Named Pipe or FIFO with example C program

In computing, a named pipe (also known as a **FIFO**) is one of the methods for intern-process communication.

- It is an extension to the traditional pipe concept on Unix. A traditional pipe is "unnamed" and lasts only as long as the process.
- A named pipe, however, can last as long as the system is up, beyond the life of the process. It can be deleted if no longer used.
- Usually a named pipe appears as a file and generally processes attach to it for inter-process communication. A FIFO file is a special kind of file on the local storage which allows two or more processes to communicate with each other by reading/writing to/from this file.
- A FIFO special file is entered into the filesystem by calling *mkfifo()* in C. Once we have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at

both ends simultaneously before you can proceed to do any input or output operations on it.

Creating a FIFO file: In order to create a FIFO file, a function calls i.e. mkfifo is used.

```
int mkfifo(const char *pathname, mode_t mode);
```

mkfifo() makes a FIFO special file with name ***pathname***. Here ***mode*** specifies the FIFO's permissions. It is modified by the process's umask in the usual way: the permissions of the created file are (mode & ~umask).

Using FIFO: As named pipe(FIFO) is a kind of file, we can use all the system calls associated with it i.e. *open*, *read*, *write*, *close*.

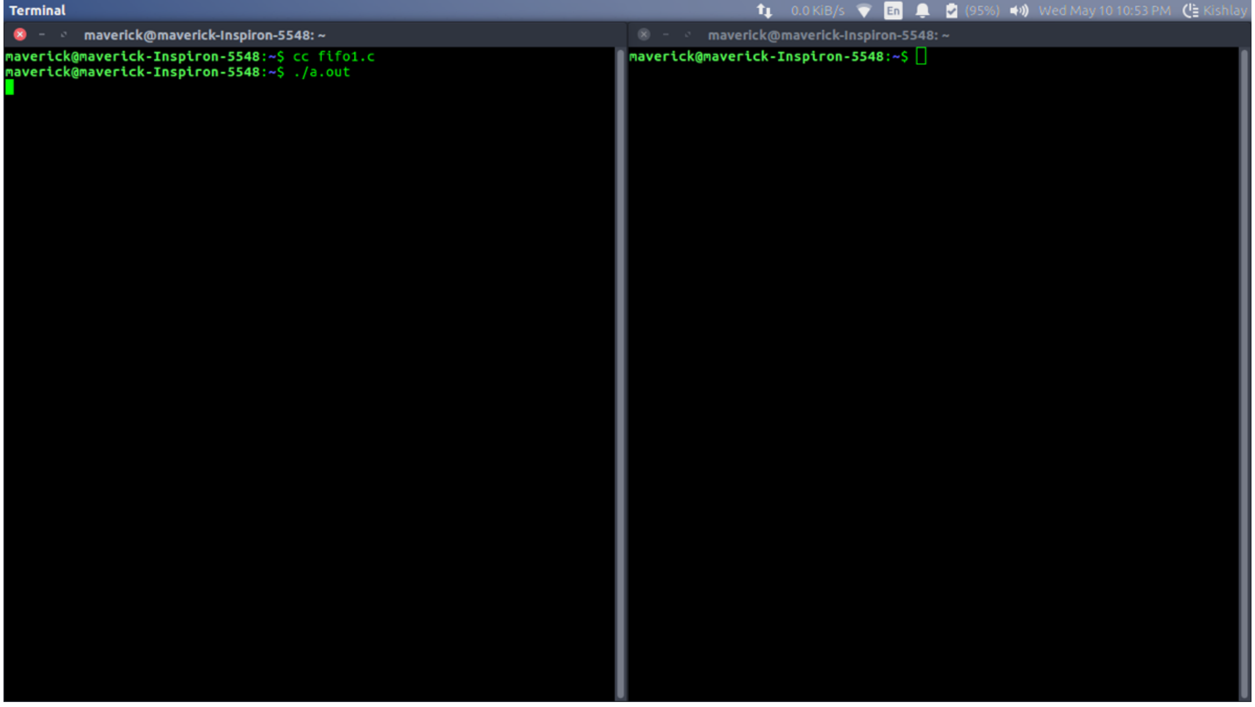
Example Programs to illustrate the named pipe: There are two programs that use the same FIFO. Program 1 writes first, then reads. The program 2 reads first, then writes. They both keep doing it until terminated.

```
// C program to implement one side of FIFO
// This side writes first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd;
    // FIFO file path
    char * myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)
    // mkfifo(<pathname>, <permission>)
    mkfifo(myfifo, 0666);

    char arr1[80], arr2[80];
    while (1)
    {
        // Open FIFO for write only
        fd = open(myfifo, O_WRONLY);

        // Take an input arr2ing from user.
        // 80 is maximum length
        fgets(arr2, 80, stdin);

        // Write the input arr2ing on FIFO
        // and close it
        write(fd, arr2, strlen(arr2)+1);
        close(fd);

        // Open FIFO for Read only
        fd = open(myfifo, O_RDONLY);

        // Read from FIFO
        read(fd, arr1, sizeof(arr1));

        // Print the read message
        printf("User2: %s\n", arr1);
        close(fd);
    }
    return 0;
}
```

Program 2(Reads First)

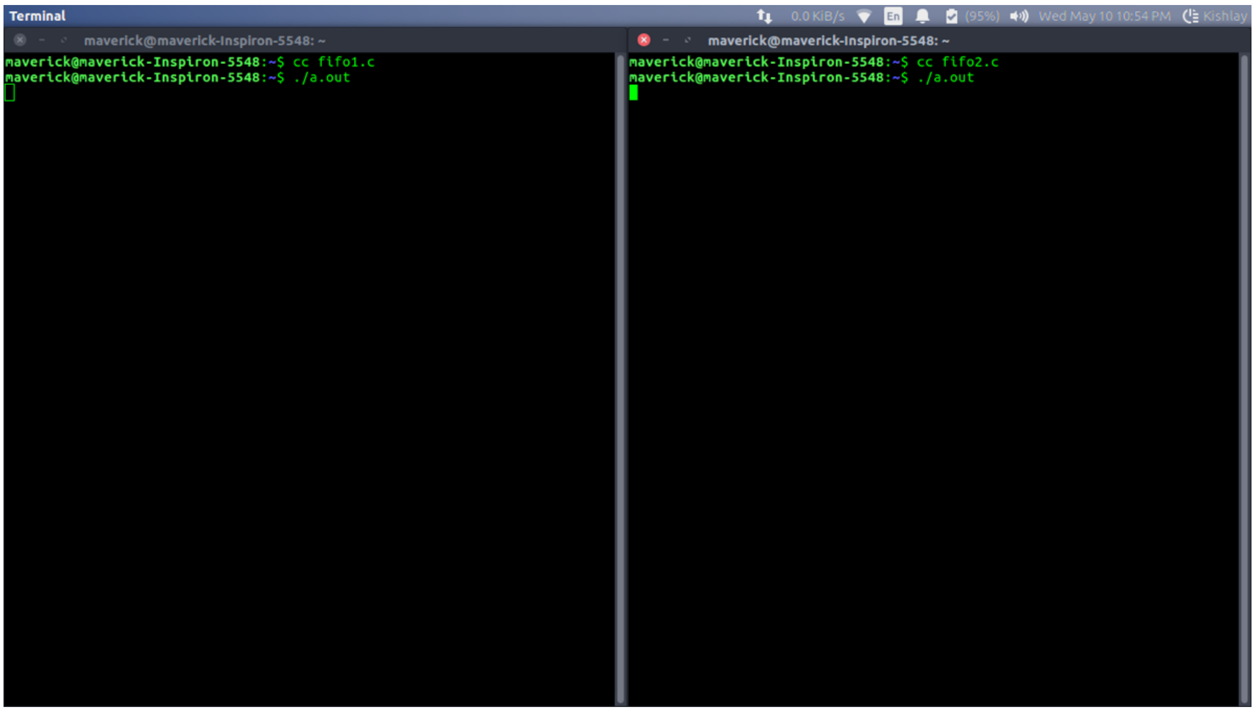
```c
// C program to implement one side of FIFO
// This side reads first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd1;

    // FIFO file path
    char * myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)
    // mkfifo(<pathname>,<permission>)
    mkfifo(myfifo, 0666);

    char str1[80], str2[80];
    while (1)
    {
        // First open in read only and read
        fd1 = open(myfifo,O_RDONLY);
        read(fd1, str1, 80);

        // Print the read string and close
        printf("User1: %s\n", str1);
        close(fd1);

        // Now open in write mode and write
        // string taken from user.
        fd1 = open(myfifo,O_WRONLY);
        fgets(str2, 80, stdin);
        write(fd1, str2, strlen(str2)+1);
        close(fd1);
    }
    return 0;
}
```

Output: Run the two programs simultaneously on two terminals.


Terminal
0.0 KiB/s
En
(95%)
Wed May 10 10:53 PM
Kishlay
maverick@maverick-Inspiron-5548: ~
maverick@maverick-Inspiron-5548: ~

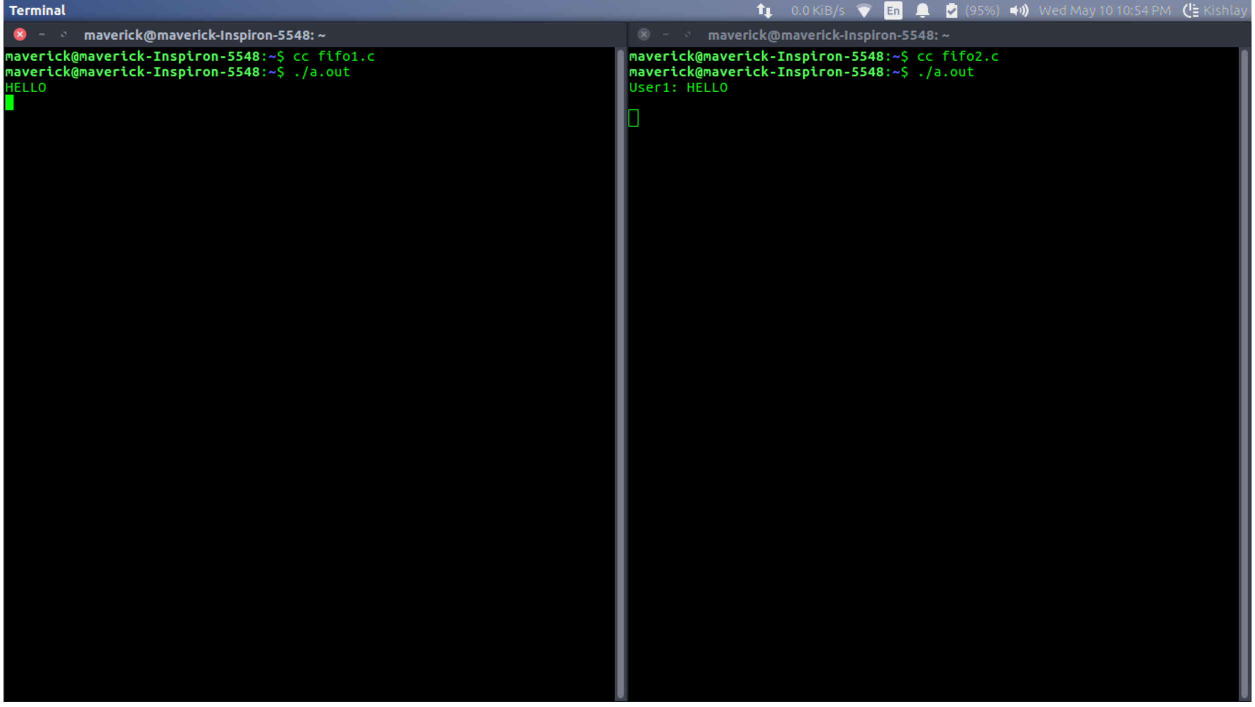
```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
```

```
maverick@maverick-Inspiron-5548:~$
```


Terminal
0.0 KiB/s
En
(95%)
Wed May 10 10:54 PM
Kishlay
maverick@maverick-Inspiron-5548: ~
maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
```

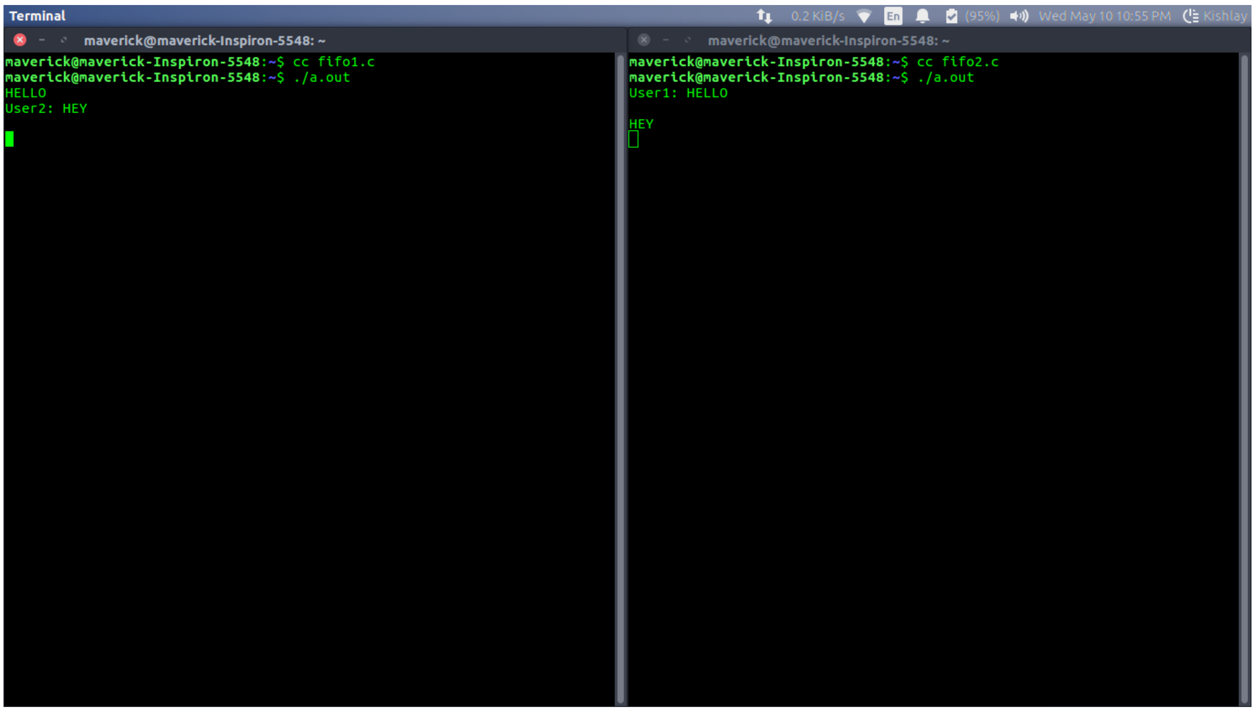
```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
```



Terminal
0.0 KiB/s
En
(95%)
Wed May 10 10:54 PM
Kishlay
maverick@maverick-Inspiron-5548: ~
maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
```

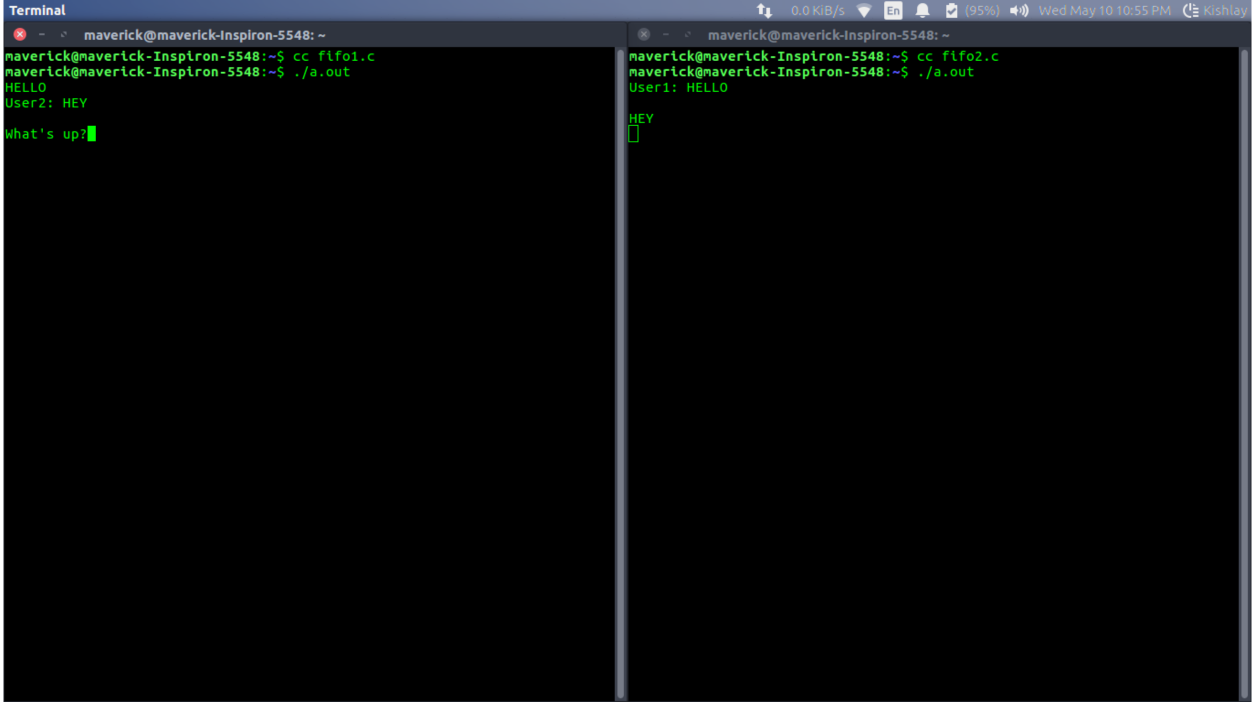
```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO
```



Terminal
0.2 KiB/s
En
(95%)
Wed May 10 10:55 PM
Kishlay
maverick@maverick-Inspiron-5548: ~
maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
```

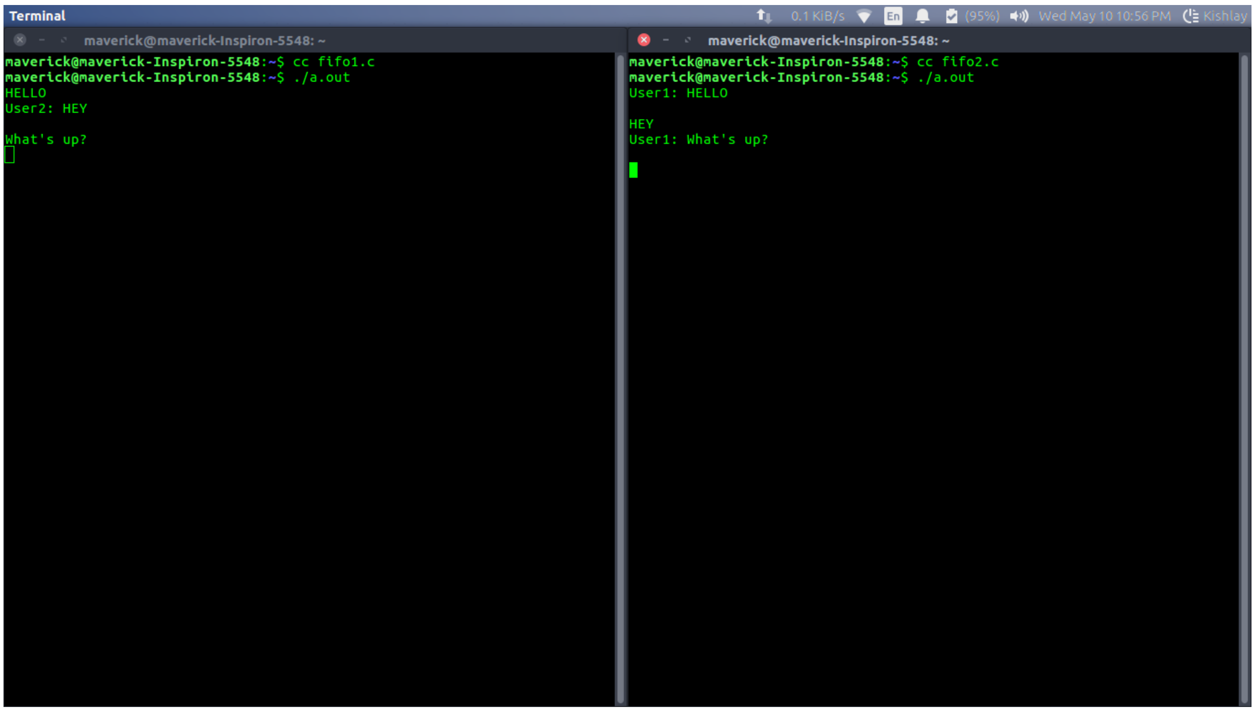


maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY

What's up?
```

maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
```

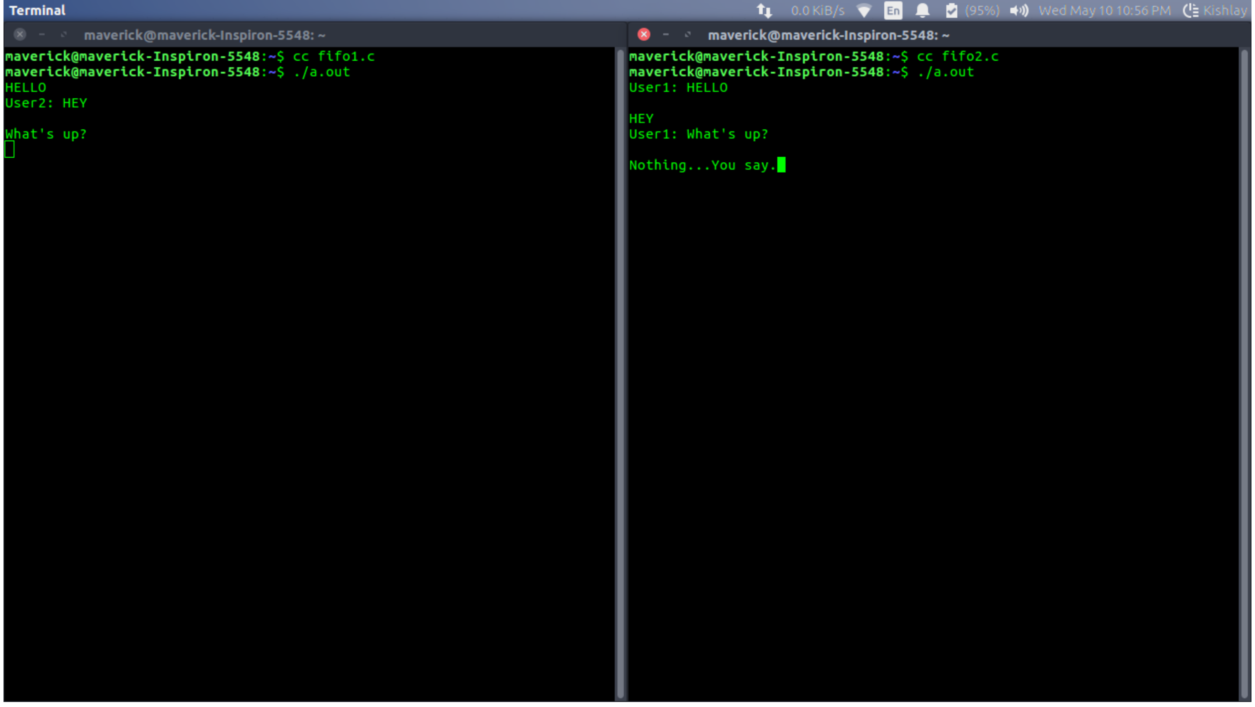


maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY

What's up?
```

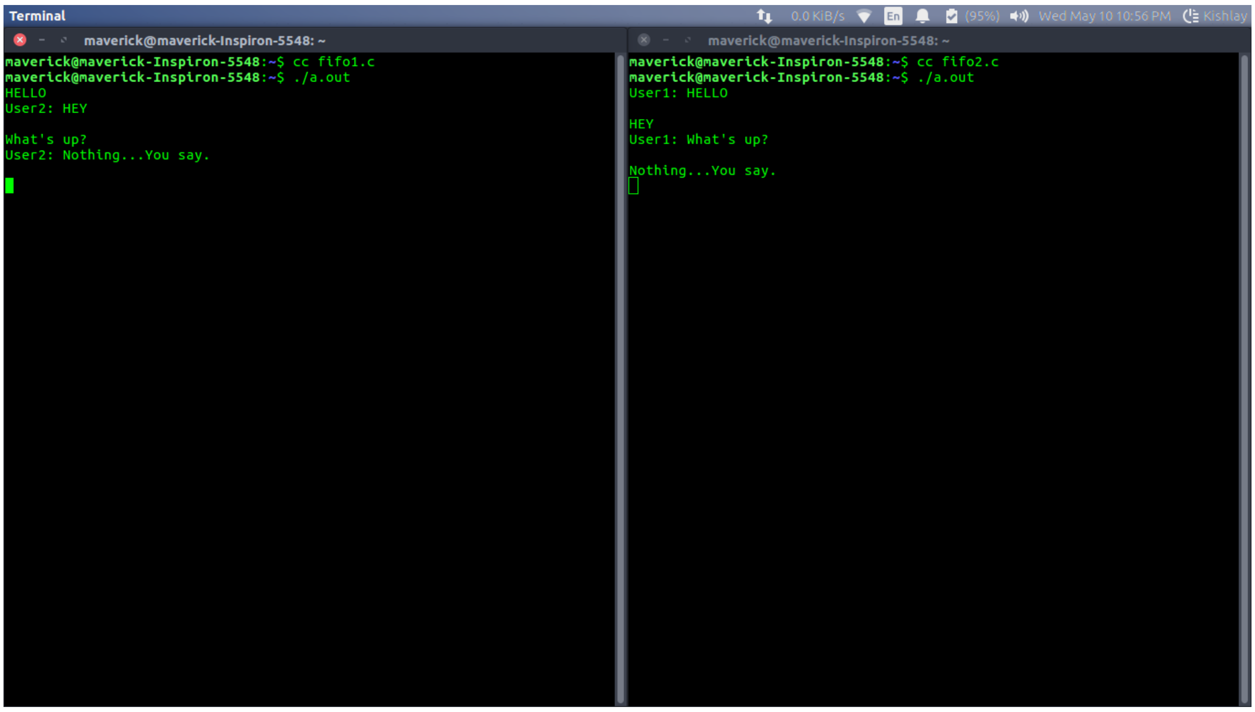
maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
User1: What's up?
```


Terminal 0.0 KiB/s En (95%) Wed May 10 10:56 PM Kishlay

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY

What's up?
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
User1: What's up?

Nothing...You say.
```


Terminal 0.0 KiB/s En (95%) Wed May 10 10:56 PM Kishlay

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY

What's up?
User2: Nothing...You say.
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
User1: What's up?

Nothing...You say.
```

This article is contributed by **Kishlay Verma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

C program for pipe in Linux

C program to demonstrate fork() and pipe()

pipe() System call

C program to print a string without any quote (singe or double) in the program

C program to detect tokens in a C program

How does a C program executes?

Output of C Program | Set 29

How to compile 32-bit program on 64-bit gcc in C and C++

Program to compute Log n

OpenMP | Hello World program

Splint -- A C program verifier

Program to check if two strings are same or not

Program for FCFS CPU Scheduling | Set 1

Program for Sum of the digits of a given number

Write a C program that won't compile in C++

Improved By : PaarmitaBhargava

Article Tags : C Linux-Unix Operating Systems

Practice Tags : Operating Systems C

4