Practical-2

**Aim:-** Using the "pipe()" system call, implement the following :-

1. Perform inter-process communication between a Parent and Child process.
2. Perform inter-process communication between Two Child processes.

# Theory:-

A pipe is 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(inter- process communication).

Pipe is one-way communication only i.e we can use a pipe such that One process write 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”.

# Code:-

* 1. Between a Parent and Child process #include <stdio.h>

#include <stdlib.h> #include <unistd.h> #include <string.h>

int main() {

int pipefd[2];

char message[100]; char buffer[100];

if (pipe(pipefd) == -1) { perror("pipe");

}

pid\_t pid = fork();

if (pid == -1) {

perror("fork"); exit(EXIT\_FAILURE);

}

if (pid == 0) { close(pipefd[1]);

read(pipefd[0], buffer, sizeof(buffer)); printf("Child received: %s\n", buffer);

close(pipefd[0]);

} else {

close(pipefd[0]);

printf("Enter a message: "); fgets(message, sizeof(message), stdin);

write(pipefd[1], message, strlen(message) + 1); close(pipefd[1]);

wait(NULL);

}

return 0;

}

# Output:-

* 1. Between Two Child Processes

#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <string.h>

int main() {

int pipefd[2];

char message[100]; char buffer[100];

if (pipe(pipefd) == -1) { perror("pipe");

}

pid\_t pid1 = fork();

if (pid1 == 0) { close(pipefd[0]);

printf("Process id for first child: %d\n",getpid()); printf("Enter a message: ");

fgets(message, sizeof(message), stdin); write(pipefd[1], message, strlen(message) + 1);

close(pipefd[1]);

} else { close(pipefd[1]); pid\_t pid2 = fork();

if (pid2 == 0) { close(pipefd[1]);

read(pipefd[0], buffer, sizeof(buffer));

printf("Process id for second child: %d\n",getpid()); printf("Second child received: %s\n", buffer);

close(pipefd[0]);

} else {

wait(NULL); wait(NULL);

}

}

return 0;

}

# Output:-

