UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

DISTRIBUTE AND CLOUD SYSTEMS PROGRAMMING (5CS022)

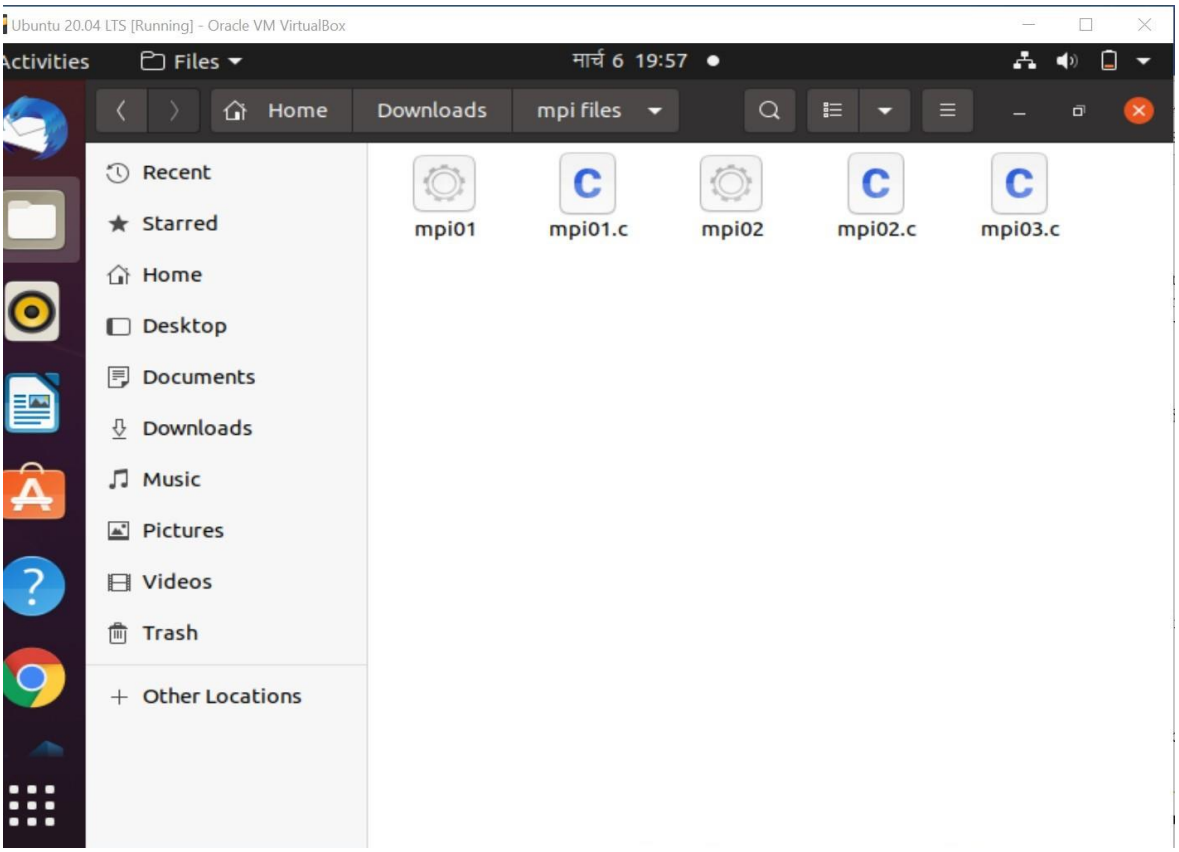
WEEK 1 WORKSHOP

Student Id : 2065697
Student Name : Dhiraj Kumar Sah Kanu
Group : L5CG12
Submitted on : March 7, 2022

5CS022 Distribute and Cloud Systems Programming Week 1 Workshop

Tasks

1. Download the sample MPI programs from Canvas 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
```

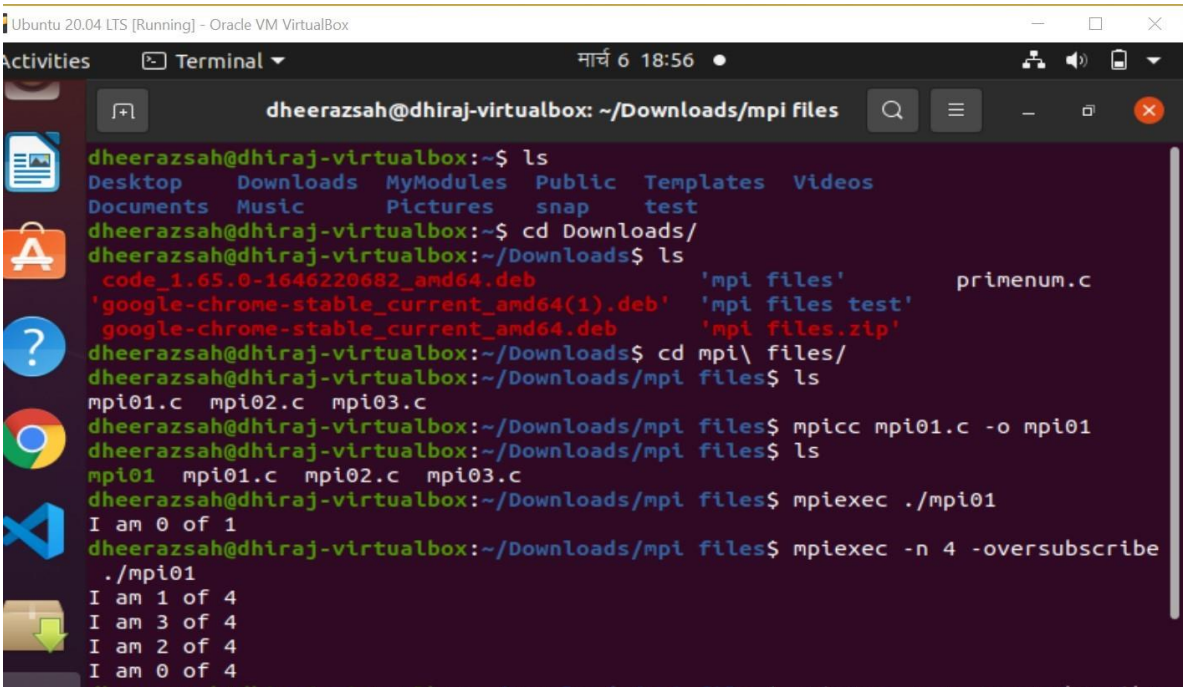
Now run it with the following:

```
mpiexec ./mpi01
```

This will (probably) only run only one process, which is not very interesting. Run it again with the following command::

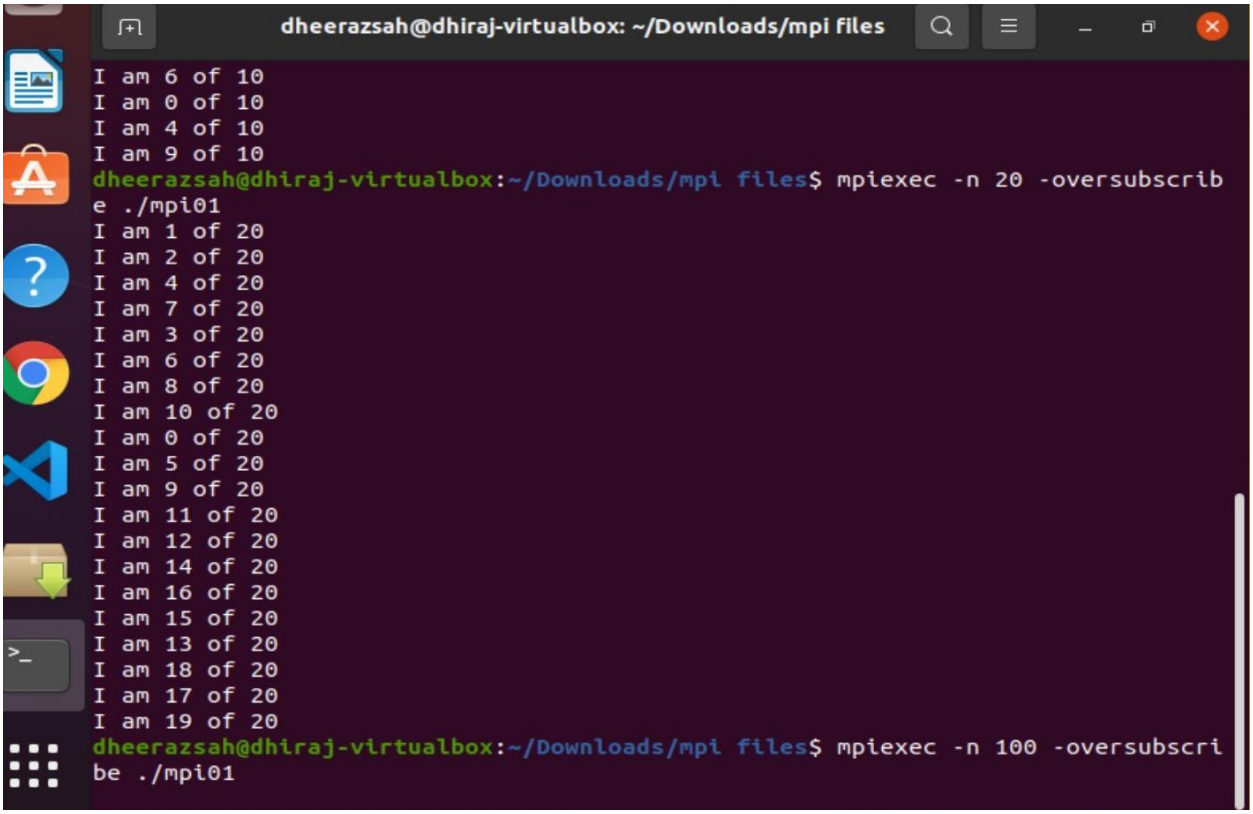
```
mpiexec -n 4 -oversubscribe ./mpi01
```

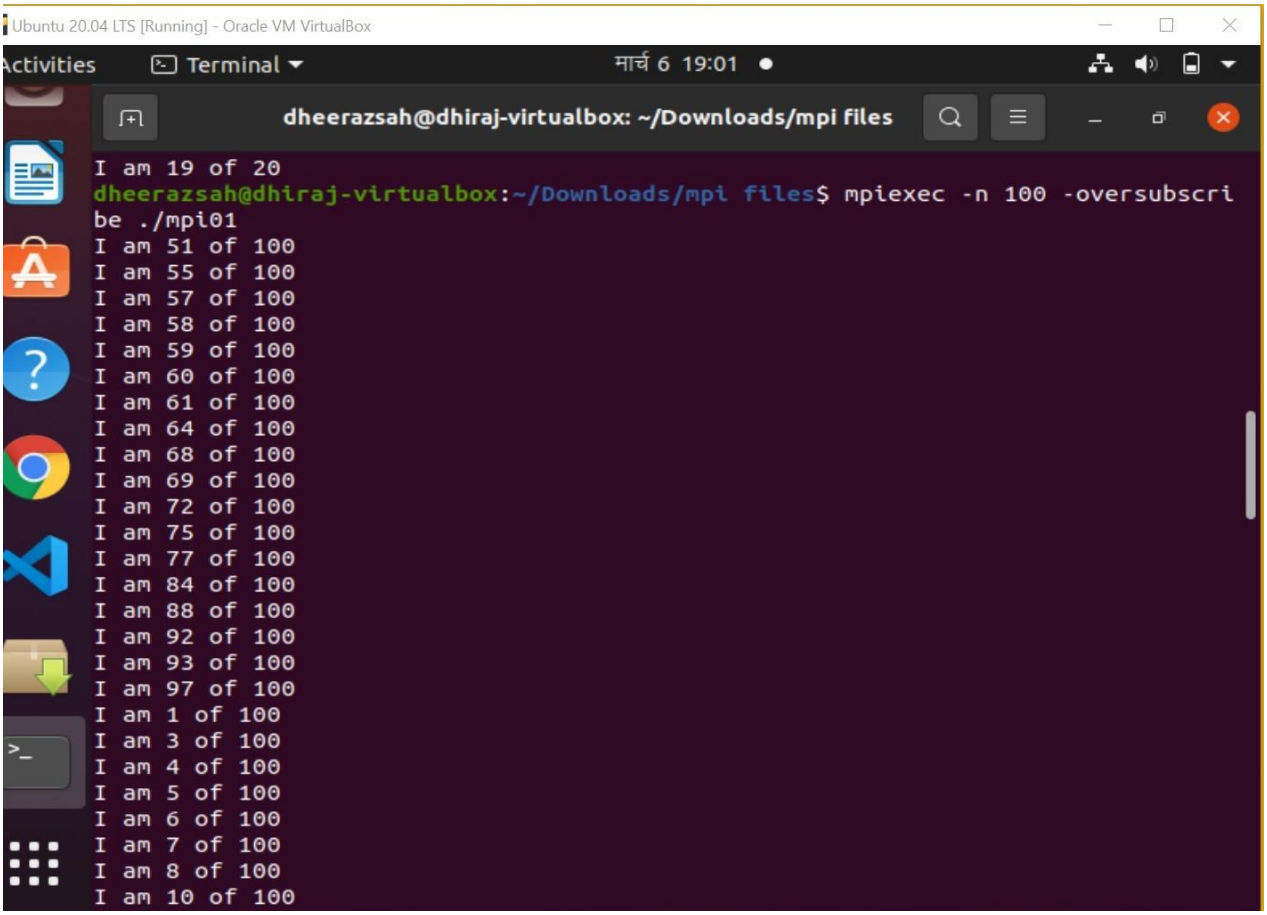
Note the output this time. It should indicate that 4 processes have run and they all have different process IDs.

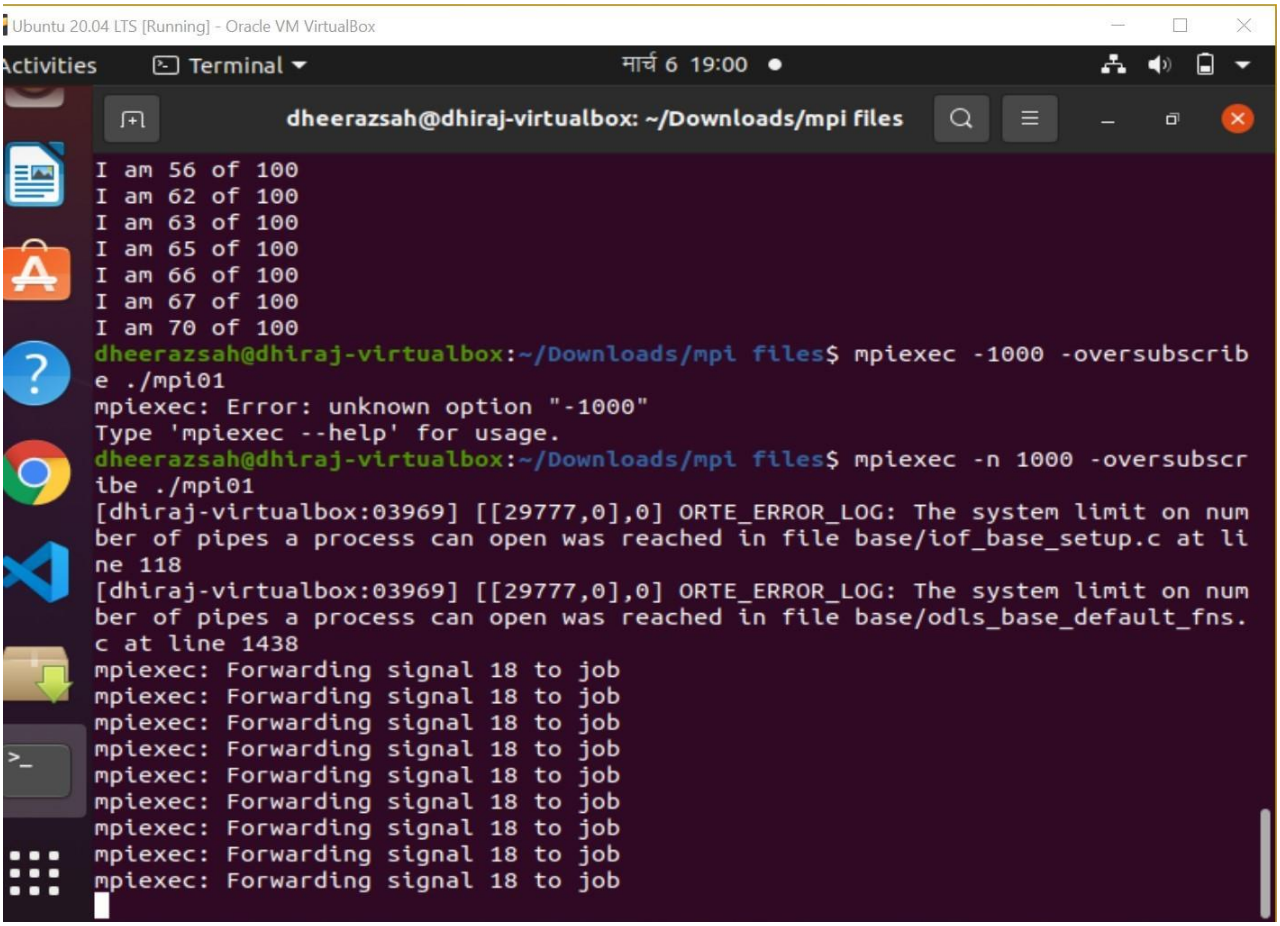


In the above screenshot, details of the available directories are checked and then is redirected to Downloads and then mpi files to compile them. The mpi01.c file consisting source code is then compiled using mpicc mpi01.c -o mpi01. Then mpiexec ./mpi01 is executed which runs only one process. To run four processes mpiexec -n 4 -oversubscribe ./mpi01 is compiled with different IDs.

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


dheerazsah@dhiraj-virtualbox: ~/Downloads/mpi files
I am 6 of 10
I am 0 of 10
I am 4 of 10
I am 9 of 10
dheerazsah@dhiraj-virtualbox:~/Downloads/mpi files$ mpiexec -n 20 -oversubscrib
e ./mpi01
I am 1 of 20
I am 2 of 20
I am 4 of 20
I am 7 of 20
I am 3 of 20
I am 6 of 20
I am 8 of 20
I am 10 of 20
I am 0 of 20
I am 5 of 20
I am 9 of 20
I am 11 of 20
I am 12 of 20
I am 14 of 20
I am 16 of 20
I am 15 of 20
I am 13 of 20
I am 18 of 20
I am 17 of 20
I am 19 of 20
dheerazsah@dhiraj-virtualbox:~/Downloads/mpi files$ mpiexec -n 100 -oversubscri
be ./mpi01


Ubuntu 20.04 LTS [Running] - Oracle VM VirtualBox
Activities
Terminal
मार्च 6 19:01
dheerazsah@dhiraj-virtualbox: ~/Downloads/mpi files
I am 19 of 20
dheerazsah@dhiraj-virtualbox:~/Downloads/mpi files$ mpiexec -n 100 -oversubscri
be ./mpi01
I am 51 of 100
I am 55 of 100
I am 57 of 100
I am 58 of 100
I am 59 of 100
I am 60 of 100
I am 61 of 100
I am 64 of 100
I am 68 of 100
I am 69 of 100
I am 72 of 100
I am 75 of 100
I am 77 of 100
I am 84 of 100
I am 88 of 100
I am 92 of 100
I am 93 of 100
I am 97 of 100
I am 1 of 100
I am 3 of 100
I am 4 of 100
I am 5 of 100
I am 6 of 100
I am 7 of 100
I am 8 of 100
I am 10 of 100

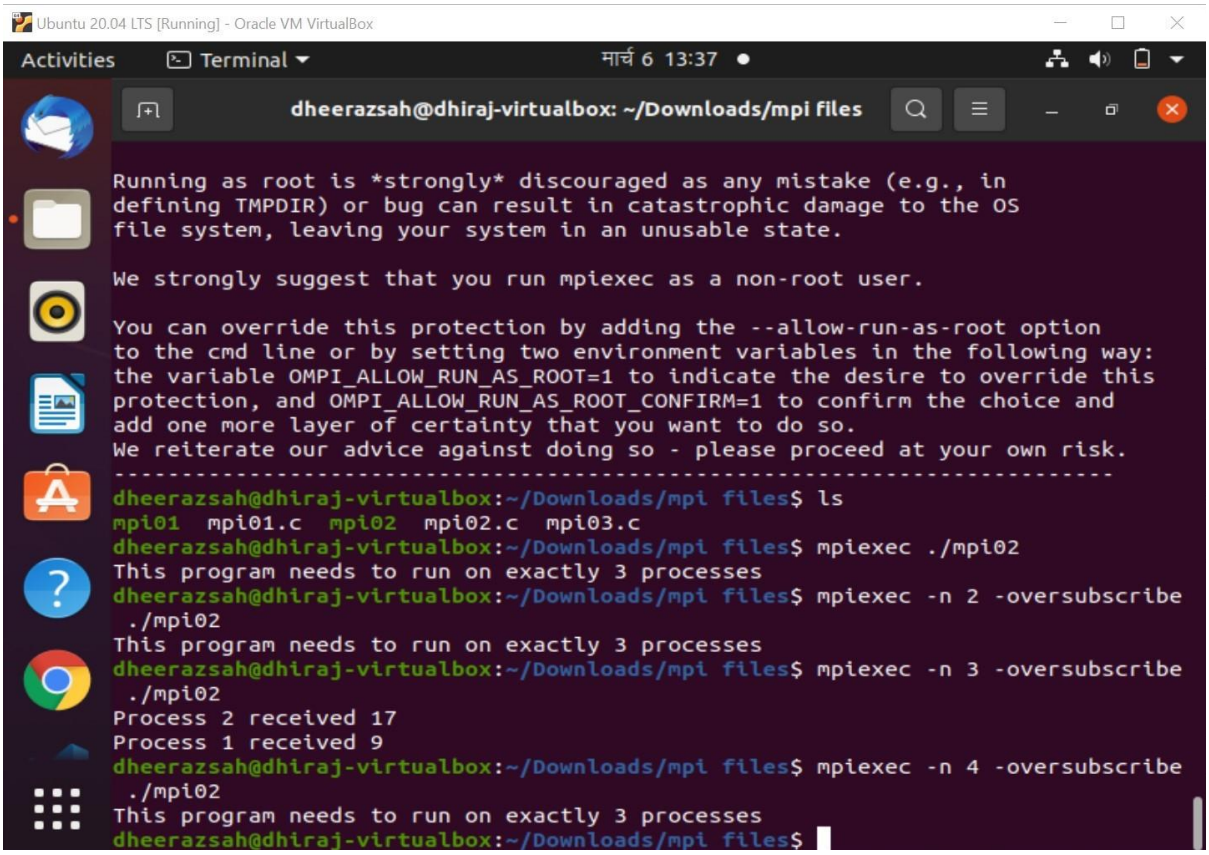


While experimenting with higher and higher number of processors, I found that after reaching a certain number it shows error when executed. To demonstrate, I used 6 processes, 10 processes, 20 processes, 100 processes then 1000 processes. Here, the program was not compiled when it reached 1000 processes. Here the number of processes were too much higher than the number of data.

2. Compile and run the program mpi02.c. Try running it with 2, 3 and 4 processes. Eg.:

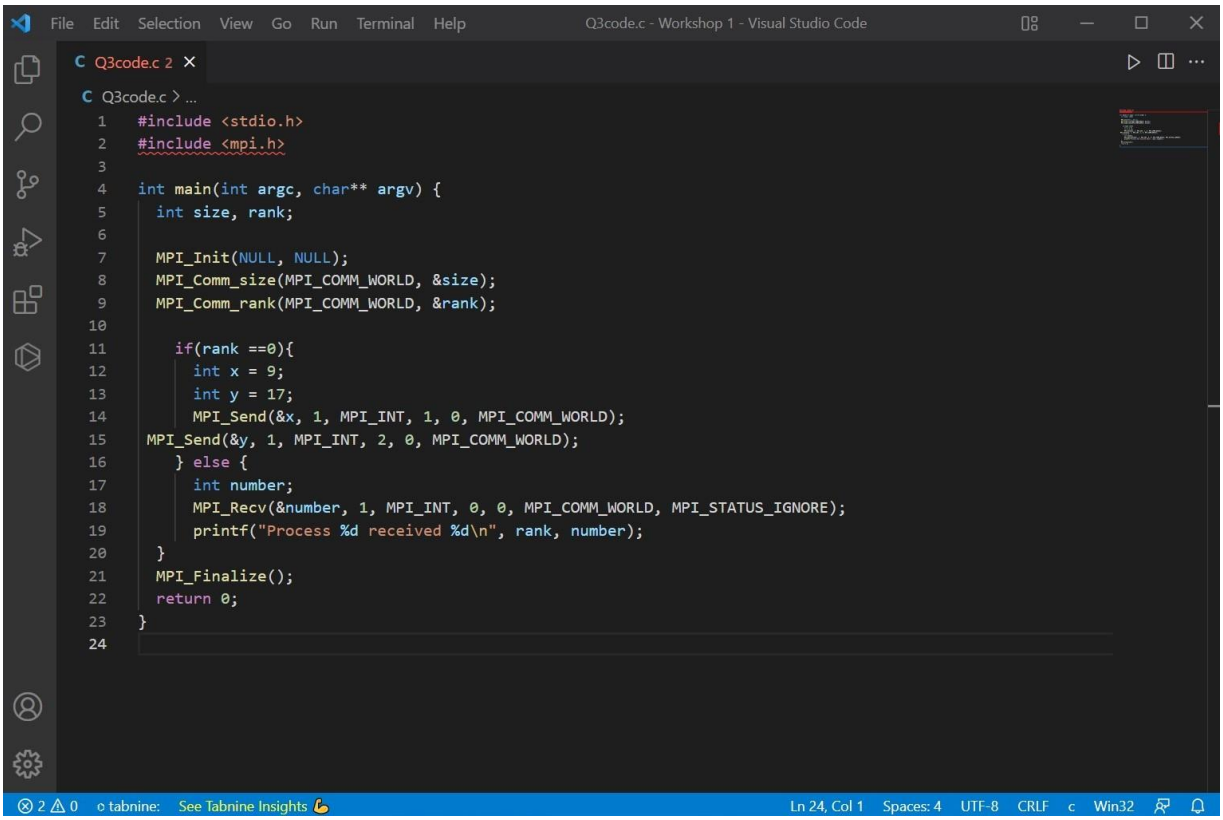
```
mpiexec -n 2 -oversubscribe ./mpi02
mpiexec -n 3 -oversubscribe ./mpi02
mpiexec -n 4 -oversubscribe ./mpi02
```

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

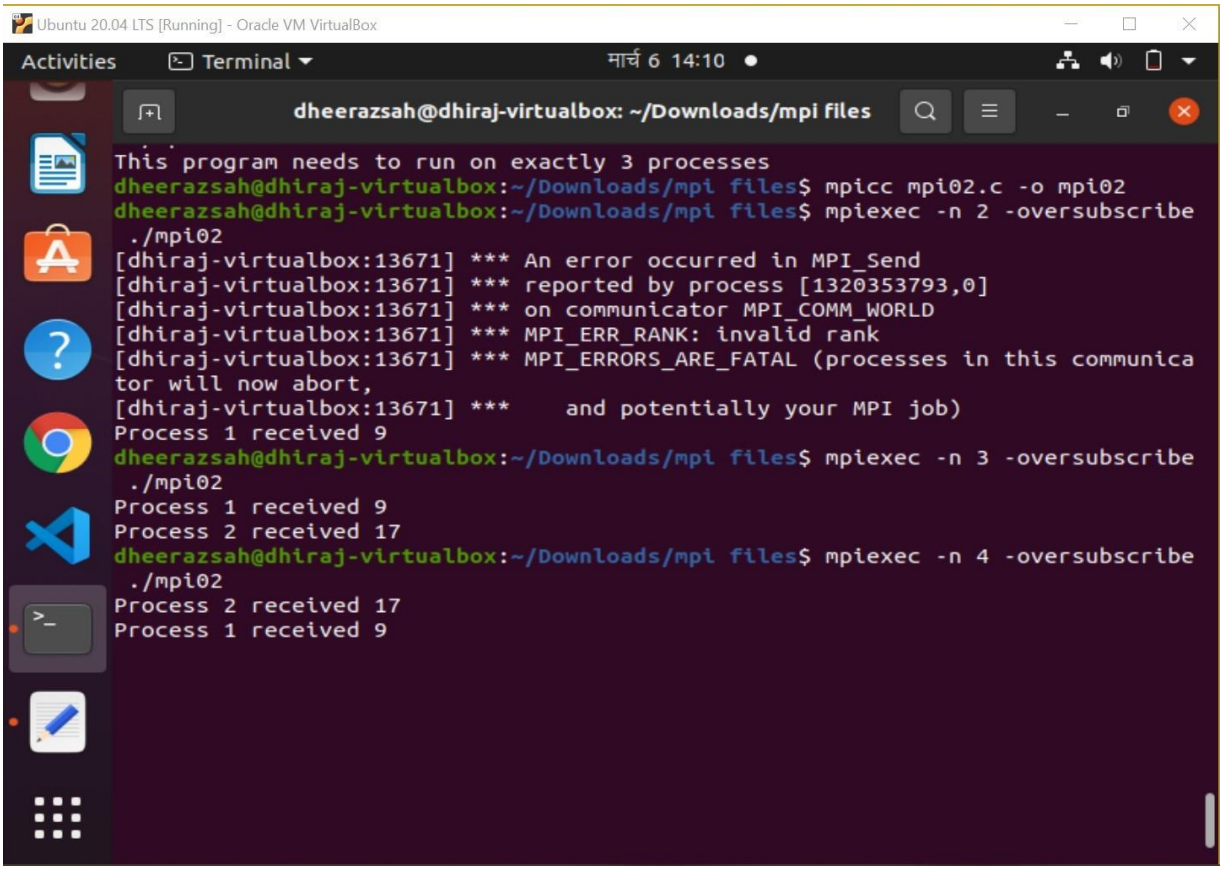


In the above screenshot, the program is executed using 2, 3, and 4 processes. At first, mpiexec ./mpi02 is compiled which runs on the processes used by the virtual box rather than using the processes of the actual device. To actually use different processes –oversubscribe is used, similarly to use a particular number of processes –n is used. The above program needs to run on exactly 3 processes. This is because the source code of this program will not accept size except 3.

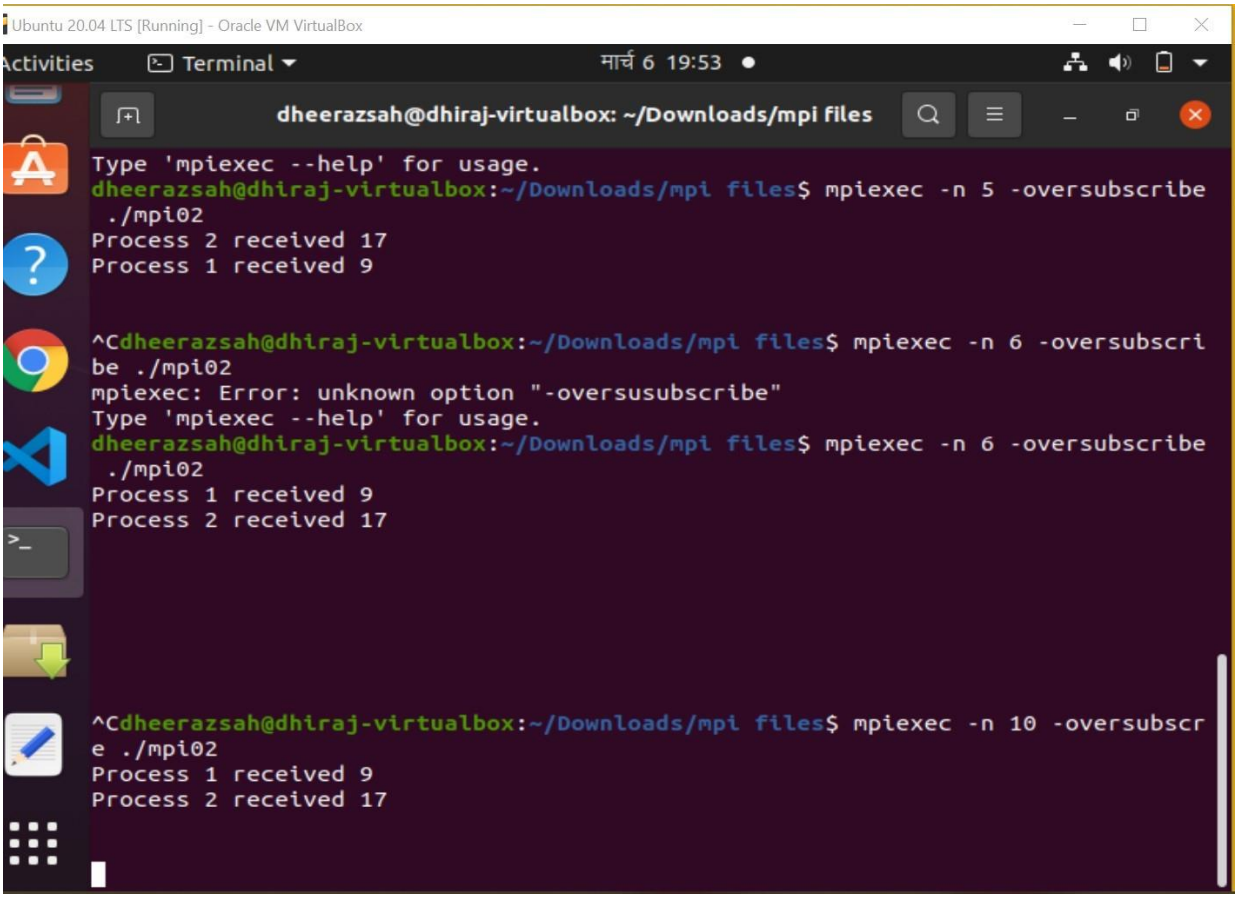
3. Now change the code so that you remove the check for only 3 processes. Now run it with 2, then 3 , then 4 and then more processes.



The above screenshot is the altered version of code of mpi02.c. Here, I have removed the if-else condition which was applied first and was not letting run any processes except 3. But now the condition if(size != 3) { if (rank == 0) { print(" This program needs to run on exactly 3 processes\n") ;}} else part was removed. The above code will let this program run in any processes.



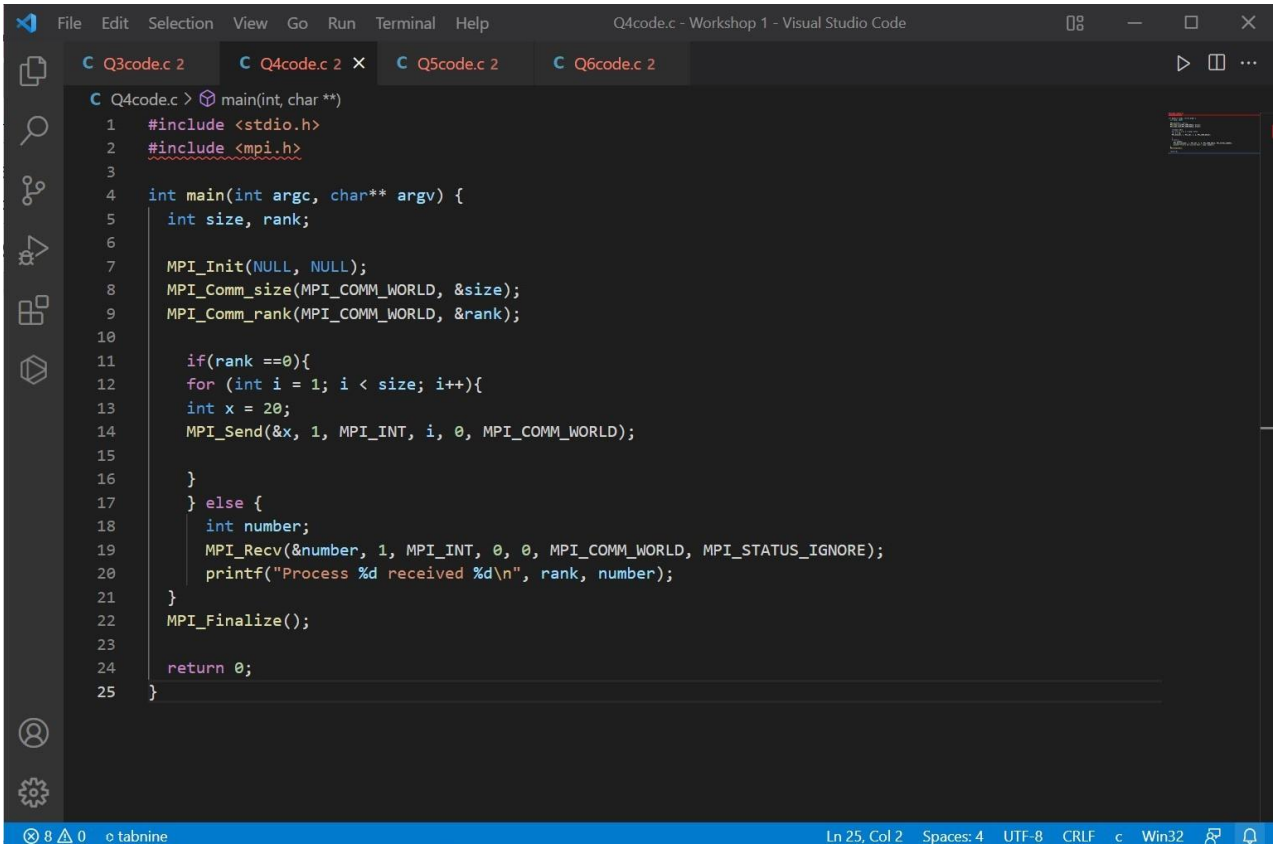
In the above screenshot, we can see there are still few errors when it was executed in 2 processes. The program was executed and it showed process 1 received 9 after few errors. When the code was executed using 3 processes, it was compiled without any error. Similarly, the code was executed using 4 processes too but here the program is trying to run more processes but it is not able to receive any data this creates a never ending process. We have to stop the executing process forcefully using Ctrl+C.



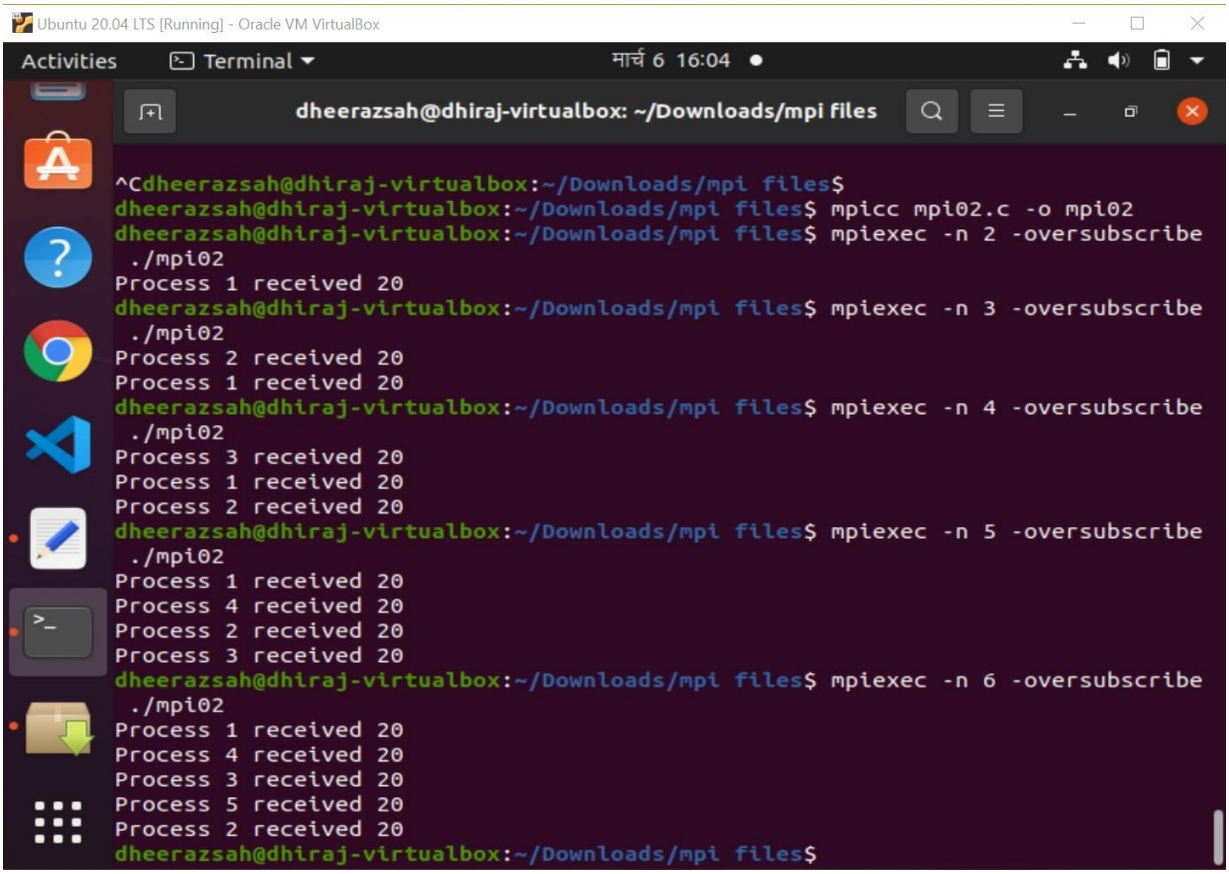
Here, I tried to use more processes, every time the program was executed but was never ending due to lack of data and every time it required to stop the program forcefully.

4. When you try to run it with 4 or more processes, it probably runs and appear 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.

The if-else condition inside the source code was not letting run any processes except 3. When I tried to run the program using 4 and more processes, it didn't let the program end, this is because there were more number of processes than the data due to which the processes started looking for data and as there was no data to left to look for, the program ended up never-ending.

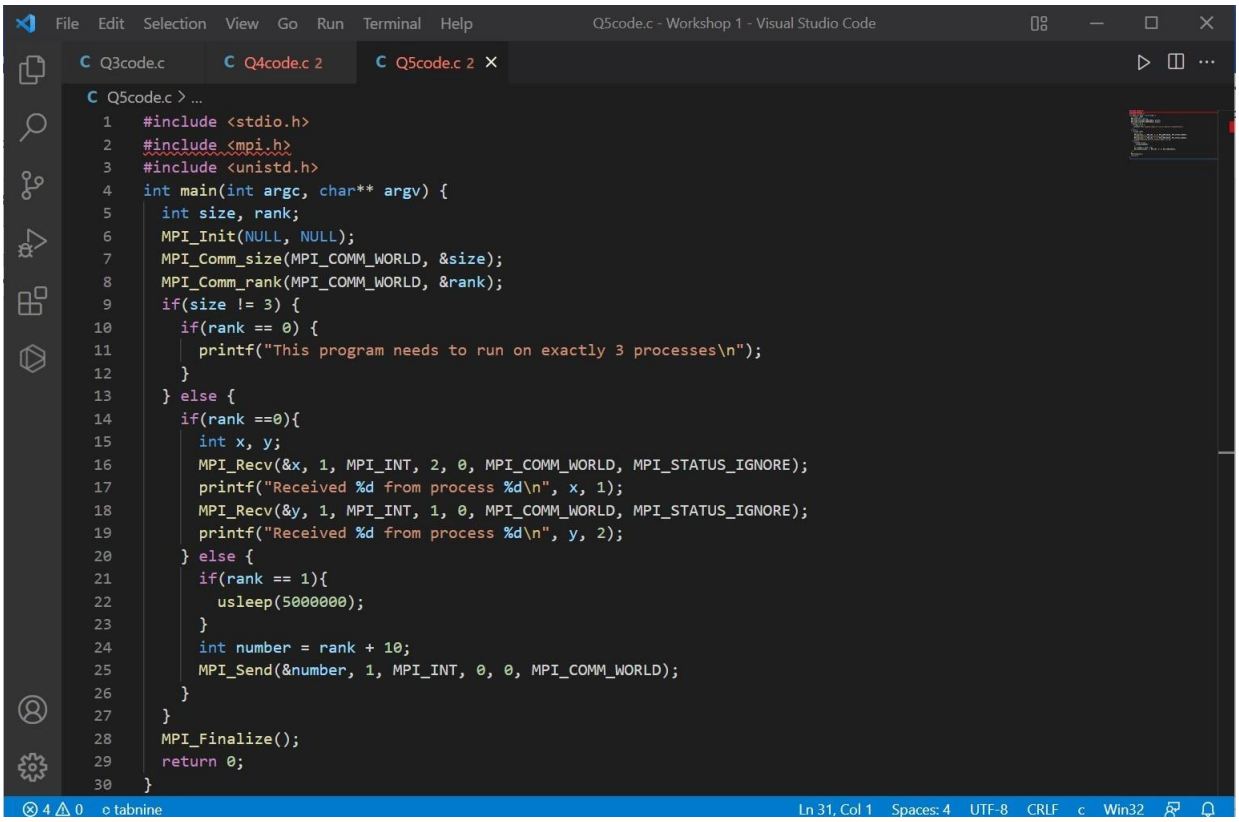


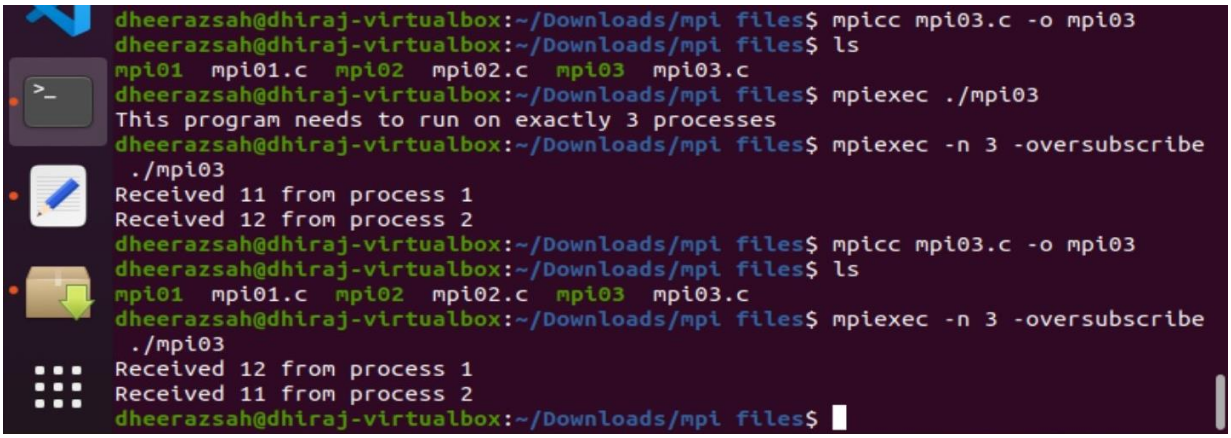
In the above code, instead fixing the ranks, initializing i=1 and creating a loop was done which selects processes form rank 1 to any large number of processes.



Here, we can see that the code is now fixed and is working on different processes properly rather than working only in 3 processes. To demonstrate, I tried to run the program in 2, 3, 4, 5 and 6 processes. Neither the program showed any error nor it ended up never-ending.

5. 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. How would you change the code so that Process 1 does not block Process 2, even if it does sleep for 5 seconds?



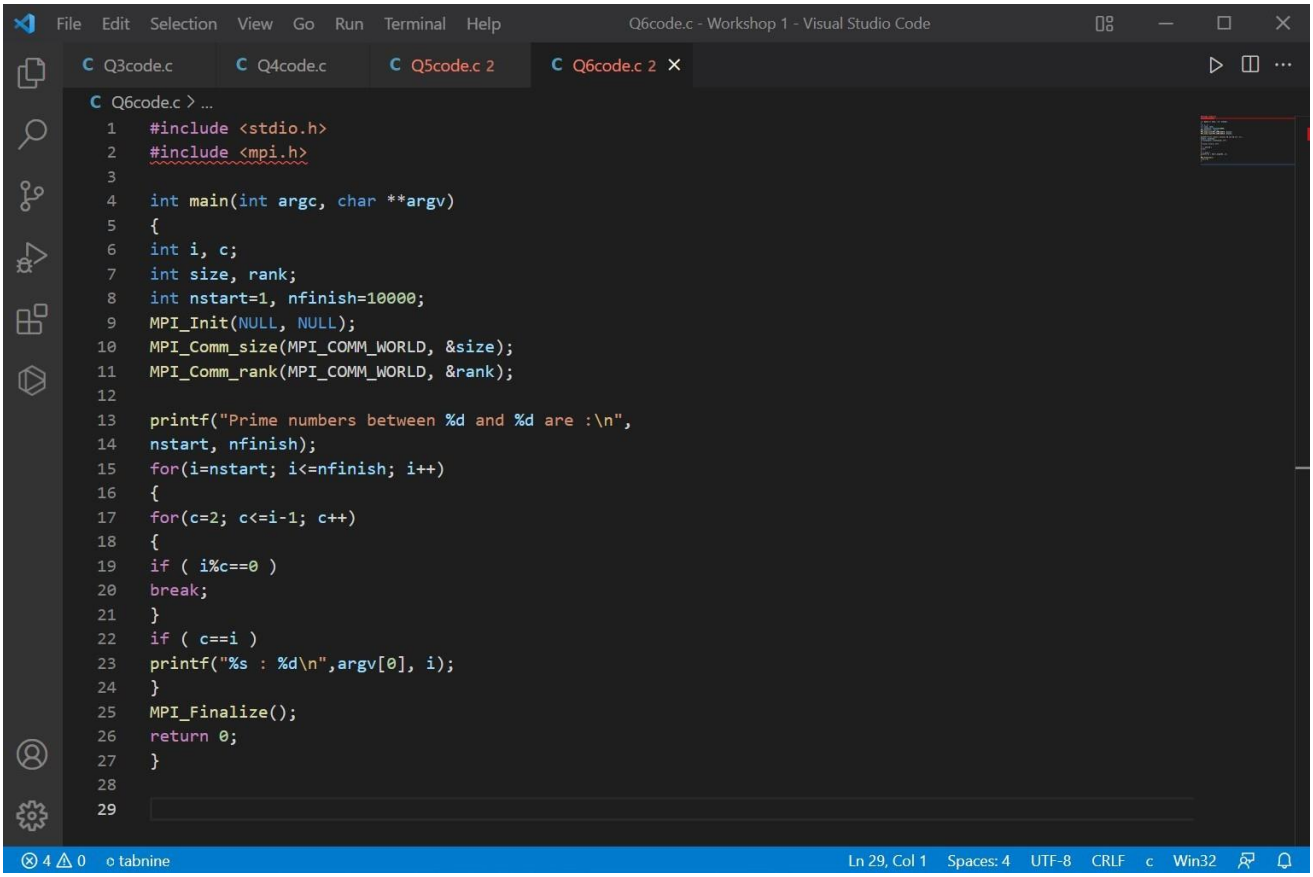


In the above screenshot, I tried to execute the mpi program without selecting a particular process, which then showed it requires exactly 3 processes to run. After executing the program with 3 processes, the program was executed but it took 5 seconds to show the result. At first while receiving we used the value of rank 1 where MPI_COMM_WORLD is helping to establish a communication area. Instead using the value of rank 1, value of rank 2 was used this is because rank 2 processes is not waiting for 5 seconds. Process 1 is blocking the executing process for few seconds so a slight change in line 16 and line 18 is done where the processes 1 and 2 are interchanged with each other. This after executing gives prompt result as 'Received 12 from process 1' and waits 5 seconds to give 'Received 11 from process 2' result.

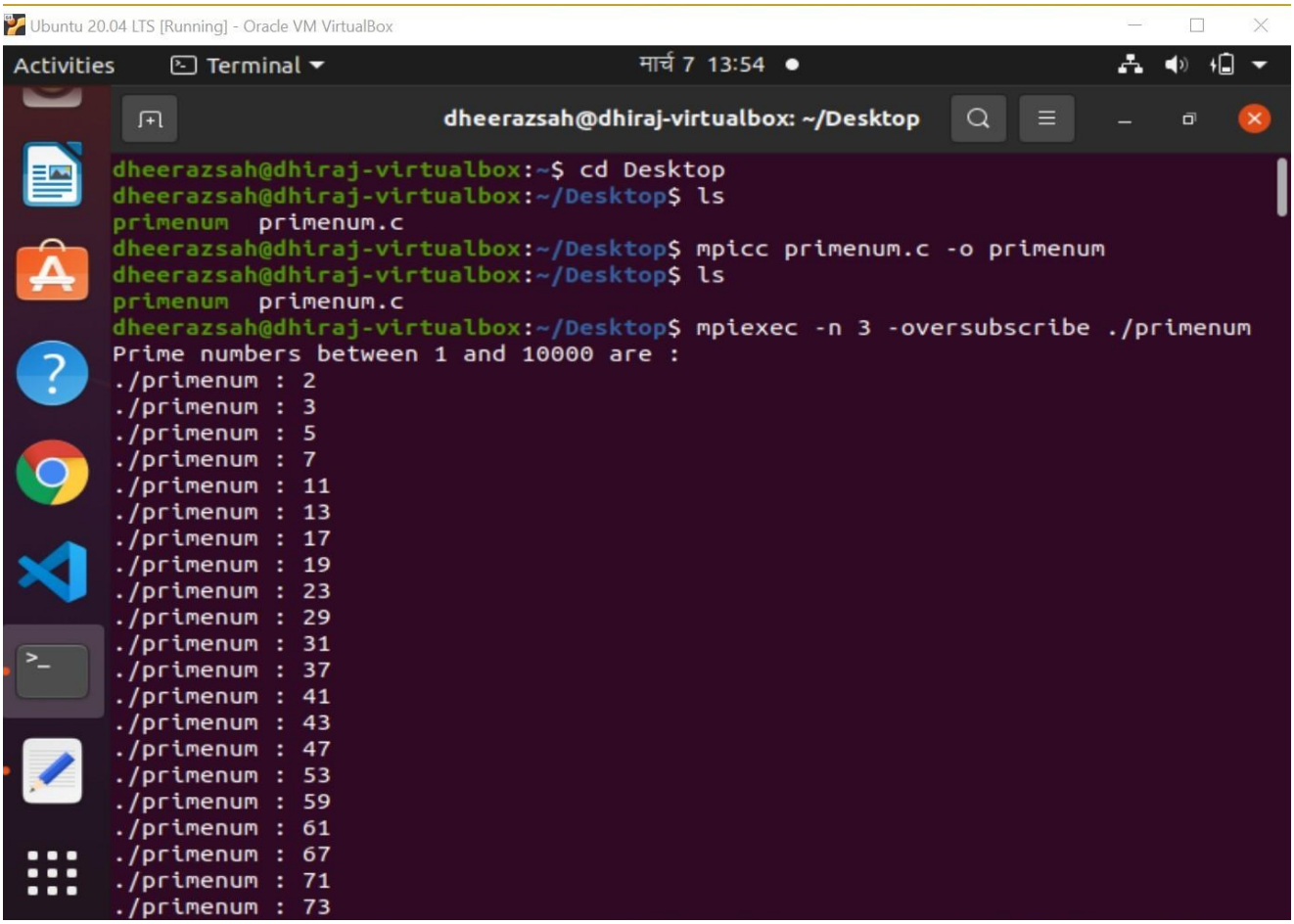
6. The following is a simple program that looks for prime numbers between 1 to 10000:

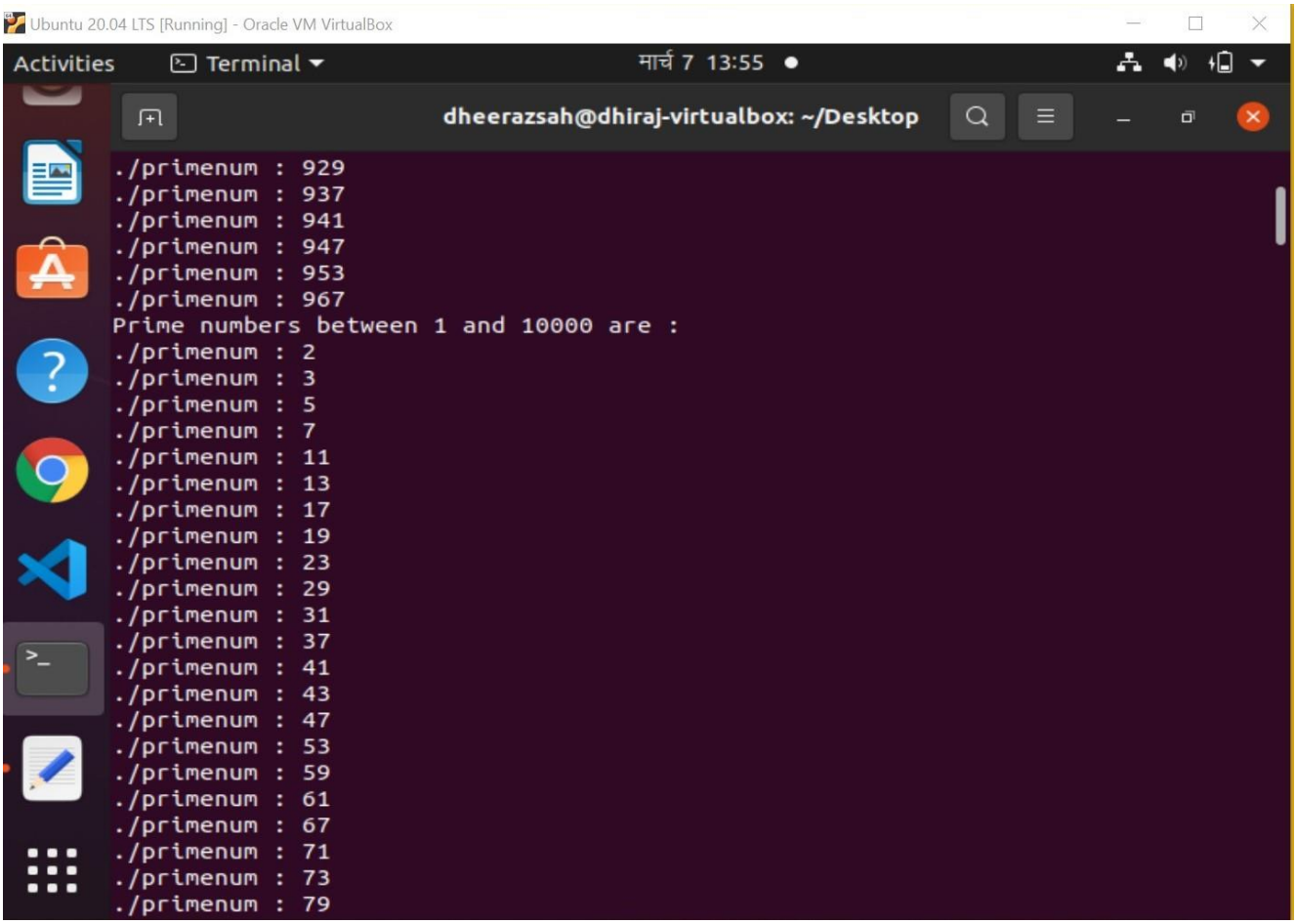
```
#include <stdio.h>

int main(int argc, char **argv)
{
  int i, c;
  int nstart=1, nfinish=10000;
  printf("%s : Prime numbers between %d and %d are :\n",
         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;
}
```

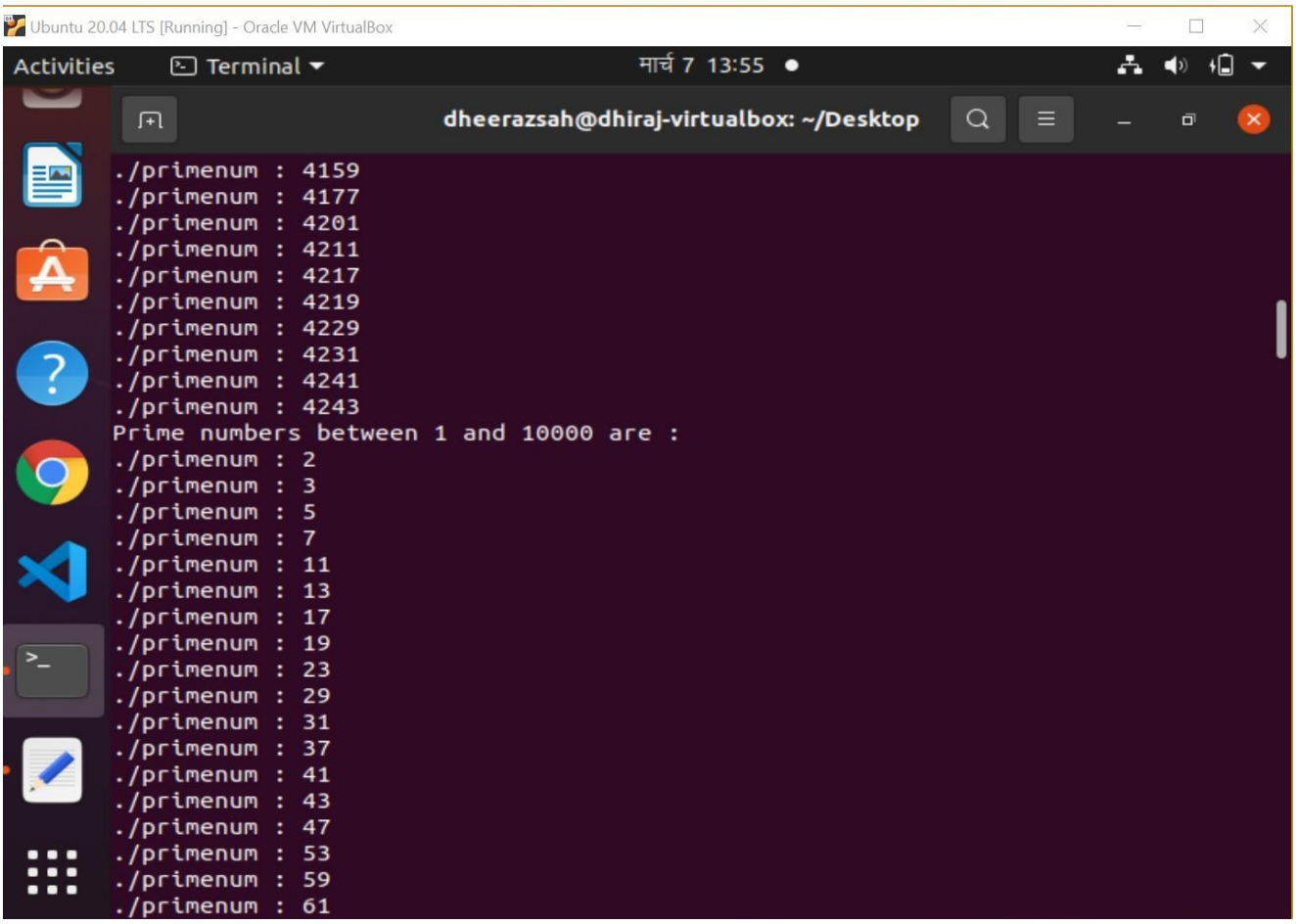
Convert it to MPI so that it can run with different numbers of processes including just one process.



In the above screenshot, I converted the simple program into MPI. First I implemented mpi library using #include <mpi.h>. Then, initilaizing and finalizing the MPI was implemented. Also MPI_COMM_WORLD was included to create a environment for communication. Starting value was entered as 1 and ending value as 1000 as mentioned in the question.




Ubuntu 20.04 LTS [Running] - Oracle VM VirtualBox
Activities
Terminal
मार्च 7 13:55
dheerazsah@dhiraj-virtualbox: ~/Desktop
./primenum : 929
./primenum : 937
./primenum : 941
./primenum : 947
./primenum : 953
./primenum : 967
Prime numbers between 1 and 10000 are :
./primenum : 2
./primenum : 3
./primenum : 5
./primenum : 7
./primenum : 11
./primenum : 13
./primenum : 17
./primenum : 19
./primenum : 23
./primenum : 29
./primenum : 31
./primenum : 37
./primenum : 41
./primenum : 43
./primenum : 47
./primenum : 53
./primenum : 59
./primenum : 61
./primenum : 67
./primenum : 71
./primenum : 73
./primenum : 79


Ubuntu 20.04 LTS [Running] - Oracle VM VirtualBox
Activities
Terminal
मार्च 7 13:55
dheerazsah@dhiraj-virtualbox: ~/Desktop
./primenum : 4159
./primenum : 4177
./primenum : 4201
./primenum : 4211
./primenum : 4217
./primenum : 4219
./primenum : 4229
./primenum : 4231
./primenum : 4241
./primenum : 4243
Prime numbers between 1 and 10000 are :
./primenum : 2
./primenum : 3
./primenum : 5
./primenum : 7
./primenum : 11
./primenum : 13
./primenum : 17
./primenum : 19
./primenum : 23
./primenum : 29
./primenum : 31
./primenum : 37
./primenum : 41
./primenum : 43
./primenum : 47
./primenum : 53
./primenum : 59
./primenum : 61