OPERATING SYSTEM

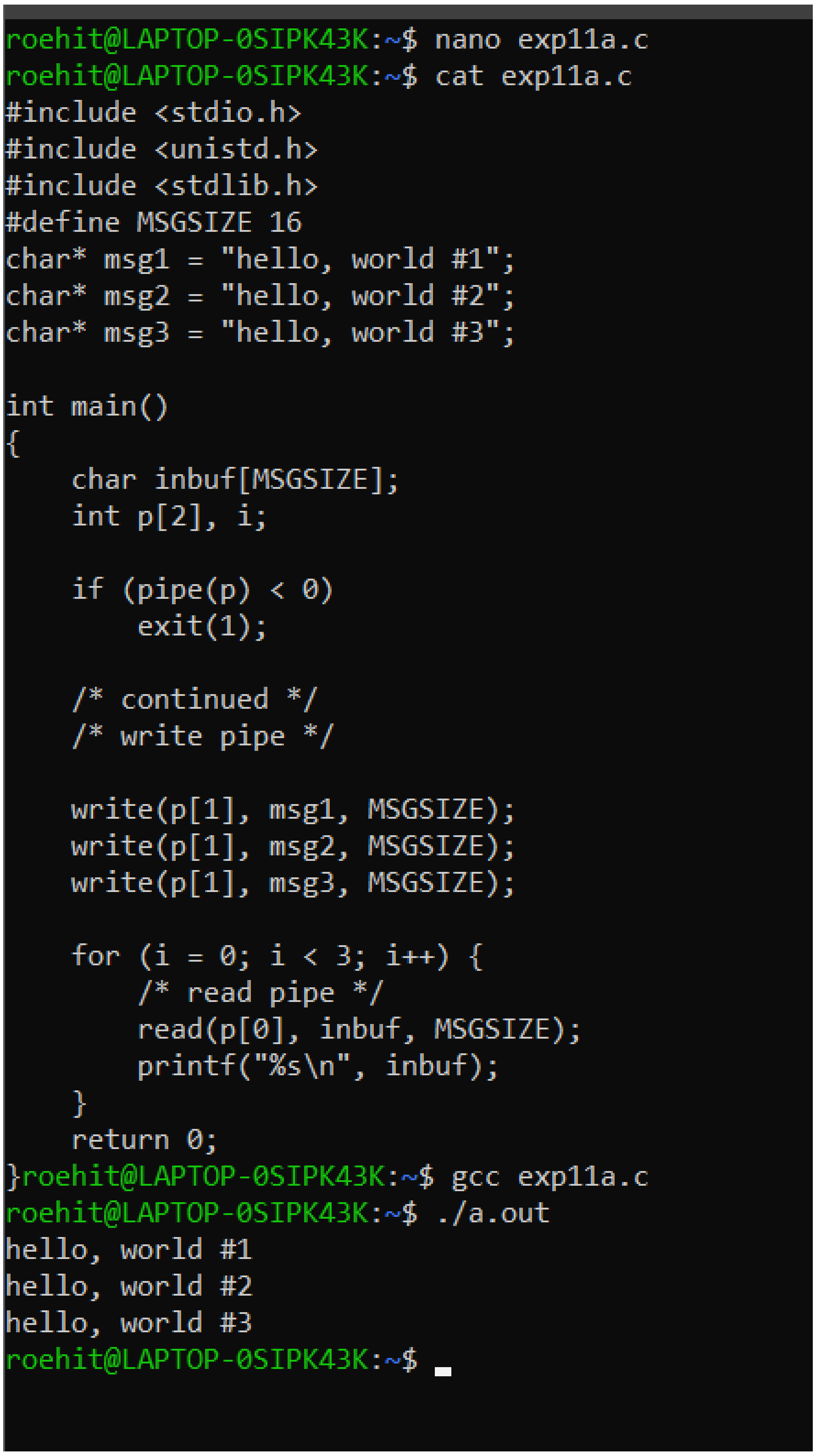
Experiment 11 Pipe System Call

L2 -SWE

ROEHIT RANGANATHAN

RA1911033010017

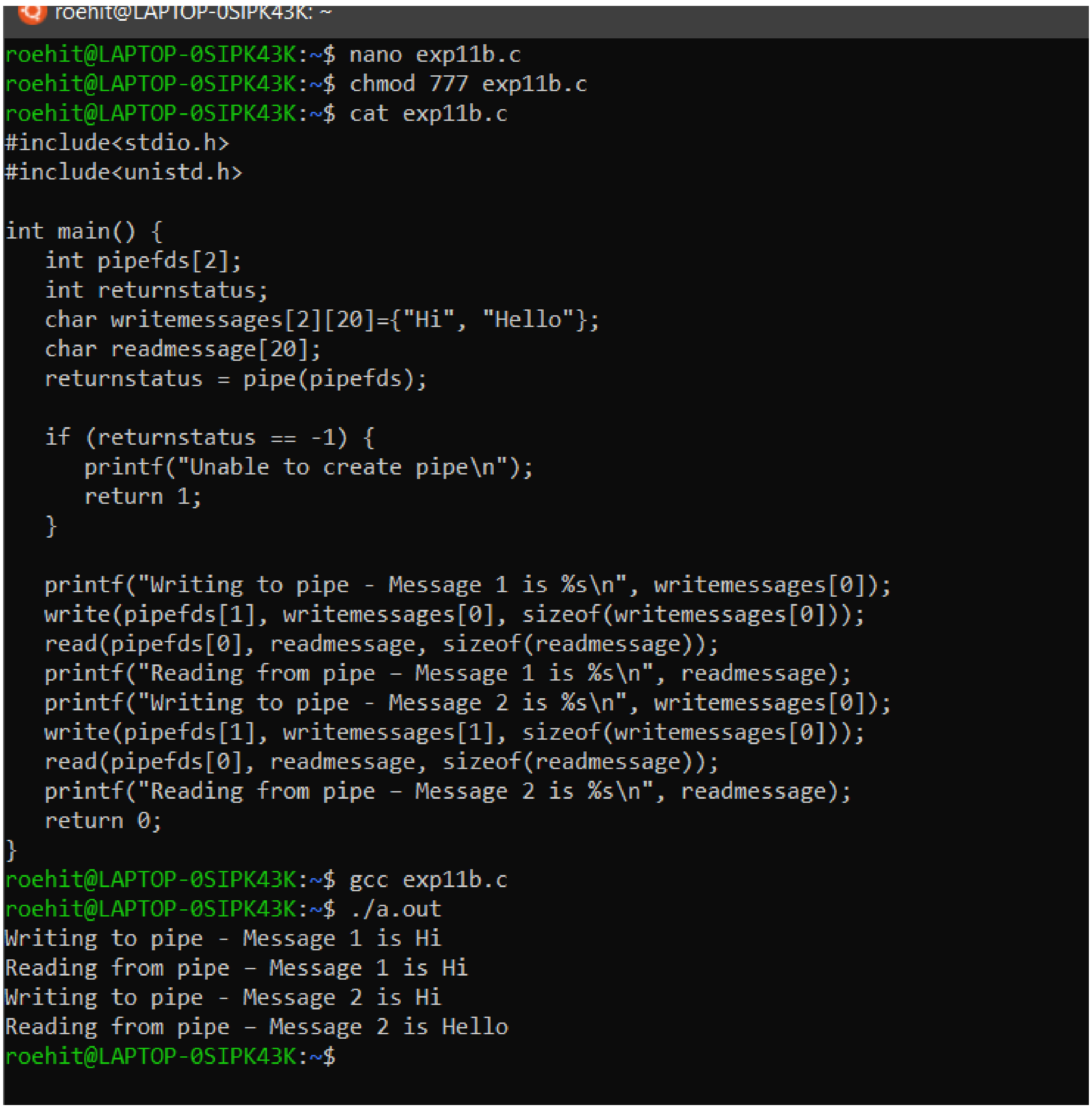
1. Pipe using FIFO.
Ref. screenshot:



```
roehit@LAPTOP-0SIPK43K:~$ nano exp11a.c
roehit@LAPTOP-0SIPK43K:~$ cat exp11a.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#define MSGSIZE 16
char* msg1 = "hello, world #1";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";

int main()
{
    char inbuf[MSGSIZE];
    int p[2], i;

    if (pipe(p) < 0)
        exit(1);

    /* continued */
    /* write pipe */

    write(p[1], msg1, MSGSIZE);
    write(p[1], msg2, MSGSIZE);
    write(p[1], msg3, MSGSIZE);

    for (i = 0; i < 3; i++) {
        /* read pipe */
        read(p[0], inbuf, MSGSIZE);
        printf("%s\n", inbuf);
    }
    return 0;
}roehit@LAPTOP-0SIPK43K:~$ gcc exp11a.c
roehit@LAPTOP-0SIPK43K:~$ ./a.out
hello, world #1
hello, world #2
hello, world #3
roehit@LAPTOP-0SIPK43K:~$
```

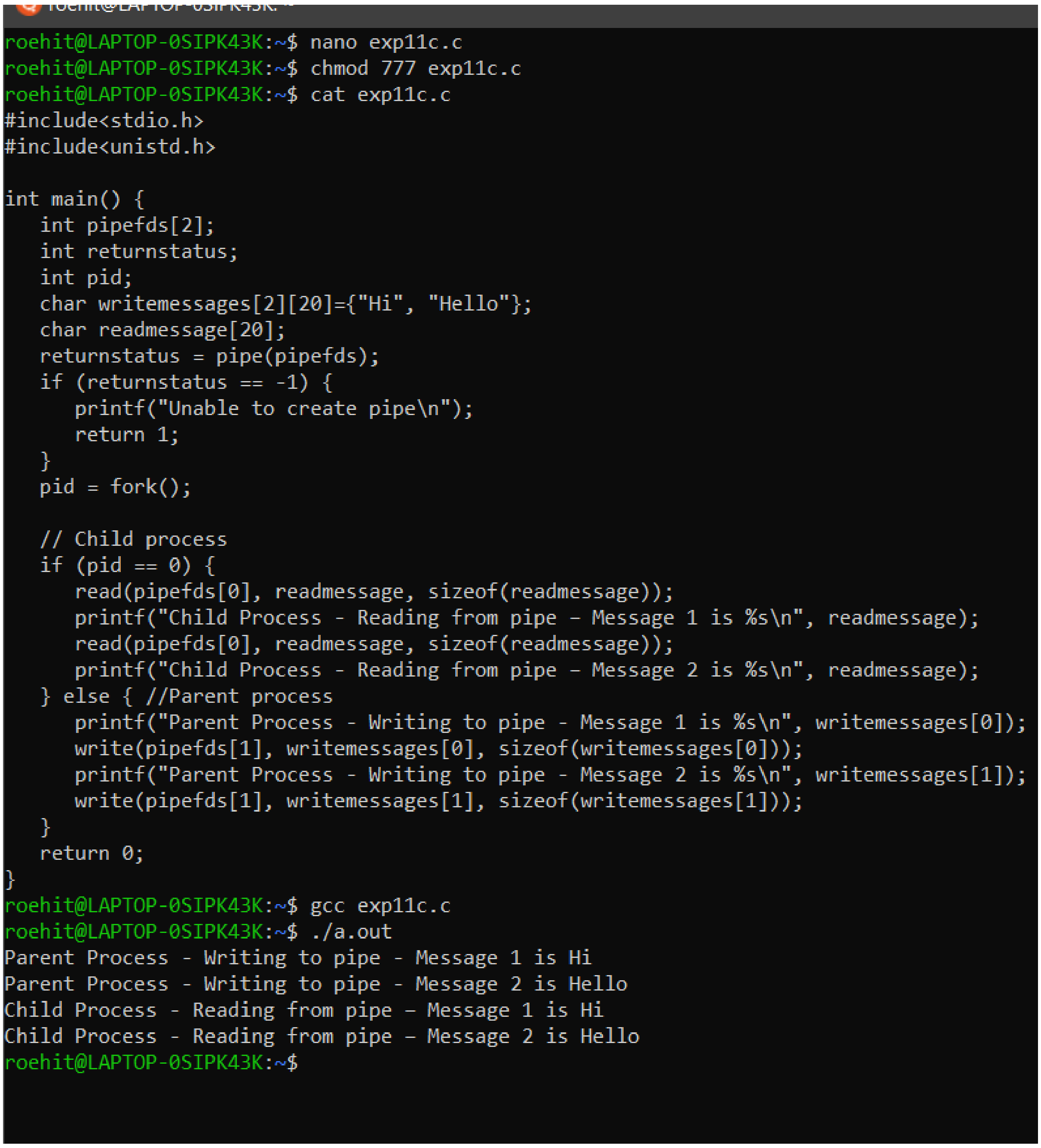
2. Program to write and read two messages using pipe.

Ref. screenshot:



```
roehit@LAPTOP-0SIPK43K: ~
roehit@LAPTOP-0SIPK43K:~$ nano exp11b.c
roehit@LAPTOP-0SIPK43K:~$ chmod 777 exp11b.c
roehit@LAPTOP-0SIPK43K:~$ cat exp11b.c
#include<stdio.h>
#include<unistd.h>

int main() {
   int pipefds[2];
   int returnstatus;
   char writemessages[2][20]={"Hi", "Hello"};
   char readmessage[20];
   returnstatus = pipe(pipefds);

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

   printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);
   write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
   read(pipefds[0], readmessage, sizeof(readmessage));
   printf("Reading from pipe – Message 1 is %s\n", readmessage);
   printf("Writing to pipe - Message 2 is %s\n", writemessages[0]);
   write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
   read(pipefds[0], readmessage, sizeof(readmessage));
   printf("Reading from pipe – Message 2 is %s\n", readmessage);
   return 0;
}
roehit@LAPTOP-0SIPK43K:~$ gcc exp11b.c
roehit@LAPTOP-0SIPK43K:~$ ./a.out
Writing to pipe - Message 1 is Hi
Reading from pipe – Message 1 is Hi
Writing to pipe - Message 2 is Hi
Reading from pipe – Message 2 is Hello
roehit@LAPTOP-0SIPK43K:~$
```

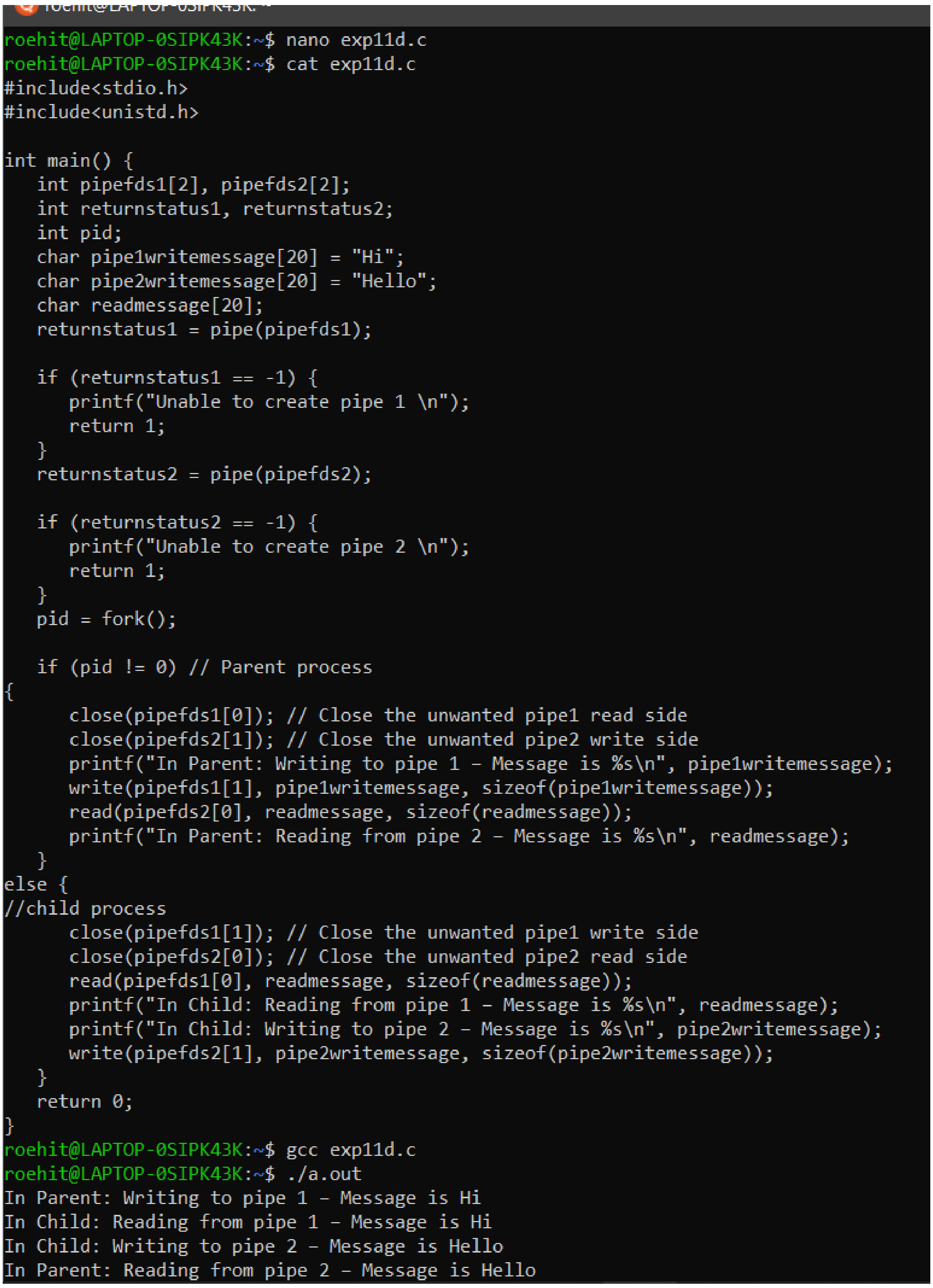
3. Program to write and read two messages through the pipe using the parent and the child processes.
Ref. screenshot



```
roehit@LAPTOP-0SIPK43K:~$ nano exp11c.c
roehit@LAPTOP-0SIPK43K:~$ chmod 777 exp11c.c
roehit@LAPTOP-0SIPK43K:~$ cat exp11c.c
#include<stdio.h>
#include<unistd.h>

int main() {
   int pipefds[2];
   int returnstatus;
   int pid;
   char writemessages[2][20]={"Hi", "Hello"};
   char readmessage[20];
   returnstatus = pipe(pipefds);
   if (returnstatus == -1) {
      printf("Unable to create pipe\n");
      return 1;
   }
   pid = fork();

   // Child process
   if (pid == 0) {
      read(pipefds[0], readmessage, sizeof(readmessage));
      printf("Child Process - Reading from pipe – Message 1 is %s\n", readmessage);
      read(pipefds[0], readmessage, sizeof(readmessage));
      printf("Child Process - Reading from pipe – Message 2 is %s\n", readmessage);
   } else { //Parent process
      printf("Parent Process - Writing to pipe - Message 1 is %s\n", writemessages[0]);
      write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
      printf("Parent Process - Writing to pipe - Message 2 is %s\n", writemessages[1]);
      write(pipefds[1], writemessages[1], sizeof(writemessages[1]));
   }
   return 0;
}
roehit@LAPTOP-0SIPK43K:~$ gcc exp11c.c
roehit@LAPTOP-0SIPK43K:~$ ./a.out
Parent Process - Writing to pipe - Message 1 is Hi
Parent Process - Writing to pipe - Message 2 is Hello
Child Process - Reading from pipe – Message 1 is Hi
Child Process - Reading from pipe – Message 2 is Hello
roehit@LAPTOP-0SIPK43K:~$
```

4. Program to write achieve two-way communication using pipes.

Ref. screenshot:



```
roehit@LAPTOP-0SIPK43K:~$ nano exp11d.c
roehit@LAPTOP-0SIPK43K:~$ cat exp11d.c
#include<stdio.h>
#include<unistd.h>

int main() {
   int pipefds1[2], pipefds2[2];
   int returnstatus1, returnstatus2;
   int pid;
   char pipe1writemessage[20] = "Hi";
   char pipe2writemessage[20] = "Hello";
   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) // Parent process
{
      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 {
//child process
      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;
}
roehit@LAPTOP-0SIPK43K:~$ gcc exp11d.c
roehit@LAPTOP-0SIPK43K:~$ ./a.out
In Parent: Writing to pipe 1 – Message is Hi
In Child: Reading from pipe 1 – Message is Hi
In Child: Writing to pipe 2 – Message is Hello
In Parent: Reading from pipe 2 – Message is Hello
```