#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

#include<iostream>

using namespace std;

int main()

{

int fd[2];

pipe(fd);

int pid=fork();

char message[] ="Hi WORLD! \n";

if(pid==0)

{

cout<<endl<<"In child 1"<<endl;

close(fd[1]);

char message1 [20];

read(fd[0],message1,20);

cout<<message1<<endl;

pid=fork();

if(pid==0)

{

cout<<endl<<"In child 2"<<endl;

close(fd[1]);

char message2 [20];

read(fd[0],message2,20);

cout<<message2<<endl;

}

}

else

{

cout<<endl<<"In the Parent"<<endl;

close(fd[0]);

write(fd[1],message,20);

}

return 0;

}

**Conclusion:**

It’s stuck in an infinity loop whenever there are two children executing at same time though one pipe. The reason is that whether you want this process to be a child of the original parent or a child of the first child process (it is usual for it to be a child of the original parent).The second child executed but unable to write. Reading junk data. The one pipe cannot communicate with two children at the same time. The blocking of one child is obvious and it wanted parent to be write, however parent is unwilling to write.

Hence we used concept of using two pipes to resolve issue of reading and writing of processes. And process them individually.