Nathaniel Trujillo
12 October 2021
CSCE 313-598

PA3 Report

This program is an improvement of PA1 where instead of only using FIFO i've implemented message queue and shared memory inter process communication methods. Each of these IPC methods are a derived class of RequestChannel which contains variables that each of the classes have in common. The destructor, cread, and cwrite functions are all purely virtual and must be implemented separately for each ipc method. All three ipc methods have the similar approach to opening up the communication where a read and write "file descriptor" is used and interchanged depending on the client or server side process. The only differences are the use of different function calls. The shared memory queue is the most unique among the three since it requires a separate class to store rfd and wfd. It also uses semaphores to keep the communication in the correct order of write then read. The only changes made in client and server were implementing new user passed arguments and making sure that the correct request channel was being made on each run.

1K data points across 5 channels
FIFO: 5.81353
MQ: 5.86218 Git:
SMQ: 5.87541

10 MB file across 50 channels
FIFO: 0.051209
MQ: 0.27461
SMQ: 0.280612

Essentially in both cases it can be seen that fifo was the fastest IPC method, MQ was the second fastest method, and SMQ was the slowest method. This has a lot to do with the implementation that I used but also with the fact that the memory queue, while easier to use, comes with a cost where it still requires a more costly implementation than FIFO. Additionally SMQs use of shared memory is the slowest because it also has to rely on the synchronization using semaphores which is slower than having the kernel take care of the synchronization process.

This discovery however may be flawed since my implementation of multiple channels is flawed where it wont successfully transfer large files. It will only transfer a portion of the file. It may be the case where in large file cases that SMQ performs better but in my code implementation I wasn't able to determine this. Therefore I've made the conclusion that fifo is fastest in the simple case and in a large case fifo still remains relevant in speed.

Nathaniel Trujillo
12 October 2021
CSCE 313-598

PA1 commit git:
https://github.com/CSCE-313-Tyagi-Fall-2021/pa3-ipc-mechanisms-n-trujillo/commit/3d767c62d2653ef0c14a020521561193c8f33b7b

PA3 commit git:
https://github.com/CSCE-313-Tyagi-Fall-2021/pa3-ipc-mechanisms-n-trujillo/commit/14450a5a29e669ac64981a0f051d26c536c500e6

Demo: https://youtu.be/bG1r3RG5z-U