**PASAJE DE MENSAJES USANDO MPI**

En este ejemplo, estamos ejecutando dos procesos MPI. El proceso 0 (proceso maestro) envía un arreglo de enteros al proceso 1 (proceso esclavo) utilizando las funciones **Send** y **Recv** proporcionadas por la API de MPJ Express.
El Manejo de Mensajes MPI (Message Passing Interface) en Java se puede realizar utilizando la biblioteca MPJ Express (Message Passing in Java Express). MPJ Express proporciona una API para programación paralela y distribuida en Java utilizando el estándar MPI. A continuación, se muestra un ejemplo simple de cómo enviar y recibir mensajes utilizando MPJ Express:

Antes de usar, chequea de tener MPJ Express instalado en tu IDE antes de ejecutar este ejemplo.

**EXPLICACION 1:**

Crear un archivo Java llamado MPJExample.java con el siguiente código:

```
import mpi.*;

public class MPJExample {
   public static void main(String[] args) throws MPIException {
      MPI.Init(args);

      int rank = MPI.COMM_WORLD.Rank();
      int size = MPI.COMM_WORLD.Size();

      if (rank == 0) {
         // Proceso maestro
         int[] dataToSend = {1, 2, 3, 4, 5};
         MPI.COMM_WORLD.Send(dataToSend, 0, dataToSend.length, MPI.INT, 1, 0);
         System.out.println("Proceso " + rank + " envió datos al proceso 1.");
      } else if (rank == 1) {
         // Proceso esclavo
         int[] receivedData = new int[5];
         MPI.COMM_WORLD.Recv(receivedData, 0, receivedData.length, MPI.INT, 0, 0);
         System.out.print("Proceso " + rank + " recibió datos: ");
         for (int i : receivedData) {
            System.out.print(i + " ");
         }
         System.out.println();
      }

      MPI.Finalize();
   }
}
```

Realizar

1. Compila el código utilizando el comando *mpijavac*: **mpijavac MPJExample.java**
2. Ejecuta el programa utilizando el comando *mpirun*: **mpirun -np 2 java MPJExample**

Este ejemplo ayuda a comprender cómo manejar mensajes MPI en Java utilizando MPJ Express. Se puede personalizar y ampliar este código según las necesidades específicas en aplicaciones distribuidas y paralelas que desee realizar el alumno

**EXPLICACION 2:**

Programa MPI en Java que envía un mensaje de una palabra de longitud desde el proceso maestro a todos los procesos esclavos. El proceso esclavo que recibe el mensaje debe imprimirlo en la consola.

```
import mpi.MPI;

public class Ejercicio2 {

   public static void main(String[] args) throws Exception {
      // Inicializa MPI
      MPI.Init(args);

      // Obtén el rango del proceso
      int rank = MPI.COMM_WORLD.Rank();

      // Si el proceso es el maestro
      if (rank == 0) {
         // Crea un número entero
         int numero = 1234;

         // Envía el número a todos los procesos esclavos
         MPI.COMM_WORLD.Bcast(numero, 0, MPI.INT, MPI.COMM_WORLD.Get_size() - 1);
      } else {
         // Recibe el número del maestro
         int numero = MPI.COMM_WORLD.Bcast(null, 0, MPI.INT, 0,
MPI.COMM_WORLD.Get_size() - 1);

         // Imprime el número
         System.out.println("Proceso esclavo " + rank + ": " + numero);
      }

      // Finaliza MPI
      MPI.Finalize();
   }
}
```

**EXPLICACION 3:**

Programa MPI en Java que envía un vector de números enteros desde el proceso maestro a todos los procesos esclavos. El proceso esclavo que recibe el vector debe imprimirlo en la consola

```
import mpi.MPI;

public class Ejercicio3 {

   public static void main(String[] args) throws Exception {
      // Inicializa MPI
      MPI.Init(args);

      // Obtén el rango del proceso
      int rank = MPI.COMM_WORLD.Rank();

      // Si el proceso es el maestro
      if (rank == 0) {
         // Crea un vector de números enteros
         int[] vector = {1, 2, 3, 4, 5};

         // Envía el vector a todos los procesos esclavos
         MPI.COMM_WORLD.Bcast(vector, 0, vector.length, MPI.INT,
MPI.COMM_WORLD.Get_size() - 1);
      } else {
         // Recibe el vector del maestro
         int[] vector = new int[5];
         MPI.COMM_WORLD.Bcast(vector, 0, vector.length, MPI.INT, 0,
MPI.COMM_WORLD.Get_size() - 1);

         // Imprime el vector
         System.out.println("Proceso esclavo " + rank + ": " + Arrays.toString(vector));
      }

      // Finaliza MPI
      MPI.Finalize();
   }
}
```

EJERCICIOS GRUPALES

Nota: Asegúrate de tener MPJ Express instalado en tu sistema antes de intentar estos ejercicios.

1. Comunicación Simple:

Crea un programa en Java utilizando MPJ Express que envíe un mensaje desde un proceso maestro (rango 0) a un proceso esclavo (rango 1). El mensaje puede ser un simple saludo.

2. Comunicación Bidireccional:

Modifica el ejercicio anterior para que el proceso esclavo también envíe un mensaje de respuesta al proceso maestro.

3. Comunicación en Anillo:

Crea un programa MPI en el que múltiples procesos se comuniquen en un patrón de anillo. Cada proceso debe recibir un mensaje de su proceso vecino y luego pasar el mensaje al siguiente proceso en el anillo hasta que vuelva al proceso original.

4. Suma de Arreglos Distribuida:

Divide un arreglo grande en varios fragmentos y distribuye estos fragmentos entre los procesos MPI. Luego, cada proceso suma su fragmento y envía el resultado al proceso maestro. El proceso maestro debe calcular la suma total.

5. Producto Escalar Distribuido:

Divide dos arreglos en fragmentos y distribuye estos fragmentos entre los procesos MPI. Cada proceso debe calcular el producto escalar de su fragmento y enviar el resultado al proceso maestro. El proceso maestro debe calcular el producto escalar total.