Lab-11

Aim: Inter Process Communication. (Use of pipe system call and mkfifo)

Explanation:

Pipe system call:

#include <unistd.h>
int pipe(int pipefd[2]);

pipe() creates a pipe, a unidirectional data channel that can be used for inter process communication.

The array *pipefd* is used to return two file descriptors referring to the ends of the pipe.

- *pipefd[0]* refers to the read end of the pipe.
- *pipefd[1]* refers to the write end of the pipe.

Data written to the write end of the pipe is buffered by the kernel until it is read from the read end of the pipe.

On success, zero is returned. On error, -1 is returned, and *errno* is set appropriately.

Close system call:

#include <unistd.h>
int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused.

Any record locks held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock).

close() returns zero on success. On error, -1 is returned, and errno is set appropriately.

Programs:

1. Write a program to create a pipe and print the values of pipe file descriptors.

Solution logic:

- The program should create a pipe and if file creation is successful, the end names should be printed.

2. Write a program to pass a message from parent process to child process through a pipe.

Solution logic:

- The program should create a pipe and if file creation is successful, a child process should be created.
- Parent should read a string from stdin and pass the same to child process.
- Child process should print the string received from parent process.

3. Write a program to pass file name from parent process to child process through a pipe, child process should pass the file contents to parent process and parent should print the contents.

Solution logic:

- The program should create a pipe and if file creation is successful, a child process should be created.
- Parent should read a file name from stdin and pass the same to child process.
- Child process should open the file received from parent process and write the content to another pipe.
- Parent should read the contents and display on output.

Assignment:

Demonstrate the use of named pipe with appropriate programs and commands both.