Practica 2 - MPI

Ampliación de sistemas operativos

Doble grado en Ing. Informática + Ing. Computadores

Javier García Borrego

Contenido

Introducción:

La practica 2 consiste en crear un parking que se puedan aparcar coches y camiones los cuales serán procesados por separado y se comunicaran con la librería open-MPI.

No se ha realizado el apartado de tener múltiples plantas.

Explicación del código Parking.c:

Parking es el encargado de gestionar la entrada/salida de los vehículos así que lo primero que se hace en el código es inicializar las variables que vamos a necesitar que son las siguientes:

```
int rank, nprocs,plazas,plazas_libres,identificador;
int esclavo[3] = {0,0,0};//id , entrar salir, plaza a ocupar si esta aparcados
int informacion[3] ={0,0,-1};
```

Rank: es el id del proceso.

Nprocs: es el número de procesos.

Plazas: es el número de plazas totales.

Plazas_libres: son el numero de plazas libres que quedan.

Identificador: es para guardar el tag que se recibe (0 si es coche o 1 si es camión)

Esclavo: es el mensaje que recibimos del proceso esclavo

Información: es el mensaje que enviamos al proceso esclavo

Luego procedemos a iniciar el entono MPI:

```
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Barrier(MPI_COMM_WORLD);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

inicialización parecida a los ejemplos.

```
plazas = atoi(argv[1]);
plazas_libres = plazas;
int aparcamiento[plazas];

int i;
for(i=0;i<plazas;i++){
        aparcamiento[i]=0;//0 significa que no hay nadie ocupandola
}
```

Se cogen el numero de plazas que se pasa como argumentos y se ponen las plazas libres iguales a las plazas totales además de que inicializamos todas las plazas a 0.

```
while(1){
        //hay que esperar hasta que alguien quiera aparcar
        MPI_Recv(&esclavo,3,MPI_INT,MPI_ANY_SOURCE,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
        identificador = status.MPI_TAG;
        informacion[0]=esclavo[0];
        informacion[1]=esclavo[1];// Queremos saber si quiere entrar o salir
        if(esclavo[1]==0){//Quiere entrar

                if(identificador == 0){//es un coche

                        for(i =0;i<plazas;i++){

                                if (aparcamiento[i]==0){

                                        aparcamiento[i]=esclavo[0];
                                        plazas_libres--;
                                        informacion[2]=i;
                                        informacion[1]=1;       //plaza ocupada
                                        printf("ENTRADA: Coche %d aparca en %d. Plazas libres %d\n",esclavo[0],i,plazas_libres);
                                        estadoParking(aparcamiento,plazas);
                                        break;
                                }
                        }

                        MPI_Send(&informacion,3,MPI_INT,informacion[0],0,MPI_COMM_WORLD);

                }else{//camion

                        for(i =0;i<plazas;i++){

                                if (aparcamiento[i]==0 && aparcamiento[i+1]==0){

                                        aparcamiento[i]=esclavo[0];
                                        aparcamiento[i+1]=esclavo[0];
                                        plazas_libres=plazas_libres-2;
                                        informacion[2]=i;
                                        informacion[1]=1;       //plaza ocupada
                                        printf("ENTRADA: Camion %d aparca en %d y %d. Plazas libres %d\n",esclavo[0],i,i+1,plazas_libres);
                                        estadoParking(aparcamiento,plazas);
                                        break;
                                }
                        }

                        MPI_Send(&informacion,3,MPI_INT,informacion[0],1,MPI_COMM_WORLD);
                }

        }else{//Quiere salir
```

Entramos en bucle while que no parara hasta que hagamos ctrl+C y terminemos el proceso luego hacemos MPI_Recv que se quedará esperando hasta que un coche o camión le envíen un mensaje cuando reciba un mensaje en identificador se guardará la TAG que será 0 si es un coche o 1 si es un camión a parte guardamos en información su id en la posición 0 y si quieres entrar o salir (0 o 1) en la posición 1.

Si el coche o camión quieren entrar se identificará si es un coche o un camión y se mirara si hay plazas disponibles si las hay se realizaran las operaciones necesarias para poner en el aparcamiento el numero del coche en la plaza adecuada y dejar las plazas disponibles además de cambiar la posición 1 del array información a 1 para saber que ha dicho coche ha aparcado y su próxima instrucción será que quiere salir al final se manda el mensaje información a coche o camión.

```
        }else{//Quiere salir

                if(identificador == 0){//es un coche

                        for(i=0;i<plazas;i++){
                                if (i==esclavo[2]){

                                        aparcamiento[i]=0;
                                        plazas_libres++;
                                        informacion[1]=0;
                                        printf("Salida: Coche %d saliendo de la plaza %d. Plazas libres %d\n",esclavo[0],i,plazas_libres);
                                        estadoParking(aparcamiento,plazas);
                                        break;
                                }
                        }

                        MPI_Send(&informacion,3,MPI_INT,informacion[0],0,MPI_COMM_WORLD);
                }else{//camion

                        for(i=0;i<plazas;i++){

                                if (i==esclavo[2]){

                                        aparcamiento[i]=0;
                                        aparcamiento[i+1]=0;
                                        plazas_libres=plazas_libres+2;
                                        informacion[1]=0;
                                        printf("Salida: Camion %d saliendo de la plaza %d y %d. Plazas libres %d\n",esclavo[0],i,i+1,plazas_libres);
                                        estadoParking(aparcamiento,plazas);
                                        break;
                                }
                        }

                        MPI_Send(&informacion,3,MPI_INT,informacion[0],1,MPI_COMM_WORLD);
                }

        }
}

MPI_Finalize();
return 0;
```

En caso de salir se realizara lo mismo la única diferencia en las operaciones que se realizaran dependiendo de si es un coche o un camion ya que uno ocupa una plaza solo y otro ocupa 2.

Explicación de coche.c y camión.c:

Se inician las variables a usar:

```
int id, nprocs;
int aparcado = 0;
int informacion[3];//id y si quiere entrar y salir y plaza ocupada
int respuesta[3];//en la primera posicion tra la id del coche y en la segunda tra si se ha aparcado o no 0 = no aparacado y 1= se ha aparcado y la plaza en la que se ha aparcado
```

Id: es el id del proceso.

Nprocs: el numero de procesos.

Aparcado: nos dice si el coche o camion ha sido aparcado.

Información: es la información que pasamos al proceso principal.

Respuesta: es la respuesta que recibimos del proceso principal.

Luego iniciamos el entorno MPI igual que en parking.c:

```
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Barrier(MPI_COMM_WORLD);
MPI_Comm_rank(MPI_COMM_WORLD, &id);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
```

Entramos en el while:

```
while(1){

        informacion[0]=id;
        informacion[1]=0;//Entrar = 0 porque quiere aparcar y salir = 1 porque ya ha aparcado
        informacion[2]=-1;//ninguna plaza ocupada

        do{
                MPI_Send(&informacion,3,MPI_INT,0,1,MPI_COMM_WORLD);//el segundo 0 es la TAG que 0 significa coche y 1 camion
                MPI_Recv(&respuesta,3,MPI_INT,MPI_ANY_SOURCE,1,MPI_COMM_WORLD,&status);
                aparcado = respuesta[1];

        }while(aparcado==0);

        sleep(rand()%21);//Dormir entre 0 y 20


        MPI_Send(&respuesta,3,MPI_INT,0,1,MPI_COMM_WORLD);
        MPI_Recv(&respuesta,3,MPI_INT,MPI_ANY_SOURCE,1,MPI_COMM_WORLD,&status);

}
MPI_Finalize();
return 0;
```

Ponemos valores al mensaje información con su id si quiere entrar o salir y la plaza ocupada que en este caso es ninguna y entramos en un bucle do while que solamente saldremos de el cuando la posición 1 de respuesta sea 1 ya que quiere decir que se ha aparcado si no volveremos otra vez a hacer el bucle y así hasta encontrar sitio.

Cuando hemos encontrado sitio hacemos que el proceso se duerma un tiempo aleatorio comprendido entre 0 y 20 (un valor alto para ver si se realizan interbloqueos o inanición).

Para finalizar mandamos que queremos salir y ponemos un MPI_Recv para que el proceso espere a la respuesta y pida volver a entrar antes de tiempo.

La diferencia entre camion.c y coche.c es la siguiente:

Camion.c:

```
MPI_Send(&informacion,3,MPI_INT,0,1,MPI_COMM_WORLD);//el segundo 0 es la TAG que 0 significa coche y 1 camion
MPI_Recv(&respuesta,3,MPI_INT,MPI_ANY_SOURCE,1,MPI_COMM_WORLD,&status);
```

Coche.c:

```
MPI_Send(&informacion,3,MPI_INT,0,0,MPI_COMM_WORLD);//el segundo 0 es la TAG que 0 significa coche y 1 camion
MPI_Recv(&respuesta,3,MPI_INT,MPI_ANY_SOURCE,0,MPI_COMM_WORLD,&status);
```

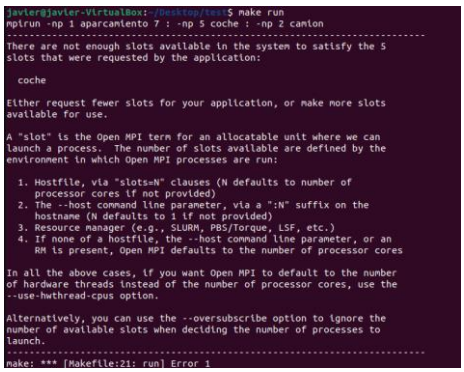
En la parte en la que iría la TAG se pone 0 o 1 dependiendo de si es un coche o un camion correspondiente.

Makefile:

Se ha realizado un makefile para una ejecución de la practica el cual es el siguiente:

```
#
# Makefile
#       Makefile global del sistema
#
all: aparcamiento coche camion

aparcamiento: aparcamiento.c

        mpicc aparcamiento.c -o aparcamiento

coche: coche.c

        mpicc coche.c -o coche

camion: camion.c

        mpicc camion.c -o camion

run: aparcamiento coche camion

        mpirun --hostfile localhost -np 1 aparcamiento 7 : -np 5 coche : -np 2 camion

clean:
        rm aparcamiento
        rm coche
        rm camion
```

El localhost es un txt en el que pongo slots para poder usar la aplicación ya que si no saltaba este error:



Ejemplos de resultados:

7 plazas 5 coches y 2 camiones:

```
javier@javier-VirtualBox:~/Desktop/test$ make run
mpirun --hostfile localhost -np 1 aparcamiento 7 : -np 5 coche : -np 2 camion
ENTRADA: Coche 1 aparca en 0. Plazas libres 6
Parking: [1] [0] [0] [0] [0] [0] [0]
ENTRADA: Coche 2 aparca en 1. Plazas libres 5
Parking: [1] [2] [0] [0] [0] [0] [0]
ENTRADA: Coche 3 aparca en 2. Plazas libres 4
Parking: [1] [2] [3] [0] [0] [0] [0]
ENTRADA: Coche 4 aparca en 3. Plazas libres 3
Parking: [1] [2] [3] [4] [0] [0] [0]
ENTRADA: Coche 5 aparca en 4. Plazas libres 2
Parking: [1] [2] [3] [4] [5] [0] [0]
ENTRADA: Camion 6 aparca en 5 y 6. Plazas libres 0
Parking: [1] [2] [3] [4] [5] [6] [6]
Salida: Camion 6 saliendo de la plaza 5 y 6. Plazas libres 2
Parking: [1] [2] [3] [4] [5] [0] [0]
ENTRADA: Camion 7 aparca en 5 y 6. Plazas libres 0
Parking: [1] [2] [3] [4] [5] [7] [7]
^CSalida: Camion 7 saliendo de la plaza 5 y 6. Plazas libres 2
Parking: [1] [2] [3] [4] [5] [0] [0]
ENTRADA: Camion 6 aparca en 5 y 6. Plazas libres 0
Parking: [1] [2] [3] [4] [5] [6] [6]
```

5 plazas, 8 coches y 4 camiones:

```
javier@javier-VirtualBox:~/Desktop/test$ make run
mpirun --hostfile localhost -np 1 aparcamiento 5 : -np 8 coche : -np 4 camion
ENTRADA: Coche 1 aparca en 0. Plazas libres 4
Parking: [1] [0] [0] [0] [0]
ENTRADA: Coche 2 aparca en 1. Plazas libres 3
Parking: [1] [2] [0] [0] [0]
ENTRADA: Coche 3 aparca en 2. Plazas libres 2
Parking: [1] [2] [3] [0] [0]
ENTRADA: Coche 4 aparca en 3. Plazas libres 1
Parking: [1] [2] [3] [4] [0]
ENTRADA: Coche 5 aparca en 4. Plazas libres 0
Parking: [1] [2] [3] [4] [5]
Salida: Coche 1 saliendo de la plaza 0. Plazas libres 1
Parking: [0] [2] [3] [4] [5]
Salida: Coche 2 saliendo de la plaza 1. Plazas libres 2
Parking: [0] [0] [3] [4] [5]
Salida: Coche 4 saliendo de la plaza 3. Plazas libres 3
Parking: [0] [0] [3] [0] [5]
Salida: Coche 5 saliendo de la plaza 4. Plazas libres 4
Parking: [0] [0] [3] [0] [0]
ENTRADA: Coche 7 aparca en 0. Plazas libres 3
Parking: [7] [0] [3] [0] [0]
ENTRADA: Coche 8 aparca en 1. Plazas libres 2
Parking: [7] [8] [3] [0] [0]
ENTRADA: Camion 9 aparca en 3 y 4. Plazas libres 0
Parking: [7] [8] [3] [9] [9]
Salida: Coche 3 saliendo de la plaza 2. Plazas libres 1
Parking: [7] [8] [0] [9] [9]
ENTRADA: Coche 4 aparca en 2. Plazas libres 0
Parking: [7] [8] [4] [9] [9]
Salida: Camion 9 saliendo de la plaza 3 y 4. Plazas libres 2
Parking: [7] [8] [4] [0] [0]
ENTRADA: Coche 1 aparca en 3. Plazas libres 1
Parking: [7] [8] [4] [1] [0]
ENTRADA: Coche 2 aparca en 4. Plazas libres 0
Parking: [7] [8] [4] [1] [2]
```

Como se puede ver no se producen interbloqueos en ninguno de los 2 ejemplos.