Abhimanyu Singh
3/23/21

PA3: IPC Methods

In this Programming Assignment, Tanzir guided us through the fundamentals of true OOP, the use of an abstract class for Request Channels made our code work great with either 3 of the classes (FIFO, MQ, SHM) that derived this abstract class, for simplicity's sake this made it so that I had to make a very small amount of additions to Client.cpp, since the handling for this was done in a couple of IF statements. The reuse of my PA1 code was not going very well so I bit the bullet and used the starter code. I implemented some for loops to take care of the request (-c) for multiple channels and I also added the tricky part of ceil (s/c) with some float arithmetic and rounding. The cleanup was done by iterating through the same bounds and sending QUIT_MSG to each chan, ending with the control_chan.

For the demo of the code, all 3 ways seemed pretty much the same in terms of timings, although there were some differences that could not be put down to margin of error (since the file sizes were so small!).

Timings:

```
$./client -p 10 -e 1 -c 5 -i f - 2.58 seconds total per channel
$./client -p 10 -e 1 -c 5 -i q- 2.58 seconds total per channel
$./client -p 10 -e 1 -c 5 -i m- 2.60 seconds total per channel
```

They all should have the same output except for debug messages

1. Run the following to request a 1 MB or larger file split through c=50 channels under each method and show the time taken. Demonstrate that the received file matches the source. Also pull your code to show that you split the file through these channels.

```
$./client -f 1MB.dat -c 50 -i f- 0.001524 //about the same
$./client -f 1MB.dat -c 50 -i q-0.001498 //about the same
$./client -f 1MB.dat -c 50 -i m-0.001871 //slightly slower than the
others
$./client -f 10MB.dat -c 5 -i f- 0.011294 //about the same
$./client -f 10MB.dat -c 5 -i q-0.011804 //about the same
$./client -f 10MB.dat -c 5 -i m-0.013281 //slightly slower
```

I think that the reason all these file transfers and requests are similar in speed is because they are quite similar, this all depends on the SSD and DRAM capabilities as well, these speeds will differ on different systems. The reason for use of one over the other is more of choice between overhead and memory safety, Message Queues are better for smaller throughput while Shared Memory can be vulnerable to memory overwrites that were not intended, for example if someone is reading from shared memory while another process is writing to it at the same time.

YOUTUBE LINKS Part 1: https://youtu.be/HXFYsbufUE0
Part 2: https://youtu.be/gf0h5p1x3wM

I apologize for the 2 links, I could not get 2 mp4 files to merge no matter what I did, on Ubuntu 20.04