Assignment 19. MPI. Dynamic process control. Client-server communication.

To complete the task, you need to create and compile two programs: server and client. In one window of the SSH client, a server is launched for one process, which gives out the port name.

An example of a command to start the server: mpiexec -n 1 ./serv.o

Then the client is launched in another window, specifying the port name separated by a space in single quotes (example command: mpiexec -n 1 ./client.o 'port name').

Understand the new functions in Assignment19_serv.c and Assignment19_client.c and explain programs execution.

Check the work by running the server and the client. Add the program and send an arbitrary message to each other.

The server should display the following messages:

Port name

Waiting for the client ...

Client connected

Server sent value: 25

Server got value: 42

The client should display the following messages:

Attempt to connect

Server connection

Client sent value: 42

Client got value: 25

In order to display these messages correctly, you need to understand what happens at each step.

Description of functions for working with server and client:

MPI_Open_port (MPI_Info **info**, char ***port_name**) Sets the port for the server on which the client can access it.

Input parameters

info - information on how to set the port for **MPI_Comm_accept** (by default MPI_**INFO**_NULL).

Output parameters

port_name - The newly installed port (string). The port name is the network address. It is unique to the communication space to which it belongs (implementation-defined) and can be used by any client in the communication space.

MPI_Open_port sets the network address, copying it into the **port_name** string, at which the server is able to accept connections from clients. **port_name** is supported by the system, possibly using the information in the info argument.

MPI copies the port name supported by the system to **port_name**. The **port_name** argument specifies the newly opened port and can be used by the client to contact the server. **MPI_MAX_PORT_NAME** defines the maximum string size that can be supported by the system.

The port name can be reused after it has been released through **MPI_CLOSE_PORT** and released by the system.

MPI_Comm_accept (char * port_**name**, MPI_Info **info**, int **root**, MPI_Comm **comm**, MPI_Comm ***newcomm**)

to accept a client connection.

IN **port_name** Port name (string, used by root only)

IN **info** Implementation specific info (descriptor, used by root only)

IN **root** Rank in comm for root (integer)

IN **comm** Intracommunicator within which a collective call is made (handle)

OUT **newcomm** Intercommunicator with client as remote group (handle)

MPI_COMM_ACCEPT establishes a client connection. This is a collective operation by means of a calling communicator. She returns an intercommunicator, allowing communication with the client.

port_name must be set through the call to **MPI_OPEN_PORT**.

The info argument is an implementation-defined string that allows fine control over the call to **MPI_COMM_ACCEPT**.

MPI_COMM_ACCEPT is a blocking call. The user can implement non-blocking access by placing **MPI_COMM_ACCEPT** on a separate thread.

MPI_Comm_free (MPI_Comm * **comm**) Communicator destruction function

MPI_Close_port (char * **port_name**) This function releases the network address represented by **port_name**.