Message Queue Program

I. Problem Statement

The assigned task was to play “hot potato” between six processes, one parent and four children. A message queue (using mq\_overview) is to be set up. The parent process is to then use the queue to send one message after another and the children processes are to fight for the message. When one child reaches 1,000 messages, they are declared the loser. Once a process loses it can no longer receive a message. This continues until we have a winner.

II. Initial Approach

I started reading up on mq\_overview to come up with a plan. My focus was on mq\_open(), mq\_send(), mq\_receive(), mq\_close(), and mq\_unlink(). I also looked up flags and modes for the creation of the attributes structure.

The steps that I planned to take were to create five children processes, then set up a message queue, figure out how to send to and read from the queue without involving the children processes, then involve the children in the queue, and finally make them count and output the results.

III. Process/Problems

A screenshot of a cell phone

Description automatically generated

**Create the five children processes:** This part of the program went about how I expected it to. One problem was if you just keep using fork() you’ll have as many parents as children and that was not what I was looking for. To get around this, I initialized *p* as 1 and sent it into a for loop. If *p* was anything but 0 then the process was the parent (as the child’s PID is returned when performing a fork() and within the parent process and it will be 0 for the child process). This ran 5 times and I tested and confirmed that all five children shared a parent.

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**Set up message queue:** I initially didn’t think I needed an attributes structure, but I only got mq\_open to work once I created one and set some values. Since I’m sending “blanks” to the buffer I only needed a message size of 1. The mode, 0777, was to make everything readable, writable, and executable so I would have nothing in my way. I had unknown errors for a while until I realized that you had to include */* in front of the name. The documentation made it sound as if this would be automatic, but it was not.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Send to and receive from message queue:** I wanted to test the send and receive features of the message queue along with the ability to count how many messages were in queue with mq\_curmsgs. So as not to involve extra difficulty I initially tested this out only within the parent process. Once I got that working the way I wanted I included the children. At first, I would send out certain number of messages and see how many each would grab and then built from there.

The picture above shows my parentProcess() and childProcess() functions that handled the sending and receiving of messages. The last thing I added was the return 0 or return -1 depending on if the function was able to send or receive properly as a status code.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Count messages and output results:** This part was simple to implement. 5000 messages are sent one at a time and the children processes fight for them. When a child process has 1000 messages, they print their PID and close their link to the message queue and exit. For the parent process, when all messages are sent it unlinks the message queue and ends the program.

IV. Build/Final Thoughts

A screenshot of a cell phone

Description automatically generated

I ran the program 3 times to make sure that it worked. One concern I had was that the processes would finish from first to last every time or some other order that was too convenient. But it seems to not be an issue. (NOTE: The folder was named Program1 on accident instead of Program2. This is the correct build.)

I ran the build many more times than pictures here and did not notice any perceivable pattern. If I had more time and understanding I would have liked to have implemented a signal handler of some type to clean up the code and allow for some more advanced functions. However, I was able to learn about mq\_overview and how the parent and child can both still use a message queue even after all of the other variables “split”.