Date
Page No.

Practical No.5

* **Aim:-** Implementation of inter process communication model.

* **Theory:-**

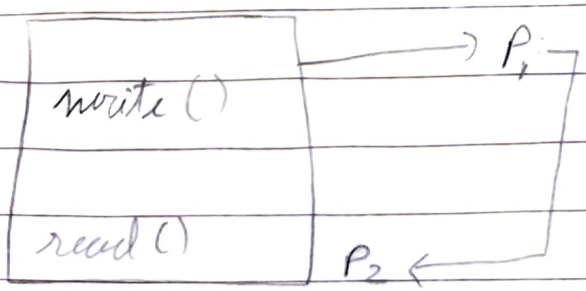
Inter process communication in OS is using by which multiple process can communicate with other. Shared memory, message passing are some ways to achieve IPC in OS.

pipe() function:-

→ It is a simple form of shared memory that allows two process to comm. with each other.

→ It has half duplex method is used for IPC between 2 related process.

→ It's like a scenario talking water with a tap into pipe & the reading process is retrieving from the pipe.

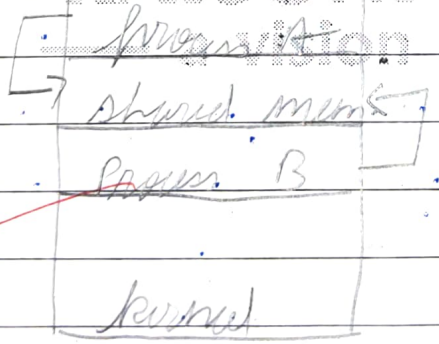


Date
Page No.

* Ways to achieve :-

+ Shared memory
+ Message passing

1) Shared memory :- It is a region of memory that is accessible to multiple process. It allows process to communicate with each other by reading & writing data from the shared memory region.
→ It is a fast and efficient way for process of communication but it can be efficient if process are not carefully synchronised.



2) Message passing :- It is a method of ipc, in OS involves the exchange of messages between process where each process send & receive message to co-ordinate its activities & exchange data with other process.

CODE:

```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>

using namespace std;

int main() {
  int pipefd[2];
  pid_t pid;
  char buffer[50];

  // Create pipe
  if (pipe(pipefd) == -1) {
    cerr << "Pipe creation failed." << endl;
    return 1;
  }

  // Fork a child process
  pid = fork();

  if (pid < 0) {
    cerr << "Fork failed." << endl;
    return 1;
  }

  if (pid > 0) { // Parent process
    close(pipefd[0]); // Close reading end of pipe in parent process

    string userInput;
    cout << "Enter a message to send to the child process: ";
    getline(cin, userInput);

    // Write user input to pipe
    write(pipefd[1], userInput.c_str(), userInput.length() + 1);

    // Wait for child process to finish
    wait(NULL);
  } else { // Child process
    close(pipefd[1]); // Close writing end of pipe in child process

    // Read data from pipe
    read(pipefd[0], buffer, sizeof(buffer));
    cout << "Child received message: " << buffer << endl;
  }

  return 0;
}
```

OUTPUT:

Output

Clear

```
/tmp/hFAYfbMDCx.o
Enter a message to send to the child process: Hi this is the messege from
    child process
Child received message: Hi this is the messege from child process
```