Universidad del valle de Guatemala
Departamento de ciencias de la computación
Programación paralela y distribuida
Miguel Novella

Andres Quinto 18288

Hoja de trabajo 2 MPICC

Ejecución normal:

```
compilation terminated.
[imag1ne@nobara-pc HT2]$ mpicc mpiHello.c -o mpiHello
[imag1ne@nobara-pc HT2]$ ./mpiHello
Hello from process 0 of 1
[imag1ne@nobara-pc HT2]$
```

Con el comando -np 4

```
[imag1ne@nobara-pc HT2]$ mpirun -np 4 ./mpiHello
Hello from process 3 of 4
Hello from process 1 of 4
Hello from process 2 of 4
Hello from process 0 of 4
[imag1ne@nobara-pc HT2]$
```

Ejecución con la parte 3 modificada:

```
[imag1ne@nobara-pc HT2]$ mpirun -np 4 ./Modifided
0 sent Hello World
1 received Hello World
2 received Hello World
3 received Hello World
[imag1ne@nobara-pc HT2]$
```

Observaciones del original vs el cambio en la parte 3:
La modificación consiste en agregar una sección condicional para verificar si el rango actual es cero. Si es cero, se crea un mensaje "Hello World" y se envía a todos los procesos diferentes de cero utilizando un bucle for y la función MPI_Send. Si el rango actual es diferente de cero, se espera a recibir el mensaje con la función MPI_Recv y luego se imprime el mensaje recibido junto con el rango del proceso que lo recibió.

Mensajes con bloque y correspondencia:

```
[imag1ne@nobara-pc HT2]$ mpirun -np 4 ./Modifided
0 sent Hello Andres Emilio Quinto 18288
2 sent Hello Andres Emilio Quinto 18288
3 received Hello Andres Emilio Quinto 18288
1 received Hello Andres Emilio Quinto 18288
[imag1ne@nobara-pc HT2]$
```

Siguiendo el formato adecuado:

```
[imag1ne@nobara-pc HT2]$ mpirun -np 4 ./Modifided
0 sent Hello Andres Emilio Quinto 18288
1 received Hello Andres Emilio Quinto 18288
2 sent Hello Andres Emilio Quinto 18288
3 received Hello Andres Emilio Quinto 18288
[imag1ne@nobara-pc HT2]$
```

Comentarios adicionales:
Este código está utilizando MPI (Message Passing Interface) para realizar comunicación entre procesos en un ambiente de programación distribuida. El programa crea varios procesos que ejecutan el mismo código, pero cada uno tiene un rango único dentro del comunicador MPI_COMM_WORLD.

En la sección de distribución de trabajo del programa, se utiliza una condición if para enviar mensajes desde los procesos pares al proceso siguiente en el rango, y recibir mensajes en los procesos impares del proceso anterior en el rango. En la función MPI_Send, el mensaje se envía con el tag MESSTAG y se especifica el tipo de dato MPI_CHAR. En la función MPI_Recv, se espera el mensaje con el tag MESSTAG y se especifica el tipo de dato MPI_CHAR.

Este código es un ejemplo simple de cómo se puede utilizar MPI para enviar y recibir mensajes entre procesos en un entorno distribuido.