Assignment #02

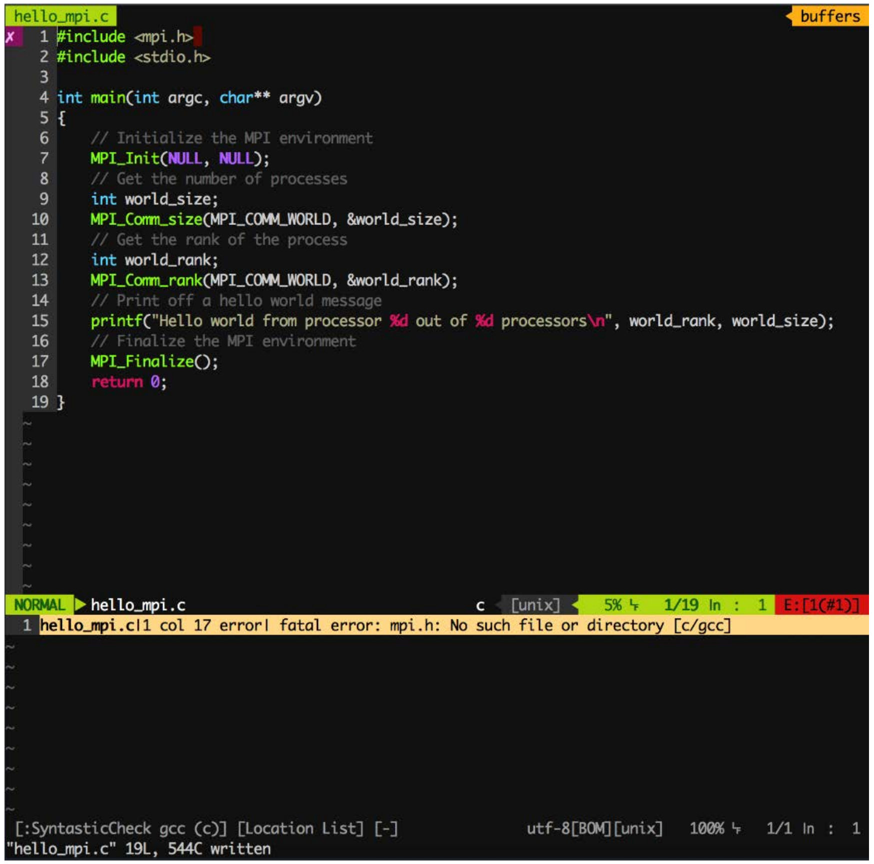
Sep, 25, 2017
JinYeong Wang
Parallel Computing Lab.
Mechanical Engineering
Hanyang University

How to install OpenMPI in UBUNTU 16.04

Install OpenMPI

- sudo apt install openmpi-common openmpi-bin libopenmpi-dev -y

vim hello_mpi.c



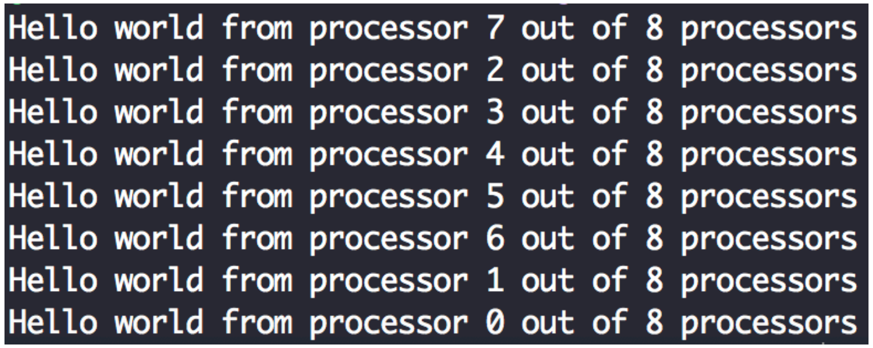
```
hello_mpi.c                                                                      buffers
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv)
{
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);
    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    // Print off a hello world message
    printf("Hello world from processor %d out of %d processors\n", world_rank, world_size);
    // Finalize the MPI environment
    MPI_Finalize();
    return 0;
}
NORMAL  hello_mpi.c                                         c  [unix]   5%   1/19 ln :  1  E:[1(#1)]
  1 hello_mpi.c|1 col 17 error| fatal error: mpi.h: No such file or directory [c/gcc]
 [:SyntasticCheck gcc (c)] [Location List] [-]            utf-8[BOM][unix]   100%   1/1 ln :  1
"hello_mpi.c" 19L, 544C written
```

Hello world using MPI

Compile and Run

- mpicc hello_mpi.c –o hello_mpi.o
- mpirun -np 8 ./hello_mpi.o

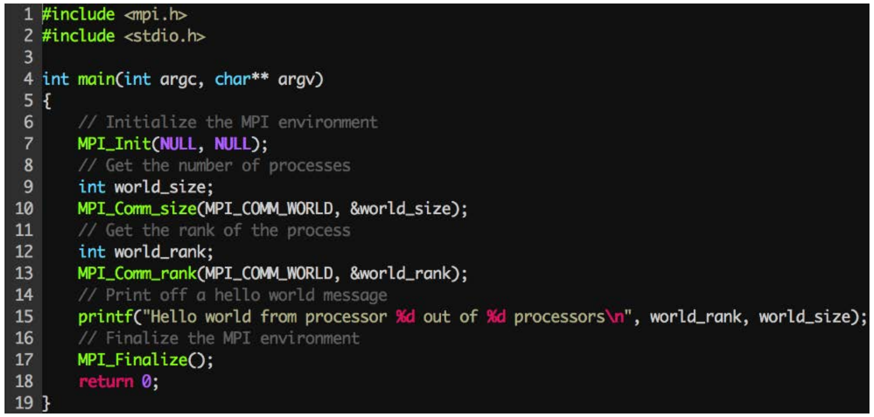
Result



```
Hello world from processor 7 out of 8 processors
Hello world from processor 2 out of 8 processors
Hello world from processor 3 out of 8 processors
Hello world from processor 4 out of 8 processors
Hello world from processor 5 out of 8 processors
Hello world from processor 6 out of 8 processors
Hello world from processor 1 out of 8 processors
Hello world from processor 0 out of 8 processors
```

Investigate MPI functions used in hello.c

Whole Code



```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv)
{
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);
    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    // Print off a hello world message
    printf("Hello world from processor %d out of %d processors\n", world_rank, world_size);
    // Finalize the MPI environment
    MPI_Finalize();
    return 0;
}
```

MPI functions

- MPI_Init(NULL, NULL);
  - Initialize the MPI environment
- MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  - Get the number of processes
- MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  - Get the rank of the process
- MPI_Finalize();
  - Finalize the MPI environment