National University
of computer and emerging sciences

FAST Islamabad Campus

Assignment #2 Report:

Interprocess Communication Using Threads

CS2006 – Operating Systems

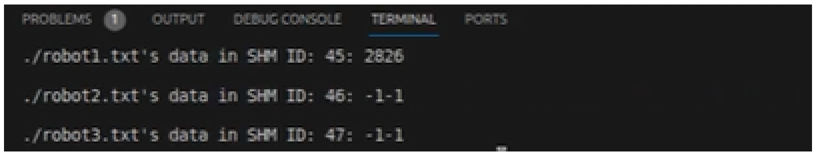
Problem statement:

The problem basically requires proximity based communication between 4 robots in an imaginary plane. Each robot in this plane has specific x and y coordinates, each robot has a certain distance from each other based on their relative coordinates. The robots should now be able to freely move about the plane and additionally be able to listen continuously for any coordinate updates in its surroundings. So that if appropriate the robot itself should decide acknowledgement is necessary. Also, this has been updated according to our professor's requirements where he requires the robots to communicate the messages in real time.

Solution: (with process threads)

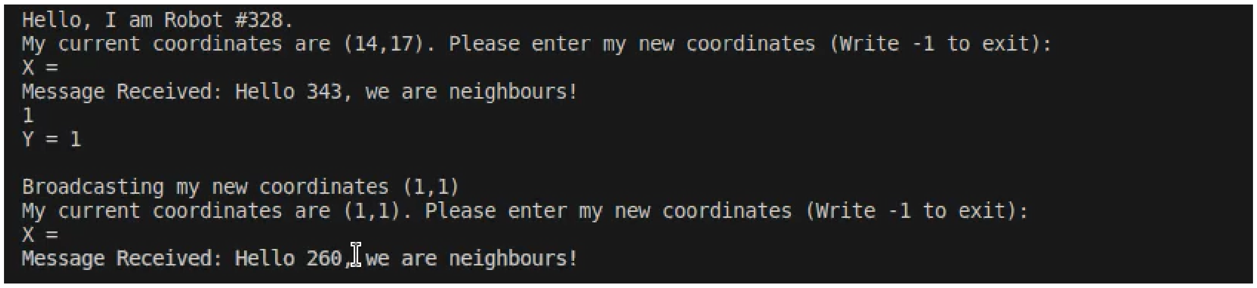
We define that there are 4 robots , but along with this we also instantiate 4 files for each robot. These files are used as individual shared memory segments for each robot process. This gives them the facility to store their own coordinates and in addition to this, read the coordinates of all other robots in the plane.

These shared memory segments allow for updation of a robot's own coordinates and for reading of other coordinates. Each robot is initialized with its own reading and writing threads. This is what allows the real time transmission and reading of coordinates data from other robots. This means that the reading thread is perpetually reading from the shared memory. Here , it is important to mention that all the robot id's in the shared memory are initialized with -1 values to indicate them in a dead state. So as each robot is initialized it updates with random coordinates initially. Below is what's in the shared memory state (from robot 0's perspective) once robot 1 was initialized with a random coordinate:



```
./robot1.txt's data in SHM ID: 45: 2826
./robot2.txt's data in SHM ID: 46: -1-1
./robot3.txt's data in SHM ID: 47: -1-1
```

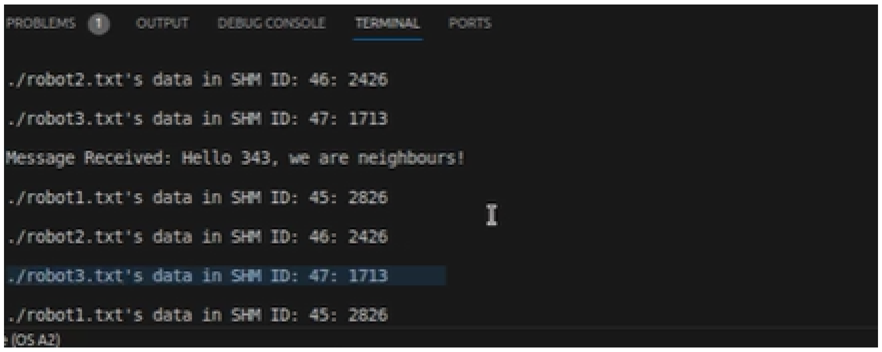
A mechanism to highlight how each robot is handling the reading writing sequence is to realize that whilst it may be perpetually reading , it only does so whilst the robot is in a writing sequence. So, essentially it will keep reading until we have stopped writing coordinates. This can be observed in an excerpt below:



```
Hello, I am Robot #328.
My current coordinates are (14,17). Please enter my new coordinates (Write -1 to exit):
X =
Message Received: Hello 343, we are neighbours!
1
Y = 1

Broadcasting my new coordinates (1,1)
My current coordinates are (1,1). Please enter my new coordinates (Write -1 to exit):
X =
Message Received: Hello 260, we are neighbours!
```

Another important thing to note is that the messages are meant to only be sent once, even though we are perpetually reading the shared memory segment. This concept of acknowledgement amongst the processes can be best captured here in another excerpt from robot 0:



```
./robot2.txt's data in SHM ID: 46: 2426

./robot3.txt's data in SHM ID: 47: 1713

Message Received: Hello 343, we are neighbours!

./robot1.txt's data in SHM ID: 45: 2826

./robot2.txt's data in SHM ID: 46: 2426

./robot3.txt's data in SHM ID: 47: 1713

./robot1.txt's data in SHM ID: 45: 2826
```

Here we can notice that as soon as it reads the new coordinates of robot3- and determines via euclidean distance that they are neighbors it will only receive a single message from robot3. This can be proven as in the next pass of reading the coordinates from shared memory robot3's coordinates don't change and no new message passing occurs.

Further details are mentioned in the video