
2nd Course in MPI: **Problem Set**

Science & Technology Support
High Performance Computing

Ohio Supercomputer Center
1224 Kinnear Road
Columbus, OH 43212-1163



1) Say that 4 MPI processes each has the following integer in its local memory

P:0 25 P:1 12 P:2 159 P:3 33

Using `MPI_Sendrecv()` only, pass these numbers between the processors until their order is reversed. So at the end of your program each processors should have the following integers:

P:0 33 P:1 159 P:2 12 P:3 25

2) Redo the program in lectures that uses `MPI_Get()`. Have the same data transferred between the same two MPI processes but use `MPI_Put()` instead.

3) Write an MPI class roster program. Process 0 should read in from standard input the number of students in the class. It should then dynamically create an integer array of the correct size. The array should be filled with ID numbers (starting at 1000). Then, Process 0 should post an `MPI_Send()` sending the array to Process 1.

Process 1 does not know how big the incoming message is, i.e., the size of the array. Therefore, Process 1 should first probe the message to find out its length. Process 1 should dynamically allocated an array of just the correct size and Then receive the message.

4) In this MPI program, Process 0 will have a normal array, that is, a statically declared array of reals of some KIND. The programmer is free to pick the dimensions and calculate the elements in whatever way they like. Process 1 has an allocatable array. Write MPI Fortran 90 code that transfers the array from Process 0 into the array of Process 1. You must work under the assumption that Process 1 knows absolutely nothing about the Process 0 array (except its dimensionality). Be sure to use the correct MPI_Datatype to transfer the KIND of reals chosen.

Have your code output whatever is necessary to confirm that it worked as expected

5) Write C++ code that will transfer a derived object from one process to another. The base class should be called Rect. It should have two data members, length and width and member functions will give values to length and width, calculate the area of the rectangle, calculate the perimeter of the rectangle, and print out the area and perimeter.

A class called Zoid will be the derived class. It should have one data member which is the height of a rectangular column. It should have member functions which initialize the height, calculate the column volume, and print out its volume.

Problem Set

Process0 should send a Zoid object to Process1. Before sending it should initialize the length and width of the Rect class. After reception, Process1 should initialize the height, and then should output all three calculated values: perimeter, area, and volume.

- 6) Write an MPI program in which each process writes out its rank in parallel to the same file. Use four MPI processes. In this case, you should create a File View for the output file. Since the output file will be composed simply of four integers, the record type and file type needed as arguments for the `MPI_FILE_SET_VIEW` command should both be just `MPI_INTEGER`.

7) For this problem you will write a *complete* MPI program in which a two-dimensional array is first initialized in parallel and then parts of the array are written by each MPI process to the same output file. The two dimensional array of integers should have 50 rows and 64 columns. Each element of the array should contain the result of the following simple arithmetic with its indices:

$$u(i,j) = i + (2 * j)$$

Use 4 MPI processes and have each responsible for initializing and writing out 16 columns of the array. Thus, the rank 0 process should work on the $j=1$ to $j=16$ columns, the rank 1 process should work on the $j=17$ to $j=32$ columns, and so on. Make the name of the joint output file `ex1.data`

USEFUL TIPS: First, you may need to use the MPI utility function `MPI_EXTENT` to determine how big an `MPI_INTEGER` is (in bytes) on the machine you are using. Second, `ex1.data` is a *binary file*. To convert it to a text file you can read, use the following command:

```
od -d -Ad ex1.data > ex1.txt
```