TSN3151: Laboratory 03

The main purpose of this laboratory is to provide more exercises on the basics of MPI programming and for those who joined the course late, some additional time to catch up in the laboratory work.

This session extends from the previous session on basic MPI programming.

Guidelines:

- You are encouraged to conduct the laboratory exercises and assist each other in groups **not** larger than 4 people. This is to ensure that the laboratory sessions do not get distractingly noisy. In general, I **will not** be demonstrating or lecturing during the laboratory session but will be around to help you. Please do not be shy to ask for help.

Part 1: Send & Receive

As I mentioned last week during the tutorial, the program from last week actually commits a major MPI faux-pas: every process writes to the output. This worked OK in your case because you were running all the processes on the same computer. In real life, the main purpose for running MPI is if you have a cluster of computers in a local area network, not a single computer. (If you had a single computer with many processors, you would use OpenMP instead – we will learn about OpenMP later in this course)

So, how do we solve the problem of the `printf`s coming from many processes?

1. Instead of every process printing, they need to send the message to process 0 to print it, since only process 0 is guaranteed to be on the computer that the user is at.

2. Process 0 then needs to receive the message and print them.

3. Google MPI_Send and MPI_Recv for the details.

4. Remember that process 0 needs to receive a whole bunch of messages, so you will need a loop in process 0 to do the receiving.

5. The first way that most students do this, you will notice that it *always* then prints out the results *in order*. Ask me once you have reached this point for what to do to print out in whatever order it happens to come in.

6. As always, please ask me for help if you have problems. *Do not sit there being confused*. I am here to help you.

Part 2: Broadcast

A common need is for one process to get data from the user, either by reading from the terminal or command line arguments, and then to distribute this information to all other processors.

Write a program that reads an integer value from the terminal and distributes the value to all of the MPI processes. Each process should print out its rank and the value it received. Values should be read until a negative integer is given as input.

What's the command you used to send the data to everyone?

Note: You do not need to hand in this laboratory sheet. Answer them and use them as notes for you to study later. The things you learn in this lab will also help you answer the quiz at the next lecture.