**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:** Pipe is a communication medium between two or more related or interrelated processes. It can be either within one process or a communication between the child and the parent processes. Communication can also be multi-level such as communication between the parent, the child and the grand-child, etc. Communication is achieved by one process writing into the pipe and other reading from the pipe. To achieve the pipe system call, create two files, one to write into the file and another to read from the file.

Pipe mechanism can be viewed with a real-time scenario such as filling water with the pipe into some container, say a bucket, and someone retrieving it, say with a mug. The filling process is nothing but writing into the pipe and the reading process is nothing but retrieving from the pipe. This implies that one output (water) is input for the other (bucket).

This system call would create a pipe for one-way communication i.e., it creates two descriptors, first one is connected to read from the pipe and other one is connected to write into the pipe.

**CODE 1:**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

int fd[2],n;

pid\_t p;

char b[20];

pipe(fd); //creates a unidirectional pipe with two end fd[0] and fd[1]

p=fork();

if(p>0)

{

printf("PARENT:Hello Child, I am sending a message to you\n");

write(fd[1],"Hii,child\n",10); //fd[1] is the write end of the pipe

}

else

{

printf("CHILD:Hello, Child this side\n");

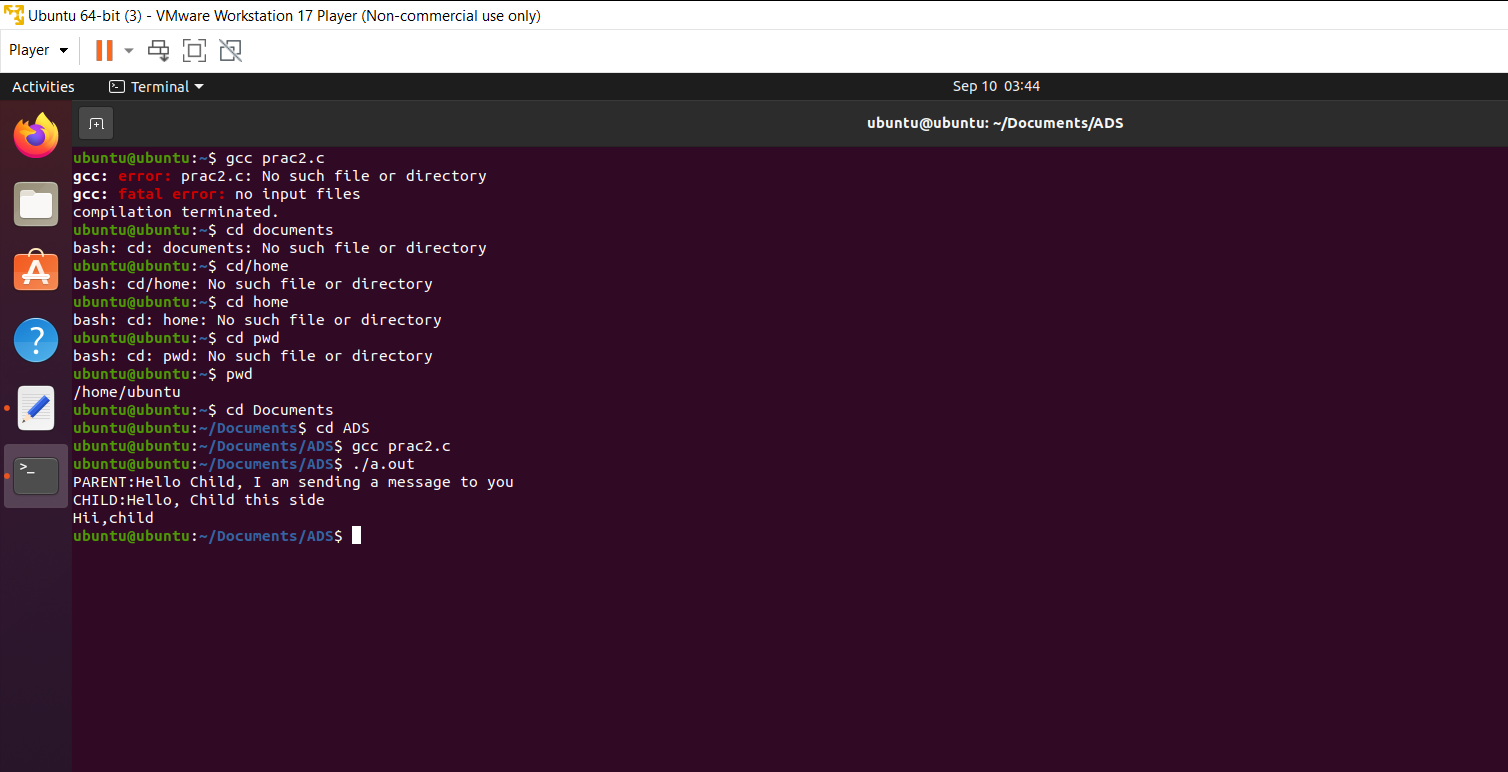
n=read(fd[0],b,100); //fd[0] is the read end of the pipe

write(1,b,n);

}

}

**OUTPUT:**

****

**CODE 2:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

int main() {

int pipefds[2],n;

pid\_t child1, child2;

char buffer[256];

if (pipe(pipefds)==-1){

printf("Pipe cannot be created");}

switch (child1 = fork()) {

case -1:

// error - abort

break;

case 0: /\* child 1 \*/

close(pipefds[0]);

write(pipefds[1], "Hello , Child1 this side", 30);

}

switch (child2 = fork()) {

case -1:

// error - abort

break;

case 0: /\* child 2 \*/

//char buffer[256];

close(pipefds[1]);

n = read(pipefds[0], buffer, sizeof(buffer) - 1);

if (n < 0) {

// handle error

} else {

buffer[n] = '\0';

printf("Child 2 Recieved Message from Child 1: '%s'\n", buffer);

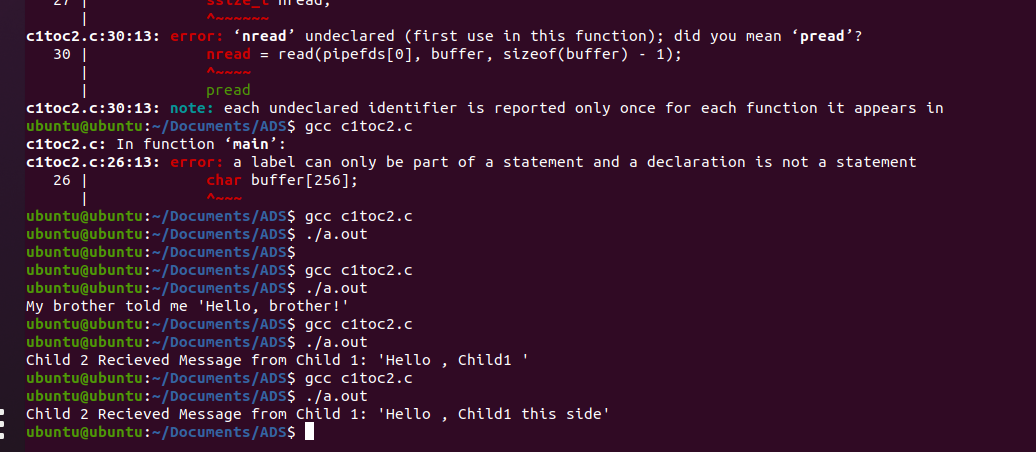
}

close(pipefds[0]);

}

}

**OUTPUT**

****