**5CS022 Distribute and Cloud Systems Programming Week 1 Workshop  
  
Overview**

The aim of this workshop is to familiarise you with building, compiling and running MPI programs. You can carry out this workshop on your own Linux system

**Tasks**

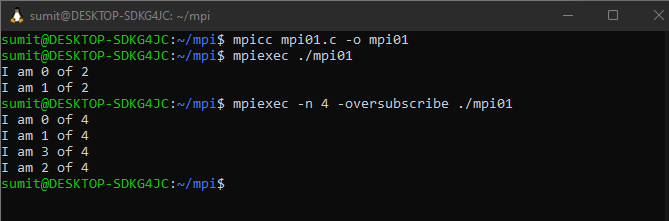
1. Download the sample MPI programs from the drive into your Linux system. Compile and run the program mpi01.c. To compile it, run the following command in the terminal:

**mpicc mpi01.c -o mpi01**

Now run it with the following:  
  
**mpiexec ./mpi01**   
  
This will (probably) only run only one process, which is not very interesting. Run it again with the following command::  
  
**mpiexec -n 4 -oversubscribe ./mpi01**

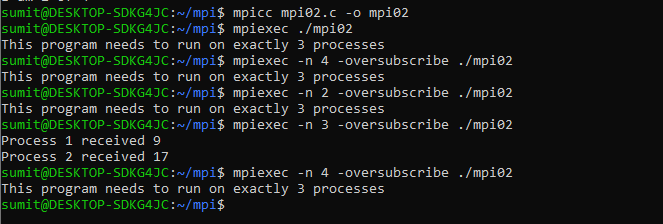
Note the output this time. It should indicate that 4 processes have run and they all have different process IDs.

Experiment with higher and higher numbers of processes until it stops running. Then have a look at the error message and try to work out why it stopped working.

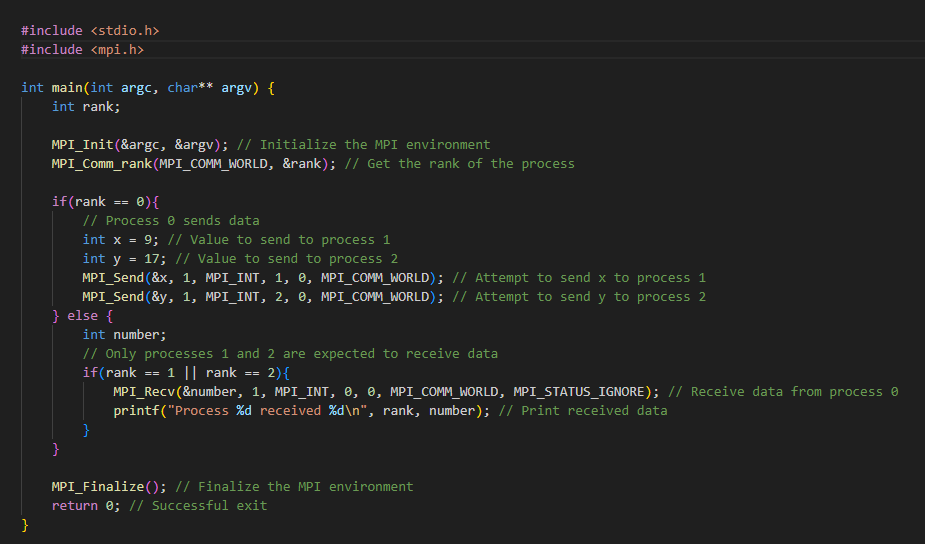


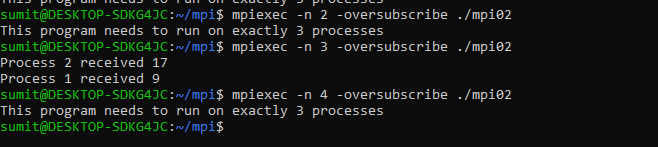
1. Compile and run the program mpi02.c. Try running it with 2, 3 and 4 processes. Eg.:  
    **mpiexec -n 2 -oversubscribe ./mpi02  
   mpiexec -n 3 -oversubscribe ./mpi02  
   mpiexec -n 4 -oversubscribe ./mpi02**

Note what happens. It doesn't let you run the program with anything other than 3 processes.



1. Now change the code so that you remove the check for only 3 processes. Now run it with 2, then 3 , then 4 and then more processes.



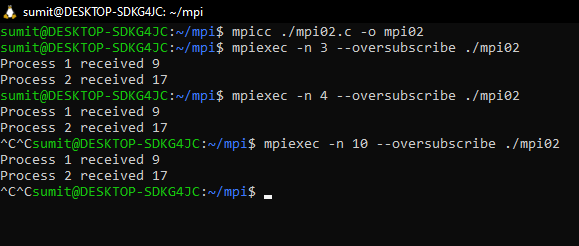


1. When you try to run it with 4 or more processes, it probably runs and appears to work, but never ends. You will have to end with "Ctrl-C". Why do you think it doesn't end when you run it with more than 3 processes? Change it so that it will work with any number of processes.

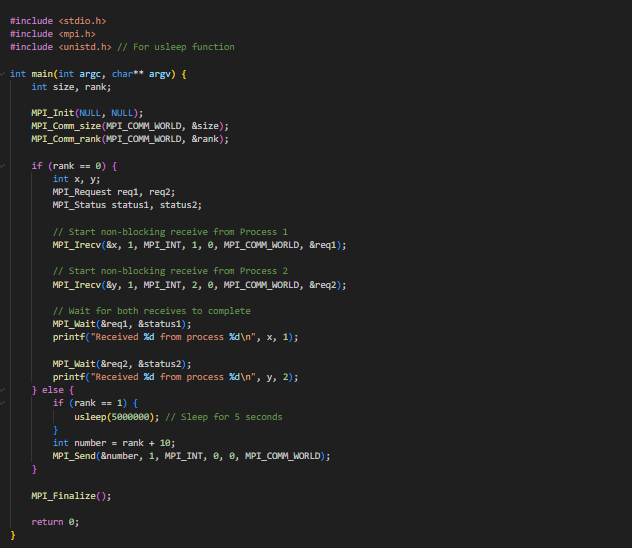
mpiexec -n 2 ./mpi02

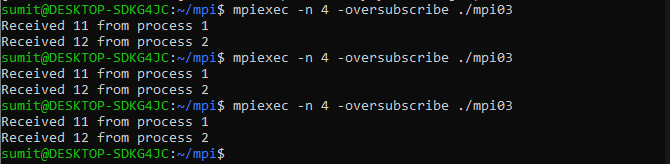
mpiexec -n 3 ./mpi02

mpiexec -n 4 ./mpi02



1. Build and run the program mpi03.c. In this program Process 0 will wait for messages from Process 1 and Process 2. However, Process 1 ends up blocking Process 2 because it sleeps for 5 seconds.





1. The following is a simple program that looks for prime numbers between 1 to 10000:

**#include <stdio.h>**

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

**{**

**int i, c;**

**int nstart=1, nfinish=10000;**

**printf("%s : Prime numbers between %d and %d are :\n",**

**argv[0], nstart, nfinish);**

**for(i=nstart; i<=nfinish; i++)**

**{**

**for(c=2; c<=i-1; c++)**

**{**

**if ( i%c==0 )**

**break;**

**}**

**if ( c==i )**

**printf("%s : %d\n",argv[0], i);**

**}**

**return 0;**

**}**

Convert it to MPI so that it can run with different numbers of processes including just one process.

#include <stdio.h>

#include <math.h>

#include <mpi.h>

// Function to check if a number is prime

int is\_prime(int n) {

    if (n <= 1) return 0; // 0 and 1 are not prime

    if (n <= 3) return 1; // 2 and 3 are prime

    if (n % 2 == 0 || n % 3 == 0) return 0; // Exclude multiples of 2 and 3

    for (int i = 5; i \* i <= n; i += 6) {

        if (n % i == 0 || n % (i + 2) == 0) return 0; // Exclude multiples of other primes

    }

    return 1; // It's prime

}

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

    int rank, size;

    int nstart = 2, nfinish = 10000; // Start and end of the range

    int local\_start, local\_end;

    int range, local\_range;

    int local\_primes = 0, total\_primes = 0;

    MPI\_Init(&argc, &argv);

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

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

    range = nfinish - nstart + 1;

    local\_range = range / size;

    local\_start = rank \* local\_range + nstart;

    local\_end = (rank + 1) \* local\_range + nstart - 1;

    if (rank == size - 1) {

        local\_end = nfinish;

    }

    // Each process counts primes in its range

    for (int i = local\_start; i <= local\_end; i++) {

        if (is\_prime(i)) {

            local\_primes++;

        }

    }

    // Reduce local counts to the root process

    MPI\_Reduce(&local\_primes, &total\_primes, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

    // Print prime numbers by the root process

    if (rank == 0) {

        printf("Total prime numbers between %d and %d are: %d\n", nstart, nfinish, total\_primes);

    }

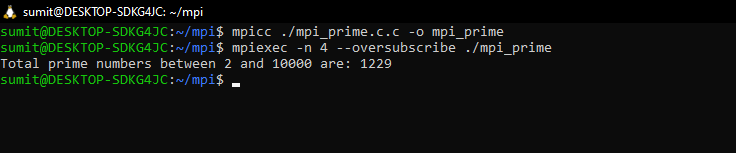
    MPI\_Finalize();

    return 0;

}

**mpicc -o mpi\_prime mpi\_prime.c -lm**

**mpiexec -n <num\_processes> --oversubscribe ./mpi\_prime**

****