

#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>

// taken from https://stackoverflow.com/questions/18945129/mpi-broadcasting-2d-array

// as an interesting starting point for our problem.

int main(int argc, char\*\* argv)

{

    int procNum, procRank;

    int m,n;

    int sumProc = 0, sumAll = 0;

    int\*\* arr;

    MPI\_Status status;

/\*

    MPI\_Init ( &argc, &argv );

    MPI\_Comm\_size ( MPI\_COMM\_WORLD, &procNum );

    MPI\_Comm\_rank ( MPI\_COMM\_WORLD, &procRank );

   \*/

    MPI\_Init(NULL, NULL);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &procNum);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &procRank);

    if (procRank == 0)

    {

        printf("Type the array size \n");

        scanf("%i %i", &m, &n);

    }

    MPI\_Bcast(&m, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

    MPI\_Bcast(&n, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

    // This bit creates the array

    /\*

    // this works but creates the array on the stack.

        int sample\_array[4][5] = {

   {50, 55, 62, 70, 85},

   {35, 42, 45, 47, 49},

   {32, 33, 36, 37, 38},

   {25, 30, 30, 35, 30},

   in CPP there are a couple of 'right' ish ways to do this:

   the syntactically correct but slow system is:

   int\*\* arr = new int\*[&m];

   for(int i = 0; i < &m; ++i)

    arr[i] = new int[&n];

  is workable syntax

  But then you need to delete everything at the end in a good program

  for(int i = 0; i < rowCount; ++i) {

    delete [] a[i];

}

delete [] a;

    Beyond the scope of today, a much faster way is to make one big memory blob -

    int \*ary = new int[sizeX\*sizeY];

    // ary[i][j] is then rewritten as

    ary[i\*sizeY+j]

https://stackoverflow.com/questions/936687/how-do-i-declare-a-2d-array-in-c-using-new

    }

    \*/

    arr = new int\*[m];

    for (int i = 0; i < m; i++)

        arr[i] = new int[n];

    if (procRank == 0)

    {

        //HPC Lab 4 This generates a random array, you want to replace this with your own static array.

        //

        for (int i = 0; i < m; i++)

        {

            for (int j = 0; j < n; j++)

            {

                    arr[i][j] = rand() % 30;

                    printf("%i ", arr[i][j]);

            }

            printf("\n");

        }

    }

    //MPI\_Bcast(&arr[0][0], m\*n, MPI\_INT, 0, MPI\_COMM\_WORLD);

    // this bit broadcasts the array to each node

    for (int i = 0; i < m; i++)

        MPI\_Bcast(arr[i], n, MPI\_INT, 0, MPI\_COMM\_WORLD);

    // you need two lines of code here to sum the array (rows and columns)

    for (int i = 0; i < m; i++)

        {

            for (int j = 0; j < n; j++)

            {

                    sumProc +=  arr[i][j];

            }

        }

     printf("Sum Proc for processor %d is %i\n", procRank, sumProc);

    MPI\_Reduce(&sumProc,&sumAll,1,MPI\_INT,MPI\_SUM,0,MPI\_COMM\_WORLD);

    if (procRank == 0)

    {

        printf("sumAll = %i \n", sumAll);

    }

    delete \*arr;

    MPI\_Finalize();

    return 0;

}