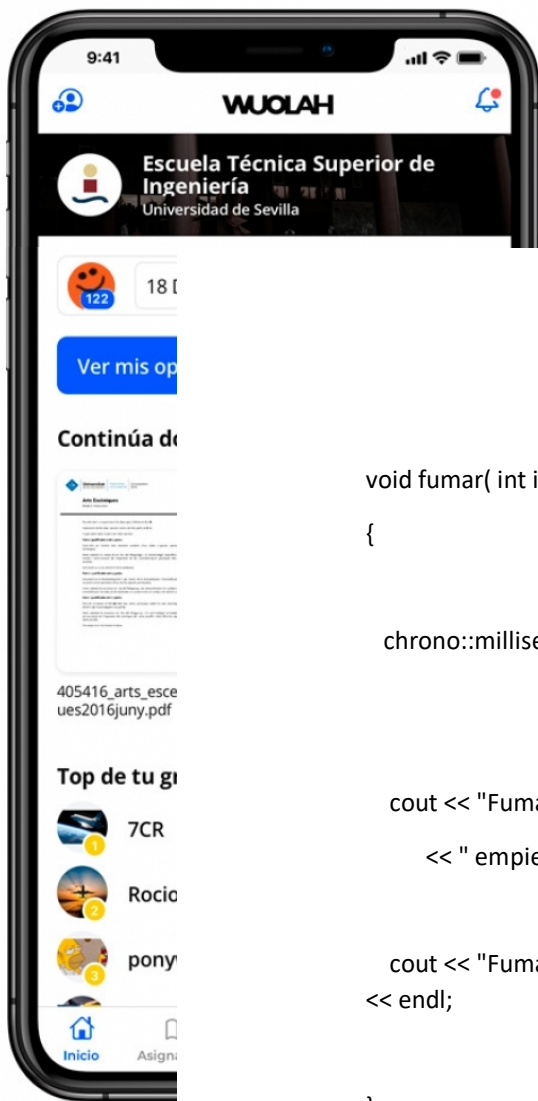
```
        int i = producir_ingrediente();
```

```
        Estanco->PonerIngrediente(i);
```

```
        Estanco->esperarRecogidaIngrediente();
```

```
    }
```

```
}
```



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



```
void fumar( int i )
{

    chrono::milliseconds duracion_fumar( aleatorio<20,200>() );

    cout << "Fumador " << i << " : "
        << " empieza a fumar ( " << duracion_fumar.count() << " milisegundos)" << endl;

    cout << "Fumador " << i << " : termina de fumar, comienza espera de ingrediente."
    << endl;

}
```

```
void funcion_hebra_fumador( int i, MRef<Estanco> Estanco)
{
    while( true )
    {
        Estanco->obtenerIngrediente (i);

        fumar (i);
    }
}
```

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

```
int main()
{
    MRef<Estanco> estanco = Create<Estanco>();
```

WUOLAH


```
thread hebra_estanquero ( funcion_hebra_estanquero, estanco );  
thread hebras_fumadores[ingredientes];  
  
for (int i = 0; i < ingredientes; i++)  
    hebras_fumadores[i] = thread( funcion_hebra_fumador, i , estanco);  
  
hebra_estanquero.join();  
  
for (int i = 0; i < ingredientes; i++)  
    hebras_fumadores[i].join();  
}
```