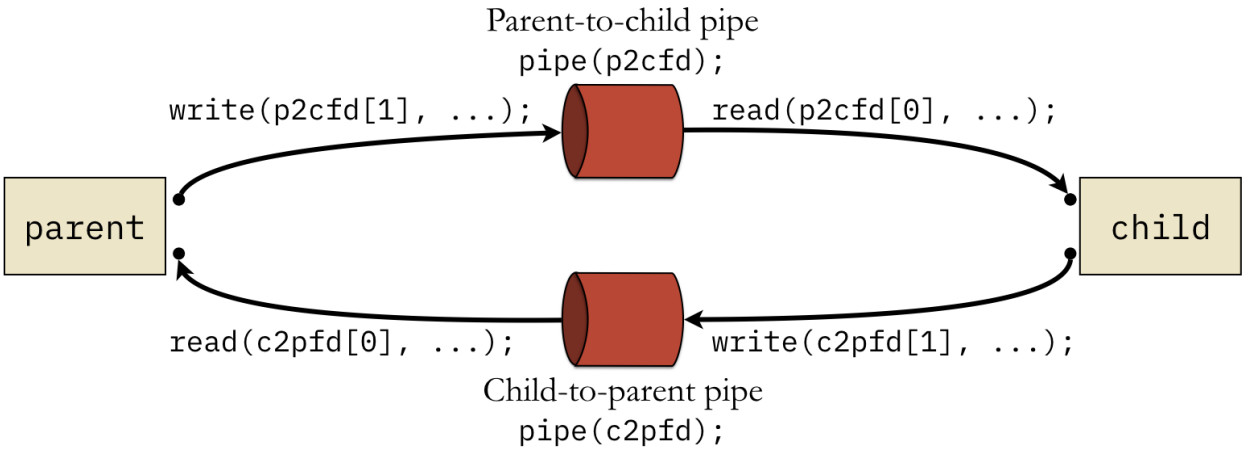
Operating Systems (CS F372)
Tutorial Sheet 3
IPC – Pipes

In this tutorial sheet, we shall explore the concepts of pipes in Inter-Process Communication (IPC).

Pipes

In operating systems, a pipe is a technique for passing information from one program, process, or command to another and can be either one-way or two-way. A pipe works on the first in, first out principle and behaves like a queue data structure. With a pipe, the output of the first process becomes the input of the second.



When a process attempts to read from a pipe that does not contain any data, the read operation typically blocks, i.e., the process will be suspended, and it will wait until data is available in the pipe. This blocking behavior allows for synchronization between processes. Once data is written into the pipe by another process, the reading process will resume and read the data.

Additionally, pipes have a limited capacity, which means they can become full if data is written into them at a rate faster than it is read. When a process attempts to write to a full pipe, the behavior depends on the type of pipe:

1. Blocking Pipes: The writing process will block until there is space available in the pipe. It will wait until some data is read from the pipe, freeing up space for new data. This ensures that the pipe doesn't overflow.

2. Non-Blocking Pipes: In this scenario, if a process attempts to write to a full non-blocking pipe, it will not block but will return an error or a specific status code indicating that the pipe is full. It's up to the process to handle this situation and decide what to do next.

Question 1:

Write a C program that creates a pipe, writes a message to one end of the pipe, and then reads and displays the message from the other end. Use appropriate file descriptors and functions for pipe creation, writing, and reading.

Question 2:

Write a C program that creates a pipe and then spawns a child process. One of the processes should write a series of integers into the writing end of the pipe, and the other process should read those integers from the reading end of the pipe. Finally, calculate and display the sum of the integers read from the pipe.

Question 3:

Write a C program that uses pipes and forks to create a pipeline between two child processes. The first child process should execute the "ls" command, and its output should be redirected to the input of the second child process, which should execute the "wc -l" command to count the number of lines in the output. This will essentially list the number of files and directories in the current directory.
Hint: Use ‘dup2’ and ‘execlp’ functions.

Question 4:

Write a C program that takes as input three positive integers. The parent process then spawns three children and communicates each of the three integers to the child processes via pipes. The child processes compute the factorial of the integers and communicate the results back to the parent using pipes, and the parent will print each of the results.