Programming Laboratory I
Group-A
Assignment No: 5

1 Title

Write an **IPC** program using pipe. Process **A accepts** a character string and Process **B inverses** the string. Pipe is used to establish **communication** between A and B processes using Python or C++.

2 Aim

To write an **IPC** program using **pipe** where one process accepts and the other inverses the string. Pipe is used to establish **communication** between the two processes.

3 Theory

IPC:

1. In computing, **inter-process communication (IPC)** is a set of methods for the exchange of data among **multiple threads** in one or more processes.
2. Processes may be running on one or more computers **connected** by a network.
3. IPC methods are divided into methods for **message passing**, **synchronization**, **shared memory**, and **remote procedure calls (RPC)**.
4. The method of IPC used may vary based on the **bandwidth** and **latency of communication** between the threads, and the type of data being communicated.
5. There are several reasons for providing an environment that allows process cooperation:
   (a) Information sharing
   (b) Computational speedup
   (c) Modularity

(d) Convenience

(e) Privilege separation

6. IPC may also be referred to as **inter-thread** communication and **inter-application** communication.

7. The combination of **IPC** with the **address space** concept is the foundation for address space independence/isolation.

8. Interprocess communication (IPC) is the **transfer** of data among processes.

9. **For example**, a Web browser may request a Web page from a Web server, which then sends HTML data.
   This transfer of data usually uses sockets in a telephone-like connection.
   In another example, you may want to print the filenames in a directory using a command such as *ls — lpr*.
   The shell creates an ls process and a separate *lpr* process, connecting the two with a *pipe*, represented by the "—" symbol.
   A pipe permits one-way communication between two related processes. The *ls* process writes data into the pipe, and the *lpr* process reads data from the pipe.

4 Mathemathical Modelling

Let S be the solution perspective of the class such that
S={s, e, i, o, f, DD, NDD, success, failure}
s={Initial state that is constructor of the class}
e={End state or destructor of the class}
i={Input of the system}
o={Output of the system}
DD={Deterministic data:it helps identifying the load store functions or assignment functions}
NDD={Non deterministic data:data of the system S to be solved}
Success={Desired outcome generated}
Failure={Desired outcome not generated or forced exit due to system error}
For class:
f={pipe()}
pipe()={establishes the connection between the two processes}

5 Algorithm

1. Start.

2. Create child process using fork system call and store its PID.

3. If current process is child process, then

   (a) Close unwanted pipe.

   (b) Accept string from the user.

(c) Write the string to pipe.

4. If current process is the parent process, then
   (a) Wait for child process to complete.
   (b) Close unwanted pipe.
   (c) Reverse the string.
   (d) Print the string.
5. Stop.

6 State Transition Diagram

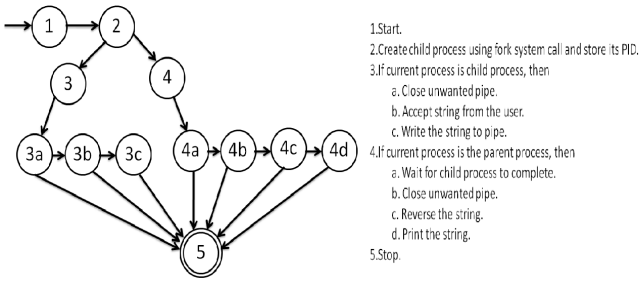

Figure 1: State Transition Diagram

7 Conclusion

Thus, we have written an **IPC** program using **pipe** where one process accepts and the other inverses the string. Pipe is used to establish **communication** between the two processes.