



HERALD
COLLEGE
K A T H M A N D U



Module Title: Distributed and Cloud Systems Programming
(5CS022)

Subject Title: Workshop 01

Student Name: Nayan Raj Khanal

Student Code: 2227486

Instructor: Mr. Prabin Sapkota

Submitted on: 04.03.2023

University of Wolverhampton

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
```

A terminal window titled 'nayan@Nayan: ~/Downloads' with search, menu, and window control icons. The terminal shows the command 'mpicc mpi01.c -o mpi01' being executed. Below the terminal, a file icon for 'mpi01' is shown, featuring a gear icon and the filename 'mpi01'.

```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpicc mpi01.c -o mpi01
nayan@Nayan:~/Downloads$
```

Now run it with the following:

```
mpiexec ./mpi01
```


A terminal window titled 'nayan@Nayan: ~/Downloads' with search, menu, and window control icons. The terminal shows the command 'mpiexec ./mpi01' being executed, followed by the output 'I am 0 of 1'.

```
nayan@Nayan:~/Downloads$ mpiexec ./mpi01
I am 0 of 1
nayan@Nayan:~/Downloads$
```

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

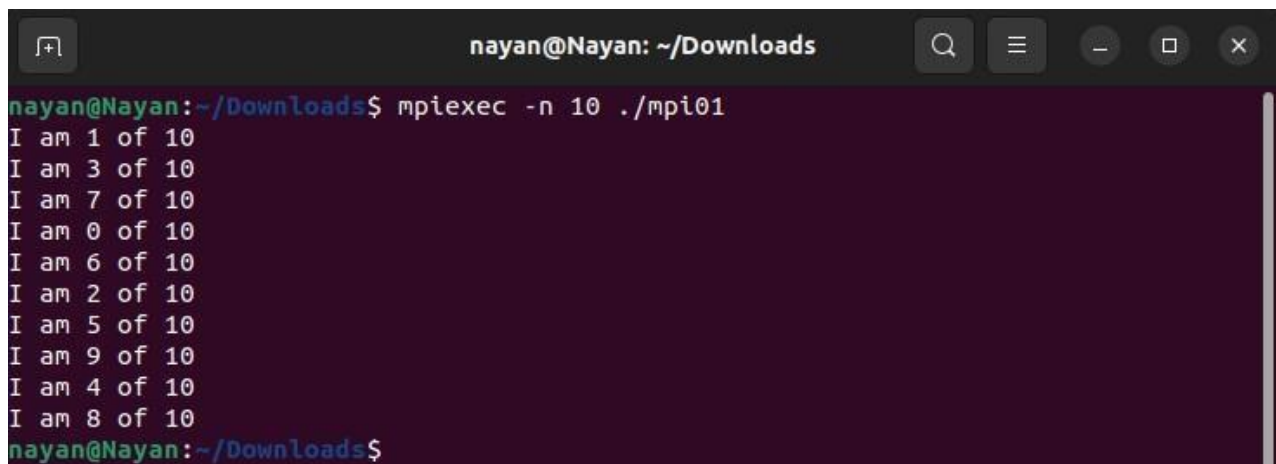
```
mpiexec -n 4 -oversubscribe ./mpi01
```

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

A terminal window titled 'nayan@Nayan: ~/Downloads' showing the execution of 'mpiexec -n 4 ./mpi01'. The output consists of four lines: 'I am 0 of 4', 'I am 1 of 4', 'I am 3 of 4', and 'I am 2 of 4', followed by the prompt 'nayan@Nayan:~/Downloads\$'.

```
nayan@Nayan:~/Downloads$ mpiexec -n 4 ./mpi01
I am 0 of 4
I am 1 of 4
I am 3 of 4
I am 2 of 4
nayan@Nayan:~/Downloads$
```

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.

A terminal window titled 'nayan@Nayan: ~/Downloads' showing the execution of 'mpiexec -n 10 ./mpi01'. The output consists of ten lines: 'I am 1 of 10', 'I am 3 of 10', 'I am 7 of 10', 'I am 0 of 10', 'I am 6 of 10', 'I am 2 of 10', 'I am 5 of 10', 'I am 9 of 10', 'I am 4 of 10', and 'I am 8 of 10', followed by the prompt 'nayan@Nayan:~/Downloads\$'.

```
nayan@Nayan:~/Downloads$ mpiexec -n 10 ./mpi01
I am 1 of 10
I am 3 of 10
I am 7 of 10
I am 0 of 10
I am 6 of 10
I am 2 of 10
I am 5 of 10
I am 9 of 10
I am 4 of 10
I am 8 of 10
nayan@Nayan:~/Downloads$
```

```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpiexec -n 1000 ./mpi01
[proxy:0:0@Nayan] HYDU_create_process (utils/launch/launch.c:21): pipe error (Too many open files)
[proxy:0:0@Nayan] launch_procs (pm/pmiserv/pmip_cb.c:730): create process returned error
[proxy:0:0@Nayan] HYD_pmcd_pmip_control_cmd_cb (pm/pmiserv/pmip_cb.c:906): launch_procs returned error
[proxy:0:0@Nayan] HYDT_dmxu_poll_wait_for_event (tools/demux/demux_poll.c:76): callback returned error status
[proxy:0:0@Nayan] main (pm/pmiserv/pmip.c:169): demux engine error waiting for event
[mpiexec@Nayan] control_cb (pm/pmiserv/pmiserv_cb.c:206): assert (!closed) failed
[mpiexec@Nayan] HYDT_dmxu_poll_wait_for_event (tools/demux/demux_poll.c:76): callback returned error status
[mpiexec@Nayan] HYD_pmci_wait_for_completion (pm/pmiserv/pmiserv_pmci.c:160): error waiting for event
[mpiexec@Nayan] main (ui/mpich/mpiexec.c:325): process manager error waiting for completion
```

There is a maximum number of processes a computer can access. If this maximum is exceeded, the OS will generate an error message signaling that many files are opened at once.

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.

```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpicc mpi02.c -o mpi02
nayan@Nayan:~/Downloads$
```



```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpiexec -n 2 ./mpi02
This program needs to run on exactly 3 processes
nayan@Nayan:~/Downloads$ mpiexec -n 3 ./mpi02
Process 1 received 9
Process 2 received 17
nayan@Nayan:~/Downloads$ mpiexec -n 4 ./mpi02
This program needs to run on exactly 3 processes
nayan@Nayan:~/Downloads$
```

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.

CODE:

```
Open  [icon] mpi02.c ~/Downloads Save [icon] [icon] [icon] [icon]
1 #include <stdio.h>
2 #include <mpi.h>
3
4 int main(int argc, char** argv) {
5     int size, rank;
6
7     MPI_Init(NULL, NULL);
8     MPI_Comm_size(MPI_COMM_WORLD, &size);
9     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
10
11     if(rank == 0){
12         int x = 9;
13         int y = 17;
14         MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
15         MPI_Send(&y, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
16     } else {
17         int number;
18         MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
19         printf("Process %d received %d\n", rank, number);
20     }
21
22     MPI_Finalize();
23
24     return 0;
25 }
```

OUTPUT:

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

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

A terminal window titled 'nayan@Nayan: ~/Downloads' showing the execution of 'mpiexec -n 2 ./mpi02'. The output shows 'Process 1 received 9' followed by a fatal error: 'Abort(939149830) on node 0 (rank 0 in comm 0): Fatal error in internal_Send: Invalid rank, error stack: internal_Send(120): MPI_Send(buf=0x7ffeedfd6060, count=1, MPI_INT, 2, 0, MPI_COMM_WORLD) failed internal_Send(78).: Invalid rank has value 2 but must be nonnegative and less than 2'. The prompt returns to 'nayan@Nayan:~/Downloads\$'.

```
nayan@Nayan:~/Downloads$ mpiexec -n 2 ./mpi02
Process 1 received 9
Abort(939149830) on node 0 (rank 0 in comm 0): Fatal error in internal_Send: Invalid rank, error stack:
internal_Send(120): MPI_Send(buf=0x7ffeedfd6060, count=1, MPI_INT, 2, 0, MPI_COMM_WORLD) failed
internal_Send(78).: Invalid rank has value 2 but must be nonnegative and less than 2
nayan@Nayan:~/Downloads$
```

A terminal window titled 'nayan@Nayan: ~/Downloads' showing the execution of 'mpiexec -n 3 ./mpi02'. The output shows 'Process 1 received 9' and 'Process 2 received 17'. The prompt returns to 'nayan@Nayan:~/Downloads\$'.

```
nayan@Nayan:~/Downloads$ mpiexec -n 3 ./mpi02
Process 1 received 9
Process 2 received 17
nayan@Nayan:~/Downloads$
```

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

A terminal window titled 'nayan@Nayan: ~/Downloads' showing the execution of 'mpiexec -n 4 ./mpi02'. The output shows 'Process 1 received 9' and 'Process 2 received 17'. The prompt returns to 'nayan@Nayan:~/Downloads\$'.

```
nayan@Nayan:~/Downloads$ mpiexec -n 4 ./mpi02
Process 1 received 9
Process 2 received 17
nayan@Nayan:~/Downloads$
```


- 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 program continues to run because only the first two processes receive messages from the process waiting for input from process 0, leaving the remaining processes waiting endlessly.

CODE:

```
1 /*
2  - Process 0 sends x to process 1 and y to processes 2. "1, MPI_INT"
3  indicates that the message consists of one integer.
4  - Processes other than rank 0 wait to receive a message using MPI_Recv.
5  The "0, 0" indicates that the message is expected from process 0 and
6  should have the tag 0. The result is stored in the number variable.
7 */
8
9 #include <stdio.h>
10 #include <mpi.h>
11
12 int main(int argc, char** argv) {
13     int size, rank;
14
15     MPI_Init(NULL, NULL);
16     MPI_Comm_size(MPI_COMM_WORLD, &size);
17     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
18
19     if(rank == 0) {
20         int x = 9;
21         int y = 17;
22         for(int dest = 1; dest < size; dest++) {
23             MPI_Send(&x, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
24             MPI_Send(&y, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
25         }
26     } else {
27         int number1, number2;
28         MPI_Recv(&number1, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
29         MPI_Recv(&number2, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
30         printf("Process %d received %d and %d\n", rank, number1, number2);
31     }
32
33     MPI_Finalize();
34
35     return 0;
36 }
```

OUTPUT:

```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpiexec -n 4 ./mpi02
Process 1 received 9 and 17
Process 2 received 9 and 17
Process 3 received 9 and 17
nayan@Nayan:~/Downloads$
```

- 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 end sup 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?


```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpicc mpi03.c -o mpi03
nayan@Nayan:~/Downloads$
```



ERROR:

```
nayan@Nayan: ~/Downloads
nayan@Nayan:~/Downloads$ mpiexec ./mpi03
Abort(469377030) on node 0 (rank 0 in comm 0): Fatal error in internal_Recv: Invalid rank, error stack:
internal_Recv(127): MPI_Recv(buf=0x7ffdf0380d74, count=1, MPI_INT, 1, 0, MPI_COMM_WORLD, status=0x1) failed
internal_Recv(78).: Invalid rank has value 1 but must be nonnegative and less than 1
nayan@Nayan:~/Downloads$
```

Process 0 is blocked and must wait for the sleep function to complete while it awaits a message from process 1 before proceeding to process 2.

By substituting non-blocking communication for blocking communication, the issue can be resolved.

CODE:

```
Open  mpi03.c  Save  -  +  x
~/Downloads

1 #include <stdio.h>
2 #include <mpi.h>
3 #include <unistd.h>
4
5 int main(int argc, char** argv) {
6     int size, rank;
7
8     MPI_Init(NULL, NULL);
9     MPI_Comm_size(MPI_COMM_WORLD, &size);
10    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11    if(size != 3) {
12        if(rank == 0) {
13            printf("This program needs to run on exactly 3 processes\n");
14        }
15    } else {
16        if(rank == 0){
17            int x, y;
18            MPI_Recv(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
19            printf("Received %d from process %d\n", x, 1);
20            MPI_Recv(&y, 1, MPI_INT, 2, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
21            printf("Received %d from process %d\n", y, 2);
22        } else {
23            if(rank == 1){
24                MPI_Request request;
25                int number = rank + 10;
26                MPI_Isend(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &request);
27                usleep(5000000);
28                MPI_Wait(&request, MPI_STATUS_IGNORE);
29            } else {
30                int number = rank + 10;
31                MPI_Send(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
32            }
33        }
34    }
35    MPI_Finalize();
36    return 0;
37 }
```

OUTPUT:

```
nayan@Nayan: ~/Downloads  Q  -  +  x

nayan@Nayan:~/Downloads$ mpiexec -n 3 ./mpi03
Received 11 from process 1
Received 12 from process 2
```

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.



CODE:

```
Open ▾ [icon] mpiPrimes.c ~/Downloads Save [menu] [minus] [square] [x]

1 #include <stdio.h>
2 #include <mpi.h>
3
4 int main(int argc, char **argv)
5 {
6     int i, c;
7     int nstart = 1, nfinish = 10000;
8     int rank, size;
9
10    MPI_Init(&argc, &argv);
11    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
12    MPI_Comm_size(MPI_COMM_WORLD, &size);
13
14    int chunk_size = (nfinish - nstart + 1) / size;
15    int start = nstart + rank * chunk_size;
16    int end = start + chunk_size - 1;
17    if (rank == size - 1) {
18        end = nfinish;
19    }
20
21    printf("Process %d: Prime numbers between %d and %d are:\n", rank, start, end);
22    for (i = start; i <= end; i++) {
23        for (c = 2; c <= i - 1; c++) {
24            if (i % c == 0) {
25                break;
26            }
27        }
28        if (c == i) {
29            printf("Process %d: %d\n", rank, i);
30        }
31    }
32
33    MPI_Finalize();
34    return 0;
35 }
```

Bracket match found on line: 5 C ▾ Tab Width: 8 ▾ Ln 35, Col 2 ▾ INS

OUTPUT:

```
[icon] nayan@Nayan: ~/Downloads [search] [menu] [minus] [square] [x]

nayan@Nayan:~/Downloads$ mpiexec ./mpiPrimes
Process 0: Prime numbers between 1 and 10000 are:
Process 0: 2
Process 0: 3
Process 0: 5
Process 0: 7
Process 0: 11
Process 0: 13
.
.
Process 0: 9967
Process 0: 9973
nayan@Nayan:~/Downloads$
```