**Experiment 11**

**AIM:** Write a program to simulate the concept of inter process communication

**THEORY:**

The system call provides an interface to the operating system services. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do.

**Types of System Calls**

There are 5 different categories of system calls:

process control, file manipulation, device manipulation, information maintenance and communication.

● **Process Control:** A running program needs to be able to stop execution either normally or abnormally. When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger.

● **File Management :** Some common system calls are create, delete, read, write, reposition, or close. Also, there is a need to determine the file attributes – get and set file attribute. Many times the OS provides an API to make these system calls.

● **Device Management** : Process usually require several resources to execute, if these resources are available, they will be granted and control returned to the user process. These resources are also thought of as devices. Some are physical, such as a video card, and others are abstract, such as a file. User programs request the device, and when finished they release the device. Similar to files, we can read, write, and reposition the device.

● **Information Management :** Some system calls exist purely for transferring information between the user program and the operating system. An example of this is time, or date. The OS also keeps information about all its processes and provides system calls to report this information.

● **Communication :**There are two models of interprocess communication, the message-passing model and the shared memory model. Message-passing uses a common mailbox to pass messages between processes. Shared memory use certain system calls to create and gain access to create and gain access to regions of memory owned by other processes. The two processes exchange information by reading and writing in the shared data.

**C++ Code:**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

int pfd[2], i;

pid\_t mypid;

if (pipe(pfd) < 0)

perror("Pipe Error");

if (!fork()) {

char data;

printf("Enter a Number…\n");

scanf("%d", & data);

write(pfd[1], & data, 1);

mypid = getpid();

printf("I am process %d\n", mypid);

printf("My parent is process %d\n", getppid());

printf("Child Exiting…\n");

exit(0);

} else {

char data1;

read(pfd[0], & data1, 1);

printf("Received %d from child \n", data1);

printf("The odd numbers are… \n");

for (i = 1; i <= data1; i += 2) {

printf("%5d", i);

sleep(2);

}

printf("\n Parent Exiting…\n");

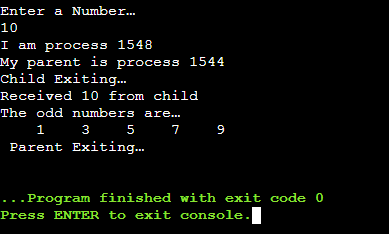
exit(0);

}

return (0);

}

**Output:**

****