

barberosimple.pdf



Cooper_3



Sistemas Concurrentes y Distribuidos



2º Grado en Ingeniería Informática

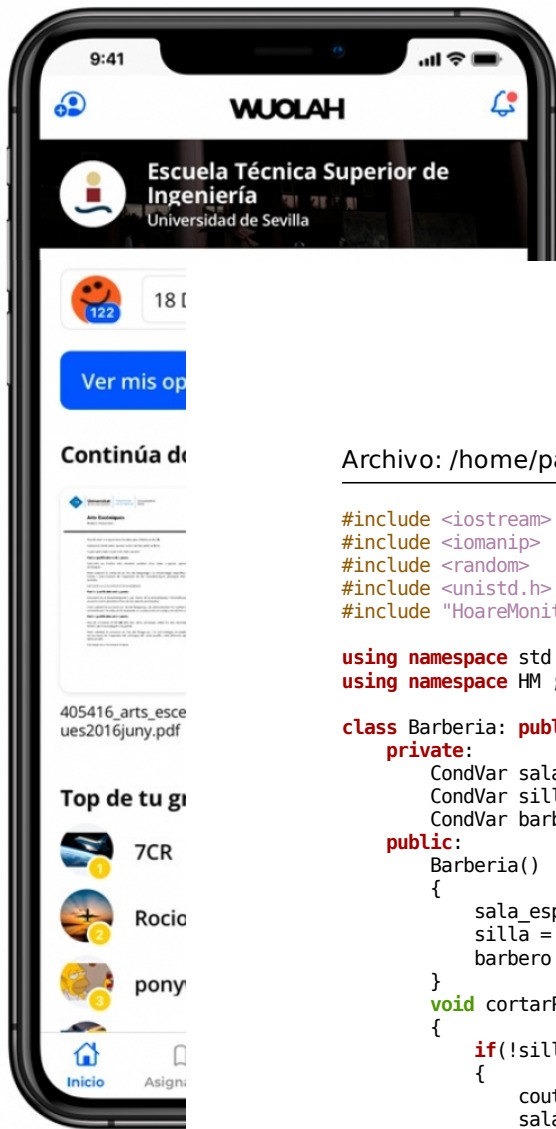


Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada



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





Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.



Archivo: /home/pablo/Documents/scd_pr...d/monitores/barbero_sir... de 2

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

using namespace std ;
using namespace HM ;

class Barberia: public HoareMonitor{
private:
    CondVar sala_espera;
    CondVar silla;
    CondVar barbero;
public:
    Barberia()
    {
        sala_espera = newCondVar();
        silla = newCondVar();
        barbero = newCondVar();
    }
    void cortarPelo(int i)
    {
        if(!silla.empty())
        {
            cout << "\nEl cliente "<<i<< " a la sala de espera.";
            sala_espera.wait();
        }
        cout<<"\nAvisando al barbero";
        barbero.signal();
        cout<<"\nEn la silla se esta pelando al cliente"<<i;
        silla.wait();

    }
    void finCliente()
    {
        cout<<"\nEl barbero ha terminado de pelar.";
        silla.signal();
    }
    void siguienteCliente()
    {
        if (silla.empty())
        {
            if(!sala_espera.empty())
            {
                cout<<"\nEl barbero llama a un cliente de la sala de espera.";
                sala_espera.signal();
            }
            else
            {
                cout<<"\nEl barbero espera dormido a que haya clientes.";
                barbero.wait();
            }
        }
    }
};

void funcion_hebra_barbero(MRef <Barberia> barberia)
{
    while(1)
    {
        barberia->siguienteCliente();
        cout<<"\nCortando el pelo";
        sleep(rand()%3); //Cortar pelo
        barberia->finCliente();
    }
}
```

```

void funcion_hebra_cliente(MRef <Barberia> barberia, int i)
{
    while(1)
    {
        barberia->cortarPelo(i);
        cout<<"\nAl cliente "<<i<<" le esta creciendo el pelo.";
        sleep(rand()%3);
    }
}

int main()
{
    MRef<Barberia> monitor = Create<Barberia>();
    thread hebra_cliente(funcion_hebra_cliente, monitor,1),
        hebra_barbero(funcion_hebra_barbero, monitor);

    hebra_cliente.join();
    hebra_barbero.join();
    return(0);
}

```