Assignment 06 | Operating System CE-092

Assignment submission for Operating System subject week 6.

nevilparmar24@gmail.com

Aim - **Inter process communication (Use of pipe system call and mkfifo)**

Pipe system call:

#include <unistd.h>

int pipe(int pipefd[2]);

pipe() creates a pipe, a unidirectional data channel that can be used for inter process communication.

The array pipefd is used to return two file descriptors referring to the ends of the pipe.

pipefd[0] refers to the read end of the pipe.

pipefd[1] refers to the write end of the pipe.

Data written to the write end of the pipe is buffered by the kernel until it is read from the read end of the pipe.

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

Close system call:

#include <unistd.h>

Int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused.

Any record locks held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock).

close() returns zero on success. On error, -1 is returned, and errno is set appropriately.

mkfifo system call: **make a fifo special file (named pipe)**

#include <sys/types.h>

#include<sys/stat.h>

Int mkfifo(const char * pathname, mode_t mode);

#include<fcntl.h>

mkfifo() makes a FIFO special file with name *pathname*.
A FIFO special file is similar to a pipe, except that it is created in a different way.
Instead of being an anonymous communications channel, a FIFO special file is entered into the filesystem by calling mkfifo().

On success mkfifo returns 0 , on error -1 is returned.

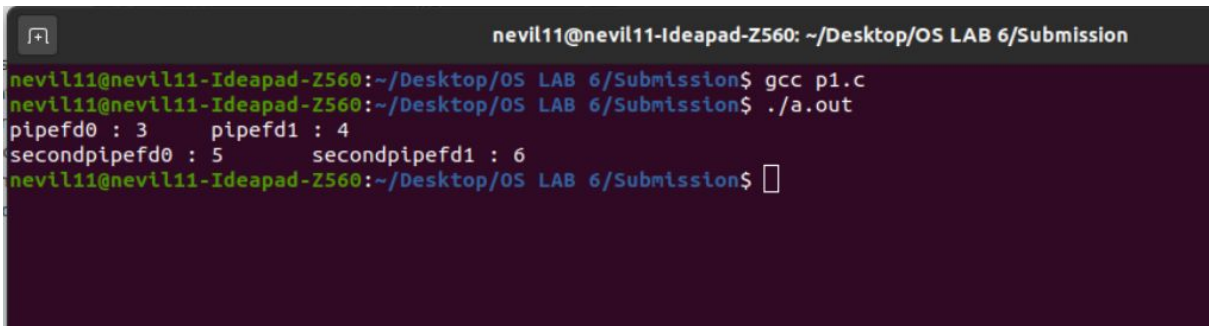
Task 1:

Write a program to create a pipe and print the values of pipe file descriptors.

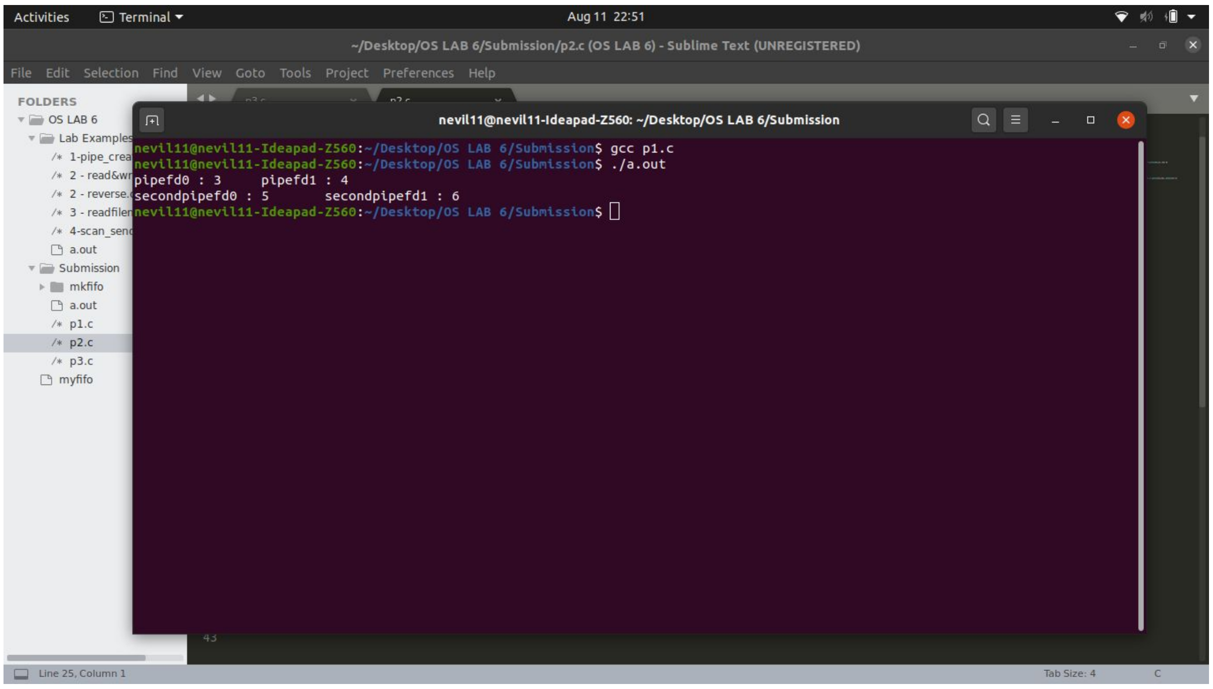
Code:

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

int main()
{
    int pipefd[2];
    int secondpipefd[2];
    int err = pipe(pipefd);
    if(err!=0)
        printf("Error creating a file.\n");
    else
        printf("pipefd0 : %d\tpipefd1 : %d\n",
pipefd[0],pipefd[1]);

    int err1 = pipe(secondpipefd);
    if(err1!=0)
        printf("Error creating a file.\n");
    else
        printf("secondpipefd0 : %d\tsecondpipefd1 :
%d\n", secondpipefd[0],secondpipefd[1]);


    return 0;
}

/*

Output:
3 4
5 6

because 0 1 2 are stdin stdout stderr
so they are reserverd and other can be accessed by the
process.
although, this values are not fixed for all the time.
System might take appropriate action and can assign the
avaialble resources.i.e pipes here.
*/
```

Output:



```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p1.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out
pipefd0 : 3     pipefd1 : 4
secondpipefd0 : 5       secondpipefd1 : 6
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```


Activities
Terminal
Aug 11 22:51
~/Desktop/OS LAB 6/Submission/p2.c (OS LAB 6) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
OS LAB 6
Lab Examples
1-pipe_crea
2 - read&wr
2 - reverse.
3 - readfile
4-scan_send
a.out
Submission
mkfifo
a.out
p1.c
p2.c
p3.c
myfifo
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 6/Submission
Line 25, Column 1
Tab Size: 4
C

```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p1.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out
pipefd0 : 3     pipefd1 : 4
secondpipefd0 : 5       secondpipefd1 : 6
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```

Task 2:

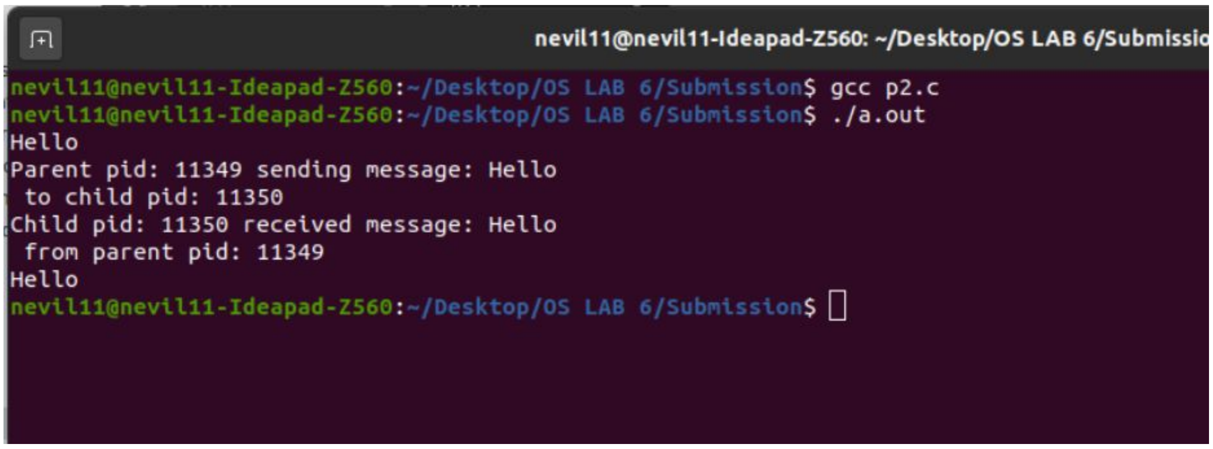
Write a program to pass a message from parent process to child process through a pipe..

Code:

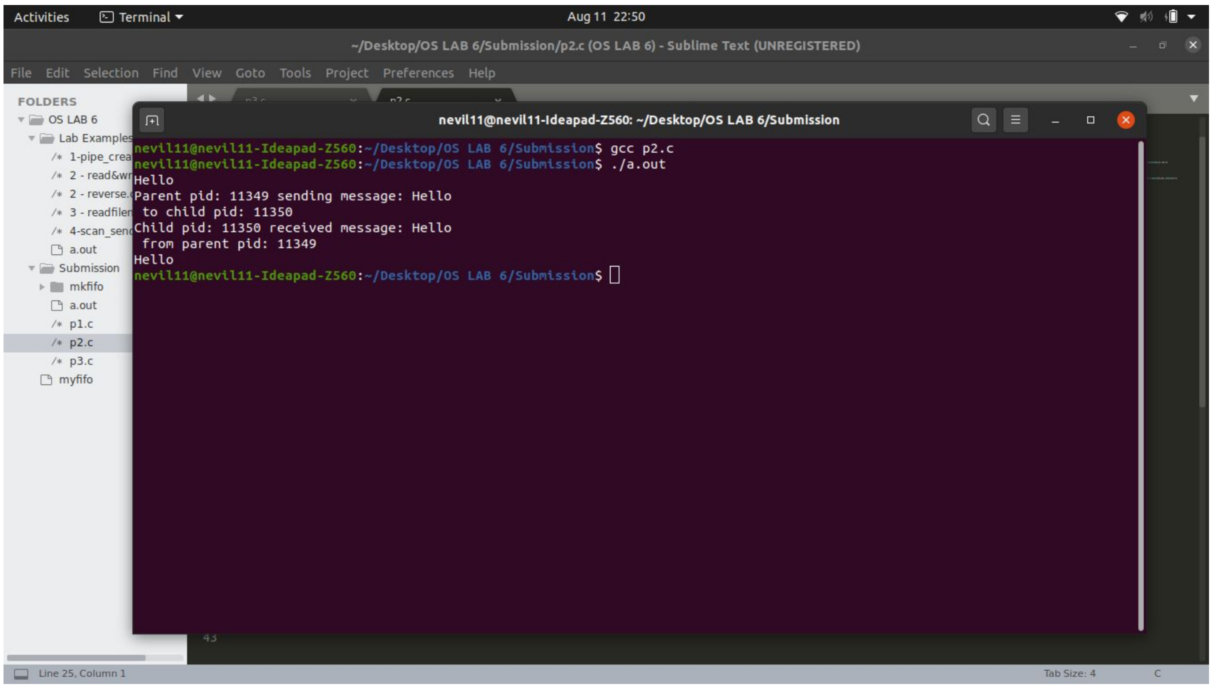
```
#include<unistd.h>
#include<stdio.h>

int main()
{
    int pid;
    int pipefd[2];
    int err = pipe(pipefd);

    if(err!=0)
    {
        write(1,"error",5);
    }

    pid = fork();

    if(pid == -1)
        printf("Process creation error\n");

    else if(pid > 0)
    {
        //parent part
        close(pipefd[0]);
        char ch[1000] = {'\0'};
        read(0,ch,1000);
        printf("Parent pid: %d sending message: %s to
child pid: %d \n",getpid(),ch, pid );
        write(pipefd[1],ch,1000);
    }
    else
    {
        //child part
        close(pipefd[1]);
        char ch2[1000] = {'\0'};
        int n = read(pipefd[0],ch2,1000);
        printf("Child pid: %d received message: %s from
parent pid: %d \n",getpid(),ch2, getppid() );
```

```
        write(1,ch2,1000);
    }

    return 0;
}
```

Output:



```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p2.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out
Hello
Parent pid: 11349 sending message: Hello
 to child pid: 11350
Child pid: 11350 received message: Hello
 from parent pid: 11349
Hello
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```



Activities
Terminal
Aug 11 22:50
~/Desktop/OS LAB 6/Submission/p2.c (OS LAB 6) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
OS LAB 6
Lab Examples
1-pipe_crea
2 - read&wr
2 - reverse.
3 - readfile
4-scan_send
a.out
Submission
mkfifo
a.out
p1.c
p2.c
p3.c
myfifo
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 6/Submission
Line 25, Column 1
Tab Size: 4
C

```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p2.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out
Hello
Parent pid: 11349 sending message: Hello
 to child pid: 11350
Child pid: 11350 received message: Hello
 from parent pid: 11349
Hello
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```

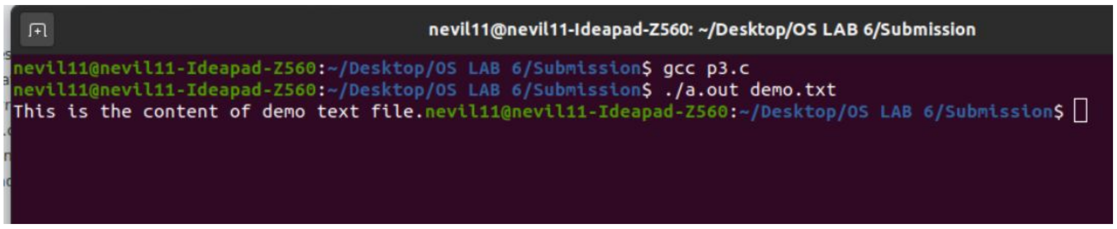
Task 3:

Write a program to pass file name from parent process to child process through a pipe, child process should pass the file contents to parent process and parent should print the contents

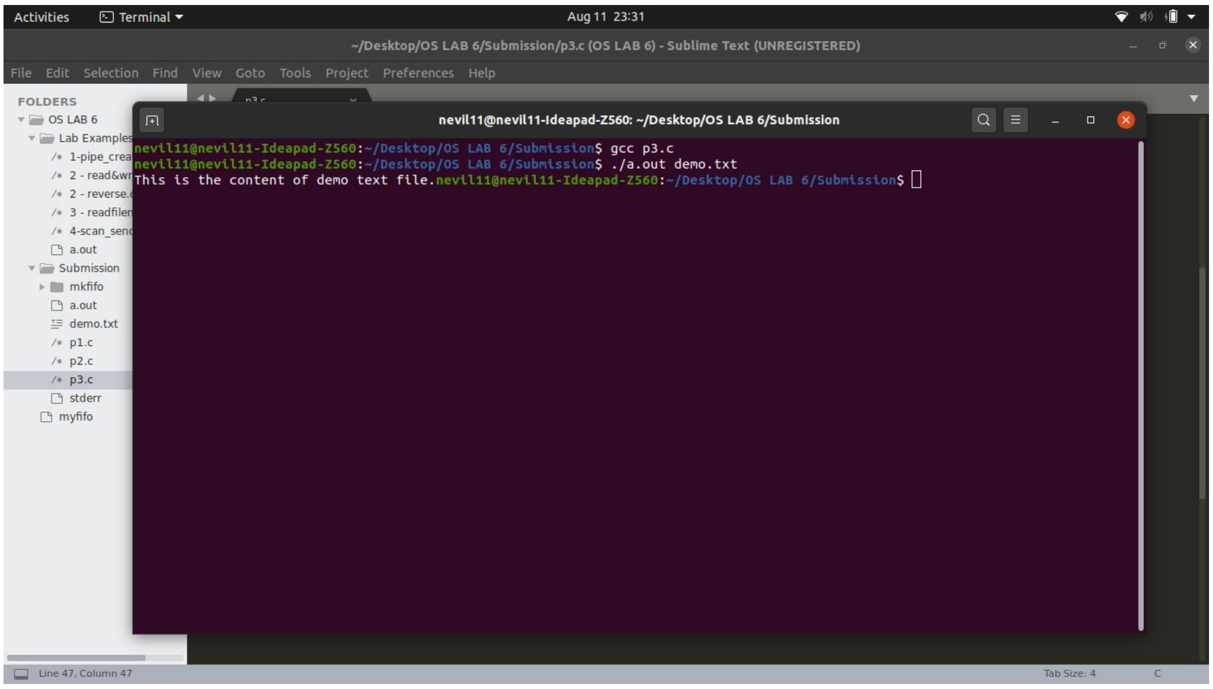
Code:

```
// parent send filename
// child send contents
// parent prints contents

#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#include<stdlib.h>
#include<sys/wait.h>

int main(int argc,char* argv[])
{
    int pipefd[2],pipefd2[2];
    pipe(pipefd);
    pipe(pipefd2);
    int pid = fork();
    char buffer[100];


    if(pid == -1)
    {
        printf("Error\n");
    }
    else if(pid > 0)
    {
        close(pipefd[0]);
        close(pipefd2[1]);
        //parent part
        write(pipefd[1],argv[1],sizeof(argv[1]));
        wait(NULL);
        bzero(buffer,sizeof(buffer));
        int m = read(pipefd2[0],buffer,sizeof(buffer));
        write(1,buffer,m);
    }
    else
    {

        close(pipefd[1]);
        close(pipefd2[0]);

        //child part
        bzero(buffer,sizeof(buffer));
        read(pipefd[0],buffer,sizeof(buffer));


        int fd = open(buffer,O_RDONLY);
        int n = read(fd,buffer,sizeof(buffer));
        write(pipefd2[1],buffer,n);

    }
    return 0;
}
```

Output:


nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 6/Submission

```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p3.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out demo.txt
This is the content of demo text file.nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```



```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ gcc p3.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ ./a.out demo.txt
This is the content of demo text file.nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission$ 
```

Task 4:

Demonstrate the use of named pipe with appropriate programs and commands.

Code:

Prog1.c

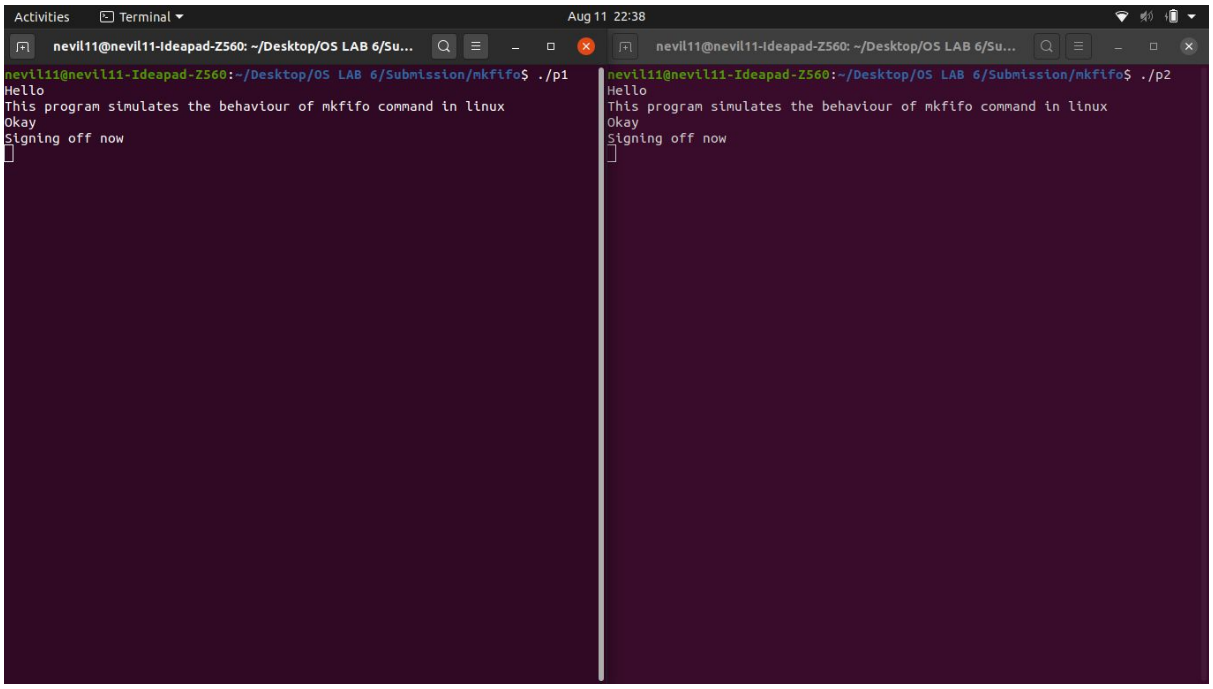
```c
/// Prog 1 to mimic the behaviour of mkfifo in linux
// It reads input from the user and write it into the
created pipe
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include<stdio.h>

int main()
{
    int fd;
    // FIFO file path in the same directory of the
program
    char * myfifo = "./customFifo";

    // Creating the named file(FIFO)
    mkfifo(myfifo, 0666);

    char  buffer[80];
    while (1)
    {
        // Open FIFO for write only
        fd = open(myfifo, O_WRONLY);

        // Take an input buffering from user.
        // 80 is maximum length
        fgets(buffer, 80, stdin);

        // Write the input buffering on FIFO
        // and close it
        write(fd, buffer, strlen(buffer)+1);
        close(fd);
    }
    return 0;
}
```

Prog2.c

```
// Prog 1 to mimic the behaviour of mkfifo in linux
// It reads from the created pipe by the first program
and outputs on the terminal
#include <string.h>
```

```c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include<stdio.h>

int main()
{
    int fd;

    // FIFO file path in the same directory of the prog
    char * myfifo = "./customFifo";

    // Creating the named file(FIFO)
    mkfifo(myfifo, 0666);

    char buffer[80];
    while (1)
    {
        // First open in read only and read
        fd = open(myfifo,O_RDONLY);
        read(fd, buffer, 80);

        // Print the read string and close
        printf("%s", buffer);
        close(fd);

    }
    return 0;
}
```

Output


Activities
Terminal
Aug 11 22:38
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 6/Su...
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 6/Su...

```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission/mkfifo$ ./p1
Hello
This program simulates the behaviour of mkfifo command in linux
Okay
Signing off now
```

```
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 6/Submission/mkfifo$ ./p2
Hello
This program simulates the behaviour of mkfifo command in linux
Okay
Signing off now
```

Nevil Parmar
CE-092
https://nevilparmar.me