Semester Project

CS 344

Spring 2019

Alex Hall, Matthieu Privat, James Trafny

Contents:

1. Lessons Learned
2. clientcopy.c (Alex Hall, James Trafny, Matthieu Privat)
3. clientWithMenu.c (Dr. Bethelmy, Alex Hall, James Trafny, Matthieu Privat)
4. DieWithError.c (Dr. Bethelmy)
5. linkedList.c (James Trafny)
6. transmssion.c ((Dr. Bethelmy, Alex Hall, James Trafny, Matthieu Privat)
7. createProject.c
8. editProject.c
9. HandleTCPClientWithMenu.c
10. servercopy.c
11. serverWithMenu.c
12. linkedList.h
13. menu.h
14. projectStructure.h

Lessons Learned

Over the course of this project, a few things stood out as lessons to all of the group members. The first and foremost of these being the importance of modular code. When the code is modular, it helps with the workload since certain functions can simply be called from before. It also helps with group work as multiple members of the group could work on different methods of the same file without stepping too much on each other's toes. Writing the code in modular blocks of functions also helped a lot with debugging, and it was much quicker to isolate the problems encountered. We encountered issues early on because we were not designing with modularity in mind. Towards the end of the project we got better at this, but some functions will stand out for being similar to other functions with the exception of the datatypes (see linkedList.c).

The group has also learned lessons on integration and working together on GitHub in general. For instance, we underestimated just how much time would be needed to integrate our individual work. When each of us set out to do our individual work, we had begun to rewrite code that others depended on; for example: when James was working on his code for file I/O he had to change the project structure to include a pointer that wasn’t there before, this effected the way that Alex and Matt’s code worked and caused conflicts after each merge. This took us a little longer to realize than it should have, and we only came together at the very end.

Time management was a big problem for us early on in this project. Each member of the group was involved with multiple projects and our class and work schedules did not mesh well together. The only time we found to work together was on Saturdays, but this caused a lot of time in between meetings where work was either being delayed until others finished their parts, or work was being done that caused problems after the merges. As discussed earlier, better design could have helped with the modularity, but it is our belief that poor time management on our part played a large role in us not hitting every requirement.

We also learned that we should only be changing on thing at a time, and start small, then extend. While we were developing, often times we found ourselves going too long between testing and had to rework a lot of code at a time, or had trouble finding problems in what we just wrote because we went to far without tests. We should have been refactoring the code each time before we added the next feature to prevent “spaghetti code” from getting unrulily. We also learned that everything takes longer than you think. We routinely ran into unexpected bugs and errors that required huge amounts of time to fix before we could move on; we should have accounted for this from the beginning.

Overall, we are dissatisfied with the final product that we are delivering; however, we all agree that we learned a lot more about how to work together as a team, and how hard managing large systems can be. If we had more time (or if we had started earlier more earnestly,) we feel that we could eventually meet the requirements laid out for this project; we feel that we are submitting more of a “walking skeleton” or proof-of-concept as opposed to a finished product.