CSCE-313

PA 3

Aakash Tyagi

Priyanshu Barnwal

10/25/2021

|  |  |  |  |
| --- | --- | --- | --- |
| 1K points | f | q | s |
| 1 | 30.5004 | 38.56736 | 34.456 |
| 2 | 36.19692 | 35.92444 | 32.9934 |
| 3 | 29.37068 | 34.93868 | 38.79888 |
| 4 | 29.86872 | 34.60492 | 31.34828 |
| 5 | 29.61044 | 34.3808 | 38.63028 |

|  |  |  |  |
| --- | --- | --- | --- |
| 10MB.dat | f | q | s |
|  | 5.92276 | 4.64266 | 7.42141 |
|  | 5.03957 | 7.1691 | 5.59963 |
|  | 5.35583 | 5.4238 | 5.01302 |
|  | 5.39674 | 7.03189 | 5.55369 |
|  | 5.9954 | 5.06459 | 5.22764 |

Starting off, I created a Request Channel file with three files stemming off of it. This file allowed me to create three different types of channels, the FIFO channel, the Message Queue channel and the Shared Memory channel.

Starting off, the FIFO channel is the exact same as we used in PA 1. It is stored in the same place as the code and all data is stored inside that directory. The Message Queue channel has three different functions, along with the constructor and the destructor. These are the cread, cwrite, and the open\_ipc functions. The cread and cwrite function simply send and receive data whereas the open\_ipc creates the pipe to connect the client and server. We then use the send and receive functions to use the one way pipe to send and receive data. Lastly is the shared memory channel, which only has a destructor and the send and receive functions. This is because since there is shared memory, we do not need to define a piping function.

Next, I changed client.cpp. I added two new cases for the switch case function. These were for the channels argument and the ipc argument, which were -c and -i respectively. I then send the IPC method and then create a channel based on which method I decide to use. I then store any extra channels in a vector so I don’t lose track of them and I check whether or not I need to make new channels using a Boolean variable.

The server then checks the buffer capacity to transfer any needed files or get data points, similar to PA1. I create any new needed files and finally start deleting channels when I am finished with the delete function. I remove them from the vector and then delete the vector altogether.

Now let’s take a look at the tables. The data clearly shows that FIFO is the fastest way of transferring data and shm is the slowest. Although sharing memory is the most cost efficient way of implementing this code, it does take a bit longer because of the memory that needs to passes back and forth through the server and client. FIFO is the most consistent out of these. Getting the 1000 point is quite slow for message queue and shared memory. This may be because of my flawed implementation, though.

Github Link: https://github.com/CSCE-313-Tyagi-Fall-2021/pa3-ipc-mechanisms-Preebie.git