Santa Clara University
COEN 383 - Advanced Operating System
Professor: Ahmed Ezzat
Winter 2021
Group 5: Anh Truong, Quan Bach, Travis Le, Yukun Zhang, Pauldin Bebla

PROJECT 6 REPORT

Overall Issue

This assignment poses some challenges to some members since the codes are written in C and not C++. Even though C and C++ are quite similar, this change requires extra effort to make the codes behave how we want them to be. It takes some extra time to familiarize with reading, understanding, and writing the syntax in C.

Additionally, not all members are familiar with **pipe()** and **select().** Even though we have been provided with the template codes for pipe and select, in order to fully grab the overall architecture of how pipe and select interact with each other, we have to spend quite a bit of time reading the manpage.

One thing to take note is that the program could not run without children. If it ran without children, the pipe would not work as expected and end up constantly reading nothing. So, the program had run with children using a fork.

The other issue is to make sure that all children processes are killed before the parent stops running. This was to prevent zombie processes and we don't want orphan processes constantly taking up CPU time.

Children Issue

Issue with debugging: While implementing the code for the child, we found that debugging child code with breakpoints is impossible. As the breakpoints only stopped when set at the parent code (i.e. main process code). When set breakpoints at the children code, the program will simply run to finish without stopping. This has made debugging children code become more challenging.

Issue with buffer size: Since the 5th child will prompt to stdout and take input message in from stdin, if the message gets too long, it will cause stack overflow and the

output will be nondeterministic. We mitigate this issue by allocating memory dynamically and increase the buffer size. In addition, we would also send messages in chunks rather than the whole user input.

Issues with timestamps and time

One issue was trying to get the time with timeval. Since the time.h only provides the time in seconds, we need to use timeval for both milliseconds and seconds. In addition, timeval was also used in select, which possibly made it easy to reuse code. To get the time, we had to call gettimeofday() to get the current time and previous time. Then, get the difference of the two to get how much time has passed.

Trying to convert the time to millisecond was a little difficult. This was because it was hard to determine what value was milliseconds. In addition, we had to find a way to throw away the first few bits. There was an issue where the milliseconds field had a number in the hundreds place because of some casting error. Unfortunately, that isn't really resolved, but it happens very rarely.

Issues with rand()

Another issue was to make sure that each process had different random values from the last. The seeds would roughly be the same if they just forked from each other. To compensate for this, the seed or number had to be the child process id + the current time. This guarantees that each child has a different seed since they should have different numbers.

Issues with buffer size

Since the parent just gets whatever is written in the pipe, it is incredibly difficult to get multiple writes without becoming its own message. So, if a child wants to write twice to the same buffer, each of those writes is its own separate message. This also places a limit on how much the user can write before the user's message gets cut off and sent as another message. Unfortunately, this issue is somewhat unresolved. So far, increasing the buffer size may have a better effect on gathering user input. However, the message will be sent in parts if the message is too big.