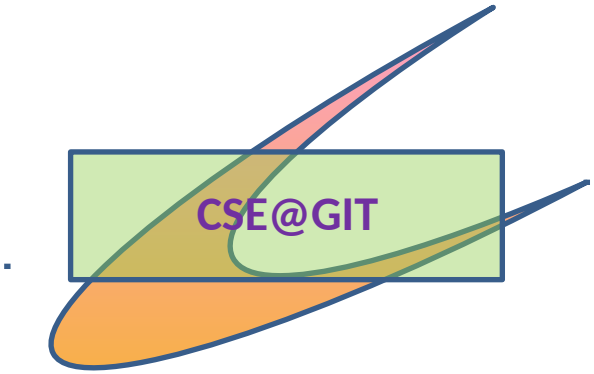
Experiment No. 12

Problem Definition:

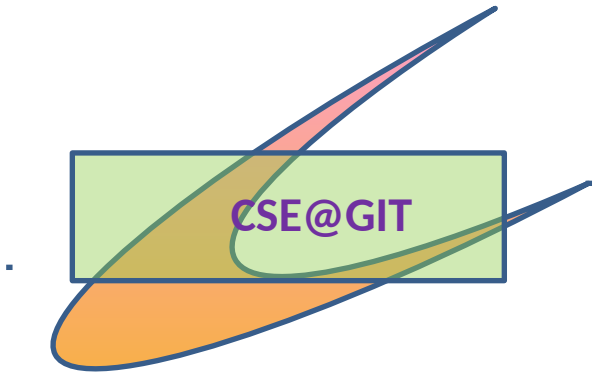
Suppose writer process generates data to be consumed by a reader process on the same machine. Develop a suitable inter process communication mechanism between the two processes that allows not just for one-time but also at subsequent times during execution.



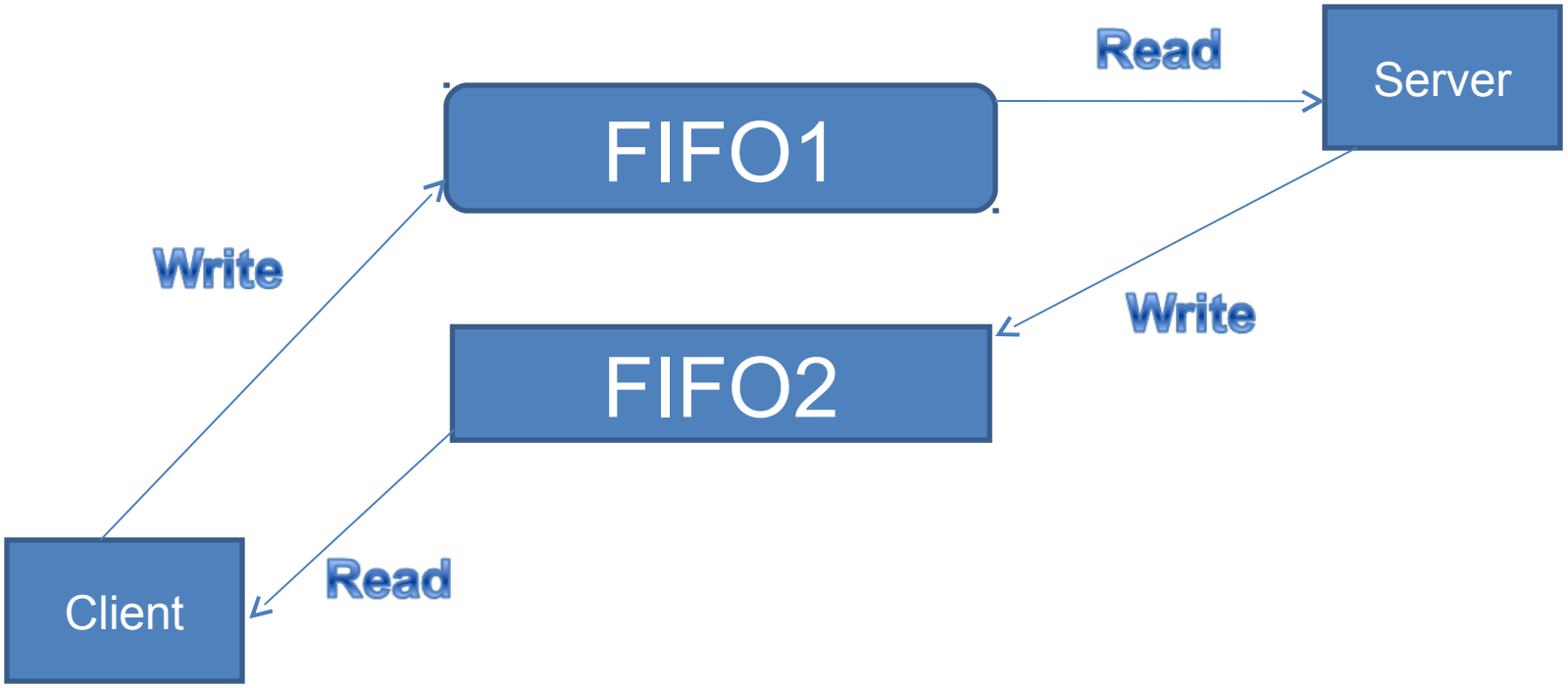
Objectives of the Experiment:

1) To familiarize with creation of FIFO file

2) To understand how the inter process communication takes place between two processes



Sturcture Of Program



Theoretical Background of the Experiment

- **FIFO - These are special device files used for inter process communication**
- **These are also known as named pipes**
- **Data written to a FIFO file are stored in a fixed-size buffer and retrieved in a first-in-first-out order**
- **To create:**

  int mkfifo(const char* path_name, mode_t mode);

  mkfifo("FIFO1", 0666);

How is synchronization provided?

- **When a process opens a FIFO file for read-only, the kernel will block the process until there is another process that opens the same file for write**
- **If a process opens a FIFO for write, it will be blocked until another process opens the FIFO for read**
- **This provides a method for process synchronization**

How is synchronization provided?

- **If a process writes to a FIFO that is full, the process will be blocked until another process has read data from the FIFO to make room for new data in the FIFO**

- **If a process attempts to read data from a FIFO that is empty, the process will be blocked until another process writes data to the FIFO**

How is synchronization provided?

- **If a process writes to a FIFO file that has no other process attached to it for read, the kernel will send a SIGPIPE signal to the process to notify it of the illegal operation**

- **If Two processes are to communicate via a FIFO file, it is important that the writer process closes its file descriptor when it is done, so that the reader process can see the end-of-file condition**

Usage

- **Uses of the fd argument are:**

Server Side

- **readfd is a file descriptor to read data from the FIFO1 file**
- **writefd is a file descriptor to write data to a FIFO2 file**

Client Side

- **writefd is a file descriptor to write data to a FIFO1 file**
- **readfd is a file descriptor to read data from the FIFO2 file**

Important system calls

open, read, write, close

- open: Open or create a FIFO
- read: Read from a FIFO
- write: Write data to a FIFO
- close: Close/destroy FIFO

Flow of implementation :

1. Declare required header files unistd.h, stdio.h

2. Create the two FIFO files for two processes called FIFO1 & FIFO2

3. In server side open the file descriptor FIFO1 for read and FIFO2 for write

4. In client side open the file descriptor FIFO1 for write and FIFO2 for read

5. Client side ,it enters the file name from user and write content into the FIFO1, server side opens the FIFO1 and reads the content of FIFO1

Flow of implementation :

6. Server opens the file mentioned in content and writes the contents into the FIFO2 file then closes the file descriptors

7. Client side , will read the content from FIFO2, and displays the read content onto the standard output then close the file descriptors

Pseudo Code / Outline of the Algorithm

Server.c

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

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666

char fname[256];

int main()
 {
   int readfd, writefd, fd;
   ssize_t n;
   char buff[512];
```

Pseudo Code / Outline of the Algorithm

Server.c

```
if (mkfifo(FIFO1, PERMS)<0)
  printf("Cant Create FIFO Files\n");

if (mkfifo(FIFO2, PERMS)<0)
  printf("Cant Create FIFO Files\n");

printf("Waiting for connection Request..\n");

readfd =open(FIFO1, O_RDONLY, 0);
writefd=open(FIFO2, O_WRONLY, 0);

printf("Connection Established..\n");

read(readfd, fname, 255);
printf("Client has requested file %s\n", fname);
```

Pseudo Code / Outline of the Algorithm

```
if ((fd=open(fname,O_RDWR))<0)
 {
   strcpy(buff,"File does not exist..\n");
   write(writefd, buff, strlen(buff));
 }
else
 {
   while((n=read(fd, buff,512))>0)
     write(writefd, buff, n);
 }

close(readfd);
unlink(FIFO1);

close(writefd);
unlink(FIFO2);

}
```

Pseudo Code / Outline of the Algorithm

Client.c

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

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666

char fname[256];

int main()
 {
   ssize_t n;
   char buff[512];
   int readfd,writefd;
```

Pseudo Code / Outline of the Algorithm

Client.c

```
    printf("Trying to Connect to Server..\n");
    writefd = open(FIFO1, O_WRONLY, 0);
    readfd = open(FIFO2, O_RDONLY, 0);

    printf("Connected..\n");
    printf("Enter filename to request from server: ");
    scanf("%s",fname);

    write(writefd, fname, strlen(fname));
    printf("Waiting for Server to reply..\n");

    while((n=read(readfd,buff,512))>0)
      write(1,buff,n);

    close(readfd);
    close(writefd);
    return 0;
}
```

OUTPUT

Output (Server)

[root@localhost USP Lab] ./a.out
Waiting for connection Request..
Connection Established..
Client has requested file 1.c
[root@localhost USPLab]

Output (Client)
[root@localhost USPLab] ./a.out
Trying to Connect to Server..
Connected..
Enter the filename to request from server:1.c
Waiting for Server to reply..
Hi Welcome to USP Lab.
Now End of file

Learning Outcomes of the Experiment

At the end of the session, students should be able to :

1) Understand creating the FIFO file[L2].
2) Understand how the interaction takes place between two processes using FIFO[L2].