FumadorMonitor.pdf

Jrg14

Sistemas Concurrentes y Distribuidos

2º Grado en Ingeniería Informática

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

SCD Fumador con monitor

```
#include <iostream>
#include <iomanip>
#include <random>
#include <thread>
#include "HoareMonitor.h"

using namespace std;
using namespace HM;

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 Fumar( int num_fumador ){

    // calcular milisegundos aleatorios de duraciÃ³n de la acciÃ³n de fumar
    chrono::milliseconds duracion_fumar( aleatorio<20,200>() );

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

    // espera bloqueada por un tiempo duracion_fumar
    this_thread::sleep_for( duracion_fumar );

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

int ProducirIngrediente(){

    chrono::milliseconds tiempo( aleatorio<20,200>() );
    this_thread::sleep_for(tiempo);

    return aleatorio<0,2>();
}

class EstancoSU : public HoareMonitor{
```

```
	private:

		int mostrador;
		CondVar estanquero, fumador[3];			// colas de espera y
seÃ±al
	public:
		EstancoSU();
		void ObtenerIngrediente(int i);
		void PonerIngrediente(int j);
		void EsperarRecogida();
} ;

// constructor
EstancoSU::EstancoSU(){
mostrador = -1;
estanquero = newCondVar();
for(int i = 0; i < 3; i++)
	fumador[i] = newCondVar();
}

// fumador espera su ingrediente, lo retira del mostrador y avisa al estanquero que el
mostrador esta vacio
void EstancoSU::ObtenerIngrediente(int i){

	if(mostrador != i)
		fumador[i].wait();
	mostrador = -1;

	cout << "Ingrediente " << i << " recogido." << endl;

	estanquero.signal();
}

// pone el ingrediente en el mostrador y avisa al fumador que necesita el ingrediente i
void EstancoSU::PonerIngrediente(int i){

	mostrador = i;

	cout << "Ingrediente " << i << " puesto en el mostrador." << endl;

	fumador[i].signal();
}
// estanquero espera a que algÃºn fumador retire el ingrediente del mostrador
void EstancoSU::EsperarRecogida()
{
```

```
        if(mostrador != -1)
        estanquero.wait();
}

void funcion_estanquera(MRef<EstancoSU> monitor){
        int ingr;
        while(true){

                ingr = ProducirIngrediente();

                monitor->PonerIngrediente(ingr);

                monitor->EsperarRecogida();
        }
}

void funcion_fumadora(MRef<EstancoSU> monitor, int i){
        while(true){

                monitor->ObtenerIngrediente(i);
                Fumar(i);
        }
}

int main(){

        MRef<EstancoSU> monitor = Create<EstancoSU>();

        thread hebra_estanquero(funcion_estanquera, monitor);

        thread hebra_fumadores[3];
        for(int i = 0; i < 3; i++)
                hebra_fumadores[i]= thread(funcion_fumadora, monitor, i);

        hebra_estanquero.join();

        for(int i = 0; i < 3; i++)
                hebra_fumadores[i].join();
}
```