

-18%

-18%

-18%

Program for IPC using named pipes (mkfifo())

[Leave a Comment](#) / [Programs](#) / By [Baljit Singh Saini](#)

Program for IPC using named pipes (mkfifo())

The third method for IPC is using mkfifo() function. mkfifo() creates a named pipe which can be used exactly like a file. So, if you know how to read/write in a file this is a convenient method for IPC

Syntax:

```
#include<sys/types.h>
#include<sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

mkfifo() makes a FIFO special file with the name specified by *pathname* and the permissions are specified by *mode*. On success mkfifo() returns 0 while on error it returns -1.

The advantage is that this FIFO special file can be used by any process for reading or writing just like a normal file. This means to sender process can use the write() system call to write data into the pipe and the receiver process can use the read() system call to read data from the pipe, hence, completing the communication.

It is same as a pipe except that it is accessed as part of the filesystem. Multiple process can access it for writing and reading. When the FIFO special files is used for exchange of data by process, the entire data is passed *internally* without writing it on the filesystem. Hence, if you open this special file there will be no content written in it.

Note: The FIFO pipe works in blocked mode(by default) i.e., the writing process must be present on one end while the reading process must be present on the other side at the same time else the communication will not happen. Operating the FIFO special file in non-blocking mode is also possible.

The entire IPC process will consist of three programs:

Program1: to create a named pipe

Program2: process that will write into the pipe (sender process)

Program3: process that will receive data from pipe (receiver process)

//Program1: Creating fifo/named pipe (1.c)

#include<stdio.h>



IMMERSE YOURSELF
WITH 16.1" DISPLAY QHD
DISPLAY

```
    printf("named pipe created\n");
}
```

//Now compile and run this program.

How it works?

This will simply create a named pipe (fifo1) with read, write and execute permission for all users. You can change this to whatever you prefer

Step 2 is to create a process which will use this pipe to send data. The below program will do that.

//**Program2:** Writing to a fifo/named pipe (2.c)

```
#include<unistd.h>
#include<stdio.h>
#include<fcntl.h>
int main()
{
    int res,n;
    res=open("fifo1",O_WRONLY);
    write(res,"Message",7);
    printf("Sender Process %d sent the data\n",getpid());
}
```

Compile this program as

```
$gcc -o 2 2.c
```

//Note: If you run this you will not see any output

How it works?

The above code opens the pipe created previously in writing mode (because it wants to send data). Then it uses "write" system call to write some data into it. Finally, it prints a message using printf. But when you compile and run it, it won't run because by default the sender runs in BLOCKING mode which means that until the receiver is not there the sender process gets blocked. Hence, you need a receiver process also.

The **third step** is to create the receiver process. The below program does so.

//**Program 3:** Reading from the named pipe (3.c)

```
#include<unistd.h>
#include<stdio.h>
#include<fcntl.h>
int main()
{
    int res,n;
    char buffer[100];
    res=open("fifo1",O_RDONLY);
    n=read(res,buffer,100);
    printf("Reader process %d started\n",getpid());
    printf("Data received by receiver %d is: %s\n",getpid(), buffer);
}
```

Compile the program as

```
$ gcc -o 3 3.c
```

How it works?

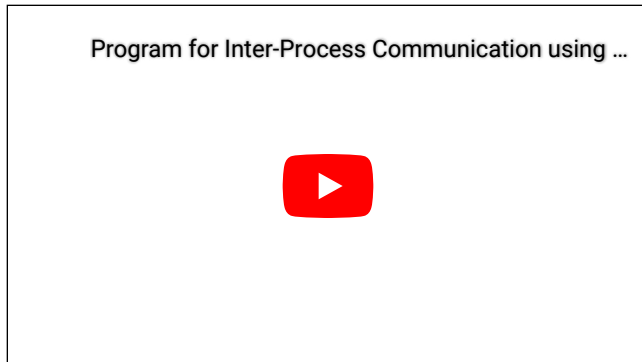
This program connects to the pipe in reading mode and reads the data into buffer and prints it. But again this program will not run. Because the receiver is BLOCKED until the sender is there.



```
baljit@baljit:~/cse325$ ./2 & ./3
[1] 33
Sender Process 33 sent the data
Reader process 34 started
Data received by receiver 34 is: Message
[1]+  Done                  ./2
baljit@baljit:~/cse325$
```

Named pipes

Video Link on IPC



Creating the named pipe in Unblocked Mode

Named pipes can also be used in Un-Blocked mode i.e., the sender (receiver) process will work independently of the other process. Refer the video above for more details of how to run named pipe in unblocked mode.

Viva Questions on Program for IPC using named pipes (mkfifo())

- Q1. Is named pipe bidirectional or unidirectional pipe?
- Q2. What is the default mode of a named pipe : blocking or non-blocking?
- Q3. What is meant by Blocking-Send?
- Q4. Can a process create two named pipes for communication with two different processes?

Relevant Programs on IPC using

- [popen\(\)](#)
- [pipe\(\)](#)
- [shared memory](#)



Leave a Comment


You must be [logged in](#) to post a comment.

Search ...

Q

OS Lab

Program to create Thread...




00:0009:29

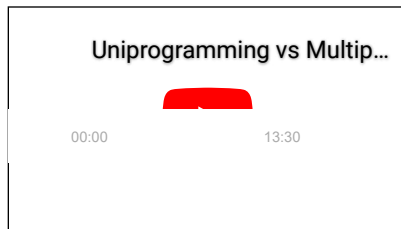


Linux Essentials Series

Using ls command in Linu...



00:0013:14



Categories

[Database](#)

[Entity-Relationship\(ER\)](#)

[Introduction to Database](#)

[Linux](#)

[commands](#)

[Shell Scripting](#)

[Operating System](#)

[CPU Scheduling](#)

[Deadlock](#)

[Disk Management](#)

[File Management](#)

[Introduction](#)

[IPC](#)

[Memory Management](#)

[Practice Problems](#)

[Process](#)

▼ [Process Synchronization](#)

[UGC-NET MCQ's for OS](#)

[Uncategorized](#)

Copyright © 2023 Dextutor | Powered by [Astra WordPress Theme](#)

