Assignment 7: Inter process communication in Linux using Pipes: Full duplex communication between parent and child processes. Parent process writes a pathname of a file (the contents of the file are desired) on one pipe to be read by the child process and the child process writes the contents of the file on the second pipe to be read by the parent process and displays on standard output.

Name:	**Chaitanya Suresh Uge**
Roll Number:	**333064**
GR Number:	**21910718**
Division: **C**	**Batch:** **C3**

Theory:

Two Way Communication using Pipes :

Pipe Communication is possible only in one way that is either process1 can write and process2 can read or vice versa. So to achieve this we use 2 pipes so that with 1 pipe process 1 can read and with pipe 2 process 2 can write, and with pipe 1 and read with pipe 2.

Pseudo code:

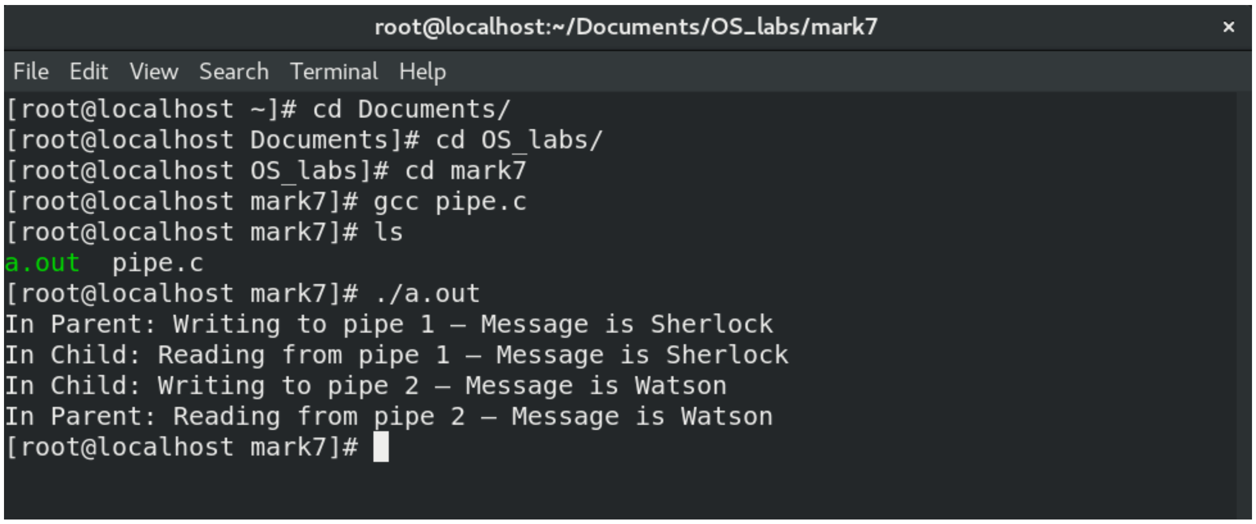
```
- Create two pipes. The first one is for the parent to write and child to read,
  say as pipe1. The second one is for the child to write and the parent to
  read, say as pipe2.
- Create a child process.
- Close unwanted ends as only one end is needed for each
  communication.
- Close unwanted ends in the parent process, read the end of pipe1, and
  write the end of pipe2.
- Close the unwanted ends in the child process, write the end of pipe1, and
  read the end of pipe2.
- Perform the communication as required.
```

Source Code:

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

int main() {
  int pipefds1[2], pipefds2[2];
  int returnstatus1, returnstatus2;
  int pid;
  char pipe1writemessage[20] = "charli";
  char pipe2writemessage[20] = "Bravo";
  char readmessage[20];
  returnstatus1 = pipe(pipefds1);

  if (returnstatus1 == -1) {
        printf("Unable to create pipe 1 \n");
        return 1;
  }
  returnstatus2 = pipe(pipefds2);

  if (returnstatus2 == -1) {
        printf("Unable to create pipe 2 \n");
        return 1;
```

```
   }
   pid = fork();
   if (pid != 0) {
      close(pipefds1[0]); // Close the unwanted pipe1 read side
      close(pipefds2[1]); // Close the unwanted pipe2 write side
      printf("In Parent: Writing to pipe 1 – Message is %s\n", pipe1writemessage);
      write(pipefds1[1], pipe1writemessage, sizeof(pipe1writemessage));
      read(pipefds2[0], readmessage, sizeof(readmessage));
      printf("In Parent: Reading from pipe 2 – Message is %s\n", readmessage);
   } else {
      close(pipefds1[1]); // Close the unwanted pipe1 write side
      close(pipefds2[0]); // Close the unwanted pipe2 read side
      read(pipefds1[0], readmessage, sizeof(readmessage));
      printf("In Child: Reading from pipe 1 – Message is %s\n", readmessage);
      printf("In Child: Writing to pipe 2 – Message is %s\n", pipe2writemessage);
      write(pipefds2[1], pipe2writemessage, sizeof(pipe2writemessage));
   }
   return 0;
}
```

OUTPUT:


root@localhost:~/Documents/OS_labs/mark7
File Edit View Search Terminal Help

```
[root@localhost ~]# cd Documents/
[root@localhost Documents]# cd OS_labs/
[root@localhost OS_labs]# cd mark7
[root@localhost mark7]# gcc pipe.c
[root@localhost mark7]# ls
a.out  pipe.c
[root@localhost mark7]# ./a.out
In Parent: Writing to pipe 1 – Message is Sherlock
In Child: Reading from pipe 1 – Message is Sherlock
In Child: Writing to pipe 2 – Message is Watson
In Parent: Reading from pipe 2 – Message is Watson
[root@localhost mark7]# 
```