33231

CLASSMATE
Date :
Page :

ASSIGNMENT 7

Title : Inter Process Communication in Linux

Problem Statement : Inter Process communication in LINUX uses foll :-

a) **Pipes** – Full duplex communication between parent and child processes. Parent process writes a pathname of file (the contents of the file are desired) on one pipe to be read by child process and child process writes the content of the file on second pipe to be read by parent process and displays on standard output.

b) **FIFOs** – Full duplex communication between two independent processes. First process accepts sentences and writes on one pipe to be read by second process and second process counts number of characters, number of words and number of lines in accepted sentences writes this output in a text file and writes the contents of the file on second pipe to be read by first process and displays on standard output.

Theory :

Pipe : – a connection between two processes, such that the standard output from one process becomes the standard input of the other process.

In UNIX, Operating System pipes are useful for communication between related processes (IPC).

Properties of Pipes :–

- Pipe is one-way communication only i.e. we can use a pipe such that one process writes to the pipe and the other process reads from the pipe. It opens a pipe, which is an area of main memory that is treated as a "virtual file".

- The pipe can be used by the creating process, as well as all its child processes for reading and writing. One process can write to this "virtual file" or pipe and another related process can read from it.
- If a process tries to read before something is written to the pipe, the process is suspended until something is written.
- The pipe system call finds the first two available positions in the process's open file table and allocates them for the read and write end of the pipe.

Syntax in C language :

```
int pipe (int fds[2]);
```

Parameters :

fd[0] will be file descriptor for read end of the pipe
fd[1] will be file descriptor for write end of the pipe

Returns : 0 on success, -1 on error

FIFOS :- a named pipe (FIFO) is one of the methods for inter-process communication.

Properties of FIFO -

- 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 ~~to~~ for inter-process communication. A FIFO 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.

Algorithm for IPC using pipes :

1) Create two pipes // pipe
2) Create child process // fork
3) Parent process should create a pathname of a file to path // write pipe1 - fd[1]
4) Child should retrieve / read the pathname from the pipe // read pipe1 fd[0]
5) Child should write the contents of the file to the pipe // write pipe2 fd[1]
6) Parent should retrieve / read the contents of the file from the pipe // read pipe2 fd[0]
7) write it to the standard output

Algorithm for IPC using FIFOs :

1) Create 1st named pipe
2) Accept sentences
3) Write into 1st named pipe
4) Open second pipe in read mode

5) Read chars, lines, words and print

Part 2

1) Open 1st named pipe in read mode (created in part 1)
2) Count no. of characters, lines and words in accepted sentence
3) Create a 2nd named pipe
4) Store chars, lines, words in buffer
5) Write buffer onto 2nd named pipe

Conclusion: I have successfully understood and implemented inter-process communication in LINUX using Pipes and FIFOs.