

This is the specifications for the CSCI-340 project 1. The following system calls will be used in this project:

fork, sigaction, pipe, dup2, open, close, read, write, kill, raise, wait

Specifications:

Have a parent process called `parent` create two child processes called `child1` and `child2`. Have `parent` read a message of length 20 characters from a data file located at `/user/faculty/hilzer/csci340/proj1_input` into a buffer called `first_parent_msg` using the `read` system call. Create three pipes called `pipe1`, `pipe 2`, and `pipe3`. Use `dup2`'s to have the parent redirect standard output to `pipe1` and redirect standard input from `pipe3`. Use `dup2`'s to have `child1` redirect standard output to `child2` and redirect standard input from `parent`. Use `dup2`'s to have `child2` redirect standard output to `parent` and standard input from `child1`. When complete, notice that you have created a ring of processes. Be sure to close any unnecessary descriptors in the ring. After the ring has been created, have the parent use the ring to send `first_parent_msg` to `child1` where the message is received in the buffer `child1_msg`. After `child1_msg` is received, `child1` uses `kill` to send the signal `SIGUSR1` to `parent` causing `parent` to invoke an interrupt handling routine that prints out "*Child1 has received the message*". Then have the `child1` use the ring to pass the `child1_msg` directly to `child2`, which in turn uses the ring to pass the message to the parent into a buffer called `second_parent_msg`. Then have the parent use the `raise` system call to raise `SIGUSR2`. This causes `parent` to invoke an interrupt handling routine that sends `second_parent_msg` to the file `/user/faculty/hilzer/csci340/proj1_output_MN`, where `MN` is a magic number that I assign to you. Close all open files before terminating and do not leave any zombies.

In words, here is what I want you to do: Pass a message that you read from my directory around a ring of three processes. Upon arrival at one child, cause the parent to invoke an interrupt handling routine that reports the arrival. When the message gets back to the parent, have the parent invoke an interrupt handling routine that writes the data to a file in my directory that belongs to you by virtue of your assigned magic number.

DO NOT

1. Have your processes create any unwanted relatives.
2. Fail to use the ring as intended.
3. Create orphans or leave any zombies.
4. Fail to invoke the interrupt handling routines.
5. Take any shortcuts or longcuts (i.e., do not use unnecessarily lengthy or stupid code).
6. Cheat. You know what cheating is. Don't make me explain it to you in the context of an F grade in the course.

DO

1. Adhere to the specifications
2. Design your project to compile using the `g++` or `gcc` compilers on Tiglon

You may do the project alone or you may work together in groups of 2.