Unix Assignment - 3.

Vivek V. Pai
1DB18IS092

① Discuss the application of FIFO?

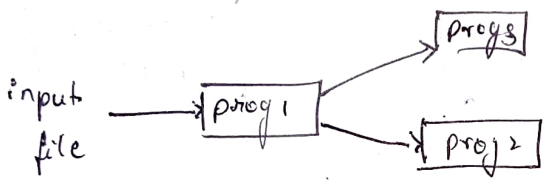
→ There are 2 uses for fifo:

* FIFO are used by shell commands to pass data from on shell pipeline to another without creating intermediate temporary files.

* FIFO are used as rendezous points in client-server applications to pass data between the clients & the servers.

Example using FIFO to duplicate Output streams:

FIFOs can be used to duplicate an output stream in a servies of shell commands. This prevents writing the data to an intermediate disk file. Consider a procedure that needs to process a filtered input stream twice



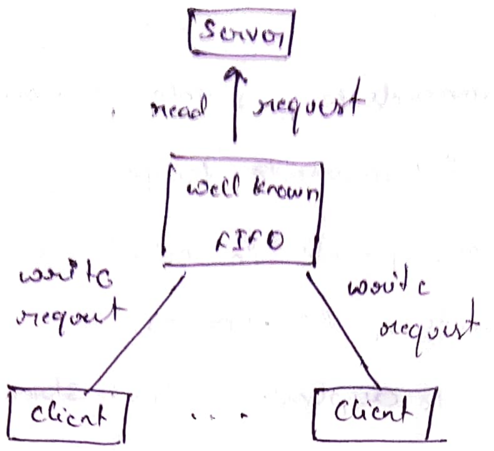
② Explain client server communication using FIFO with a neat diagram.

→ * FIFOs can be used used to send data between a client & server. If we have a server that is contacted by numerous clients, each client can write its request to a well-known FIFO. that the server creates. Since there are multiple writers for the FIFO the requests sent by the clients to the server need to be less than PIPE BUF bytes in size

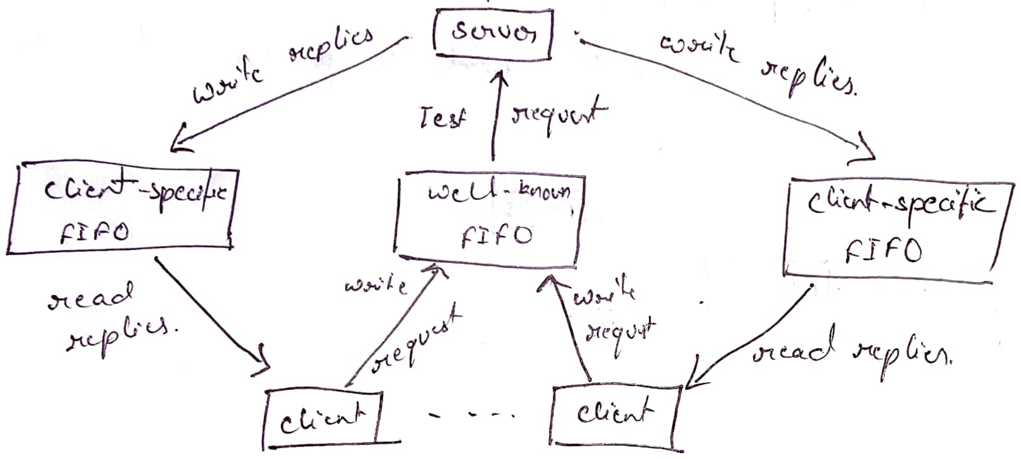
* This prevents any interleaving of the client writes. The problem in using FIFO's for this type of client, server communication is how to send replies back from the server to each client.

* A single FIFO can't be used, as the clients would never know when to read their response versus responses for other clients. One Solution is for each client to send its process ID with the request. The server then creates a unique FIFO for each client,

using a path name based on the client sprocess ID



clients sending request to a server using a FIFO

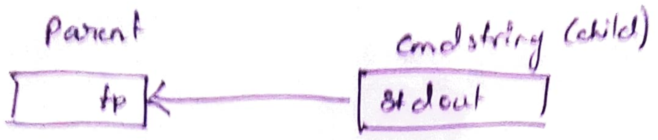


③ Explain popen & pclose functions?

Since a common operation is to create to pipe to another process, to either read its output or send it input, the standard I/O library has historically provided the popen & pclose functions These two function handle all the dirty work that we've been doing ourselves; creating a pipe forking a child; closing the unused ends of the pipe, executing a shell to run the command, and waiting for the command to terminate

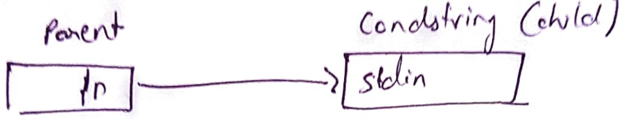
```
# include <stdio.h>
FILE * poper ( const char * cmdsting , const char * type;
Return : termination status of cmdstring (or) 1 on error int
pclose (FILE * fp);
Return; file printer if ok, NULL on error.
```

The function popendoes a forkand execto execute the cmdstring and returns : ~~file pointer~~ a standard I/O file printer. If type is "r" the file pointer is connected to the standard output of cmdstring



Result of fp = open (Cmdstring, "r")

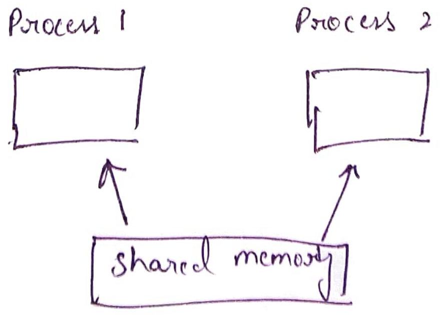
If type "w", the file pointer is connected to the standard. input of cmdstring, as shown:



Result of fp = popen (Cmdstring, "w")

(a) write short note on shared memory?

Shared memory is a feature supported by unix system V, including Linux. One process must explicitly ask for an area, using a key, to be shared by other process. This process will be called the server. All other process, the client that know the shared area can access it. However, there is no protection to a shared memory and any process that knows it can access it freely. To protect a shared memory from being accessed at the same time by several processes, a synchronization protocol must be setup.



Process 1
Process 2
Shared memory