# COEN 383: Project 6

Group 3
Sruthi Thiyagarajan
Mitali Sahoo
Bharat Chadalawada

# Report

In this Project, we simulated communication between parent and child processes using a pipe. A pipe is a virtual file created in memory having both writes as well as a read file descriptor, but the file is never written to the file system. The design of our simulation is simple. We have created 5 pipes in the main process and then forked 5 separate child processes. Then we have assigned one pipe to each process of the child. The child process is using this pipe to send messages to the main process as per project requirement & the 'main' process reads from all these child processes and writes the output to "Output.txt" as soon as it reads the pipe.

We have faced the following issues:
1. Parent process was unable to detect when the child process closes dedicated pipe's write file Descriptor.
**Challenge:** To describe this issue, consider that each child process must close its pipe after simulation by using a "close" system call. The parent process can detect it using a combination of "select" and "read" system call. Specifically, when the child process closes its pipe, then "select" system call tells the parent that there is data at the end of the pipe, however, when parent read that data then number of bytes read is "0", indicating the pipe has been closed by the child process.

**Issue:** EOF byte is appended on pipe only when the pipe has no open write file descriptor. When we fork a child after creating a pipe, then both the main process as well as the child process have pipes write file descriptor and therefore violating the above condition. And therefore, the closing pipe by a child's end doesn't indicate any EOF character in the pipe to the parent.

**Resolution:** Parent and child both must close the unused file descriptor. For example, the parent should close the write descriptor for the pipe and the child should close the read descriptor for the pipe.

2. All the five pipes created were shared between all the five children after forking
Context: Whenever the main process forks a child process; its entire memory the stack is replicated to the child process.
**Challenge:** In our solution, we need to create 5 pipes and then assign each pipe to each child. Doing so raises the same issue as discussed in the previous section. However, here pipes that

don’t belong to a particular child are also shared with it, making backpropagation of closure of pipe impossible for the parent to detect.
**Resolution:** Each child must close both read and write file descriptors for the unused pipe.

3. Terminating 5th child after the end of the simulation
Context: 5th child in this project has the special property that it must read input from the terminal and then communicate it back to the parent process. Another requirement of this project was to terminate the simulation after 30 seconds by closing the dedicated pipe.

**Challenge:** Initially, we were using “scanf” to read from the stdin. Now, “scanf” is a blocking call and dependent on the user’s response. Thus, the use of “scanf” stops the execution of the program and waits for USER I/O operation. This dependency created the problem when We tried to terminate the child process at the end of the simulation. Since our 5th child was waiting for the user’s input, it cannot continuously check for the simulation time.

**Resolution:** We resolved this challenge by using “select” and “read” on “STDIN”. Thus, whenever there is data to read from standard input, then only the 5th child would read from the stdin. And thus, 5th child can continuously monitor the simulation time in a non-blocking fashion.
**OUTPUT**

```
0:630.037: Parent - 0:629.263: Child 0 message 1
0:634.095: Parent - 0:629.411: Child 1 message 1
0:637.291: Parent - 0:629.458: Child 2 message 1
0:646.923: Parent - 0:629.464: Child 3 message 1
2:630.908: Parent - 2:630.589: Child 0 message 2
2:637.957: Parent - 2:630.582: Child 1 message 2
2:641.316: Parent - 2:630.629: Child 2 message 2
2:645.760: Parent - 2:630.768: Child 3 message 2
2:649.300: Parent - 2:630.767: Child 0 message 3
2:652.686: Parent - 2:630.921: Child 1 message 3
2:655.877: Parent - 2:631.503: Child 2 message 3
2:659.133: Parent - 2:631.450: Child 3 message 3
3:318.825: Parent - 3:318.672: Child 4: 1 message from terminal: new
4:631.054: Parent - 4:630.880: Child 0 message 4
4:635.032: Parent - 4:631.024: Child 0 message 5
4:643.483: Parent - 4:631.033: Child 1 message 4
4:646.724: Parent - 4:631.545: Child 2 message 4
4:650.026: Parent - 4:631.562: Child 3 message 4
4:653.636: Parent - 4:631.124: Child 1 message 5
4:658.008: Parent - 4:631.652: Child 2 message 5
4:667.327: Parent - 4:631.648: Child 3 message 5
5:313.935: Parent - 5:313.858: Child 4: 2 message from terminal: hello
6:342.253: Parent - 6:342.096: Child 4: 3 message from terminal: 1
6:631.389: Parent - 6:631.287: Child 1 message 6
6:643.838: Parent - 6:631.808: Child 0 message 6
6:654.838: Parent - 6:631.854: Child 2 message 6
6:663.546: Parent - 6:631.863: Child 3 message 6
7:208.495: Parent - 7:208.373: Child 4: 4 message from terminal: 1
8:501.845: Parent - 8:501.712: Child 4: 5 message from terminal: 2
8:653.963: Parent - 8:653.921: Child 1 message 7
8:656.893: Parent - 8:653.909: Child 2 message 7
8:658.958: Parent - 8:653.851: Child 3 message 7
8:660.941: Parent - 8:653.957: Child 0 message 7
9:563.491: Parent - 9:563.368: Child 4: 6 message from terminal: 3
9:659.372: Parent - 9:659.296: Child 0 message 8
9:663.963: Parent - 9:659.294: Child 1 message 8
9:667.067: Parent - 9:659.254: Child 2 message 8
9:670.446: Parent - 9:659.103: Child 3 message 8
10:24.010: Parent - 10:23.935: Child 4: 7 message from terminal: 4
10:873.725: Parent - 10:873.548: Child 4: 8 message from terminal: 5
11:667.133: Parent - 11:666.996: Child 0 message 9
11:670.212: Parent - 11:666.933: Child 2 message 9
11:672.643: Parent - 11:667.053: Child 1 message 9
11:675.269: Parent - 11:667.082: Child 3 message 9
13:668.805: Parent - 13:668.150: Child 0 message 10
13:683.481: Parent - 13:668.378: Child 1 message 10
13:695.645: Parent - 13:668.321: Child 2 message 10
"output.txt" 108L, 5644B
```