Ex.No:10	IMPLEMENT IPC USING PIPE

AIM:

To write a C Program to implement IPC using Pipe

ALGORITHM:

Step 1: Start the program
Step 2 : Create a pipe and child process.
Step 3: Parent process writes to the pipe for both one way and two way communication.
Step 4: Child process retrieves the message from the pipe and writes it to the standard output
Step 5: Stop the program

ONE WAY COMMUNICATION:
PROGRAM:

```
#include<stdio.h>
#include<unistd.h>

int main()
{
        int pipefds[2];
        int returnstatus;
        int pid;
        char writemessages[2][20]={"Hi", "Hello"};
        char readmessage[20];
        returnstatus = pipe(pipefds);
  if (returnstatus == -1) {
    printf("Unable to create pipe\n");
    return 1;
  }
  pid = fork();

  // Child process is created
  if (pid == 0)

        {
                read(pipefds[0], readmessage, sizeof(readmessage));
               printf("Child Process - Reading from pipe – Message 1 is %s\n", readmessage);
               read(pipefds[0], readmessage, sizeof(readmessage));
                printf("Child Process - Reading from pipe – Message 2 is %s\n", readmessage);
         }
        else
        {
                //Parent process
         printf("Parent Process - Writing to pipe - Message 1 is %s\n", writemessages[0]);
        write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
        printf("Parent Process - Writing to pipe - Message 2 is %s\n", writemessages[1]);
        write(pipefds[1], writemessages[1], sizeof(writemessages[1]));
```

```
        }
          return 0;
        }
```

OUTPUT:
Parent Process - Writing to pipe - Message 1 is Hi
Parent Process - Writing to pipe - Message 2 is Hello

TWO WAY COMMUNICATION:

```
#include<stdio.h>
#include<unistd.h>

int main() {
  int pipefds1[2], pipefds2[2];
  int returnstatus1, returnstatus2;
  int pid;
  char pipe1writemessage[20] = "Hi";
  char pipe2writemessage[20] = "Hello";
  char readmessage[20];

        returnstatus1 = pipe(pipefds1);
       if (returnstatus1 == -1)
        {
         printf("Unable to create pipe 1 \n");
         return 1;
        }

          returnstatus2 = pipe(pipefds2);
          if(returnstatus2 == -1)
       {
         printf("Unable to create pipe 2 \n");
       return 1;
        }
    pid = fork();
   if(pid != 0)
   {
   close(pipefds1[0]);
   close(pipefds2[1]);
   printf("In Parent: Writing to pipe 1 – Message is %s\n", pipe1writemessage);
   write(pipefds1[1], pipe1writemessage, sizeof(pipe1writemessage));
   read(pipefds2[0], readmessage, sizeof(readmessage));
   printf("In Parent: Reading from pipe 2 – Message is %s\n", readmessage);
  }
 else
  {
   close(pipefds1[1]);
   close(pipefds2[0]);
   read(pipefds1[0], readmessage, sizeof(readmessage));
   printf("In Child: Reading from pipe 1 – Message is %s\n", readmessage);
   printf("In Child: Writing to pipe 2 – Message is %s\n", pipe2writemessage);
   write(pipefds2[1], pipe2writemessage, sizeof(pipe2writemessage));
  }
```

```
  return 0;
}
```

OUTPUT:

```
In Parent: Writing to pipe 1 – Message is Hi
In Child: Reading from pipe 1 – Message is Hi
In Child: Writing to pipe 2 – Message is Hello
In Parent: Reading from pipe 2 – Message is Hello
```

Result: Thus the c program to implement pipes for both one way and two way communication is executed .