Joseph Shumway

2/11/2022

830003458

PA 1: Report

Code Discussion:

In my code, I decided to focus on modularity and simplicity. I initially didn't do so and it quickly became apparent that the new channel functionality would easily double the size of the code if I didn't separate my sections into separate functions. I made a function for requestSinglePoint, requestMultiplePoints, and requestFile, moving the variables such as opt, p, t, e, etc. outside the main function. This was done to ensure that I didn't have to create parameters and local variables for common variables.

Within the fork/exec client section, I handled the input arguments using if statements that checked the values collected by getopt(). Afterwards, I created the control channel and checked if the new channel arg was passed. If so, I then I created a new channel on the heap as well with the channel name given by the server. I then called the appropriate function for the arguments passed and passed in a reference to the channel to use. Afterwards, I sent the QUIT_MSG to the channel used, and to the control channel if it wasn't already used. Finally, I freed the heap memory previously allocated.

This implementation shifted around in its framework throughout the process of coding this assignment. In the future, I plan to take all the requirements into account and design a framework first so that I don't have to spend as much time adapting to new concepts and requirements as I go. I really liked the modular design once it was put into place and plan on using that when possible as well. Overall, I'm proud of my code given that I had no concept of memcpy, strcopy, or other system calls before this class.

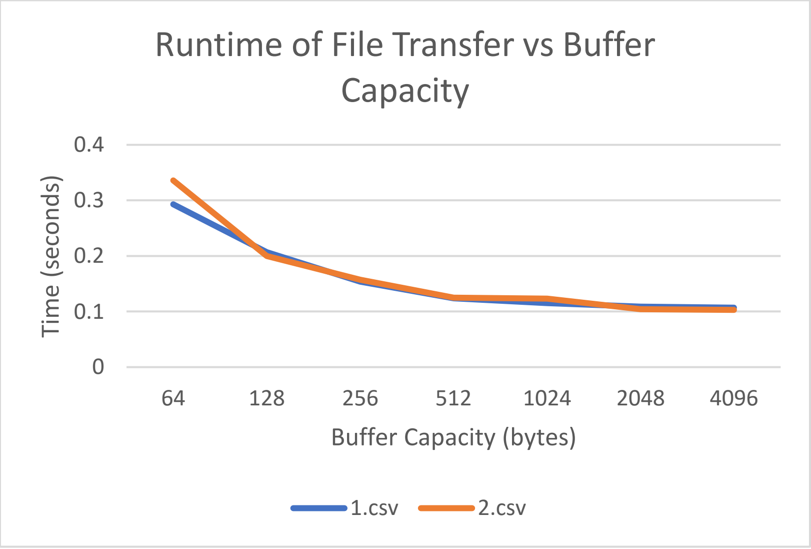
Timing Data:

1000 Data Points: Patient 1

Buffer Capacity	Time
64	5.306s
128	5.319s
256	5.328s
512	5.317s
1024	5.305s
2048	5.304s
4096	5.315s

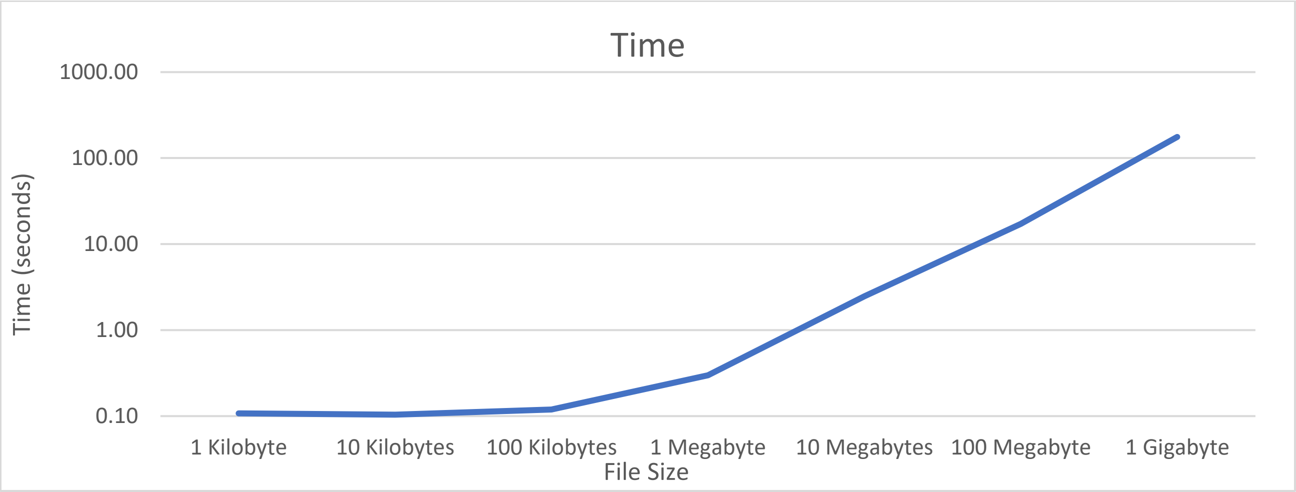
Text File Transfer:

File Name	Buffer Capacity	Time
1.csv	64	0.293s
1.csv	128	0.207s
1.csv	256	0.154s
1.csv	512	0.124s
1.csv	1024	0.115s
1.csv	2048	0.109s
1.csv	4096	0.107s
4.csv	64	0.336s
4.csv	128	0.200s
4.csv	256	0.157s
4.csv	512	0.125s
4.csv	1024	0.123s
4.csv	2048	0.104s
4.csv	4096	0.103s



Binary File Transfer:

File Name	File Size	Time
1K.bin	1 Kilobyte	0.108s
10K.bin	10 Kilobytes	0.104s
100K.bin	100 Kilobytes	0.119s
1M.bin	1 Megabyte	0.298s
10M.bin	10 Megabytes	2.485 s
100M.bin	100 Megabyte	17.194s
1G.bin	1 Gigabyte	176.383s



Timing Discussion:

For the data point transfer, it was interesting to see that the time varied, but there wasn't an obvious trend. This leads me to believe that the time depends on how the buffer is filled and how much extra space is left over. Otherwise, it's difficult to say much more about this data since the trend wasn't easy to see.

For the text file transfer, there was a very clear trend that as the buffer size increased, the time taken to transfer the data decreased. This relationship seems to be linear, as expected. The different files had more variation at smaller buffer capacities but converged more and more s the buffer capacity increased.

For the binary file transfer, there was a clear trend that as the file size increased, so did the time to transfer the file. This seems to be a linear trend, with an initial time greater than the time discrepancy between file sizes. This is expected since the time taken to transfer a file should be directly proportional to the file size.