

WUOLAH



PablaO

www.wuolah.com/student/PablaO



235

filosofos_camarero.pdf

Practica 3-Ejercicios



2º Sistemas Concurrentes y Distribuidos



Grado en Ingeniería Informática



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

```

#include <iostream>
#include <time.h>      // incluye "time"
#include <unistd.h>    // incluye "usleep"
#include <stdlib.h>    // incluye "rand" y "srand"
#include <mpi.h>

using namespace std;

void Filosofo( int id, int nprocesos);
void Tenedor ( int id, int nprocesos);
void Camarero( int id, int nprocesos);

// -----

int main( int argc, char** argv )
{
    int rank, size;

    srand(time(0));
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );

    if( size!= 11)
    {
        if( rank == 0)
            cout << "El numero de procesos debe ser 11" << endl << flush ;
        MPI_Finalize( );
        return 0;
    }

    if (rank == 10)
        Camarero(rank, size);
    else if (rank != 10 && (rank%2) == 0)
        Filosofo(rank,size); // Los pares son Filósofos
    else
        Tenedor(rank,size); // Los impares son Tenedores

    MPI_Finalize( );

    return 0;
}
// -----

void Filosofo( int id, int nprocesos )
{
    int izq = (id+1) % nprocesos;
    int der = ((id+nprocesos)-1) % nprocesos;
    int cam = 10;
    int soltar_tenedor_izq = 0, soltar_tenedor_der = 1;

    while(1)
    {
        // Pregunta al camarero si puede sentarse
        cout << "Filosofo " << id << " solicita al camarero poder sentarse
en la mesa." << endl;
        MPI_Ssend(&id, 1, MPI_INT, cam, 0, MPI_COMM_WORLD);
        cout << "Filósofo " << id << " se sienta en la mesa." << endl <<
flush;

        // Solicita tenedor izquierdo
        cout << "Filosofo " << id << " solicita tenedor izq ..." << izq << endl
<< flush;
        MPI_Ssend(&id, 1, MPI_INT, izq, 0, MPI_COMM_WORLD);

```

```
// Solicita tenedor derecho
cout << "Filosofo " << id << " coge tenedor der ..." << der << endl <<
flush;
MPI_Ssend(&id, 1, MPI_INT, der, 0, MPI_COMM_WORLD);

cout << "Filosofo " << id << " COMIENDO" << endl << flush;
sleep((rand() % 3)+1); //comiendo

// Suelta el tenedor izquierdo
cout << "Filosofo " << id << " suelta tenedor izq ..." << izq << endl
<< flush;
MPI_Ssend(&id, 1, MPI_INT, izq, soltar_tenedor_izq, MPI_COMM_WORLD);

// Suelta el tenedor derecho
cout << "Filosofo " << id << " suelta tenedor der ..." << der << endl
<< flush;
MPI_Ssend(&id, 1, MPI_INT, der, soltar_tenedor_der, MPI_COMM_WORLD);

// Piensa (espera bloqueada aleatorio del proceso)
cout << "Filosofo " << id << " PENSANDO" << endl << flush;

// espera bloqueado durante un intervalo de tiempo aleatorio
// (entre una décima de segundo y un segundo)
usleep( 1000U * (100U+(rand()%900U)) );

// Deja la mesa
MPI_Ssend(&id, 1, MPI_INT, cam, 0, MPI_COMM_WORLD);
cout << "Filósofo " << id << " deja la mesa." << endl << flush;
}
}
// -----

void Tenedor(int id, int nprocesos)
{
    int buf;
    MPI_Status status;
    int Filo;

    while( true )
    {
        // Espera un peticion desde cualquier filosofo vecino ...
        // ...

        // Recibe la peticion del filosofo ...
        MPI_Recv(&Filo, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);

        cout << "Ten. " << id << " recibe petic. de " << Filo << endl <<
flush;

        // Espera a que el filosofo suelte el tenedor...
        MPI_Recv(&Filo, 1, MPI_INT, status.MPI_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
        cout << "Ten. " << id << " recibe liberac. de " << status.MPI_SOURCE
<< endl << flush;

    }
}

void Camarero(int id, int nprocesos) {
    int Filo;
    int sentados = 0;
    int flag;
```

```

    MPI_Status status;

    while (1) {
        MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &flag,
&status);

        if(sentados < 4) {
            //Espera a que llegue un filósofo para sentarse
            MPI_Recv(&Filo, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
            sentados++;

            //Espera a que un filósofo se vaya
            MPI_Recv(&Filo, 1, MPI_INT, status.MPI_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
            sentados--;
        }
    }
}
// -----

```