**CSCI 3431 Operating Systems**

**Iteration 2 Report**

Tsubasa Hirooka

Lianzhu Shi

Yiying Zhang

Yucheng Liu

**1. Introduction**

Our group decided to build up a simple UNIX shell, which basically has four functions:

history, pipeline, redirection and tab completion. The shell is written in C language.

**2. Functions**

**History**: assign to Tsubasa Hirooka

The history function would display all the commands the user has entered so far. There are two main functionalities: adding a command using add\_history() function provided by a library, readline/history.h, and viewing commands using do\_history(), our own function. For data structure, we use a two-dimensional array of characters (a global variable) to store commands. Our shell can hold 10 commands maximum. The strength of this function is that we can reduce the number of lines of codes using a library function, add\_history(). A simple data structure also makes simple logic in do\_history() when showing commands (loop 10 times and show each command in the data structure). Since we can hold limited number of commands, we do not consume memory.

**Pipeline**: assign to Lianzhu Shi

**Redirection:** assign to xxxx

**Tab completion**: assign to Yiying Zhang and Yucheng Liu

**3. Challenges**

One of the technical challenges in our project is GitHub. Since we had used GitHub a few times before this course, it took time to be familiar with GitHub commands, branches, conflicts, collaborators, and so on. Especially when conflicts occurred, we did not know the step to solve it even though we learned it in tutorial. So, we asked Professor Valencik about commands to solve it. Furthermore, our weakness in GitHub is that we did not commit codes so often and we did not leave comments so much. Although we divided our tasks, and we know who are responsible for those tasks, we should have committed and commented frequently, so that we could share our ongoing tasks. For example, each time we finished one function, we could commit and leave a comment. Through comments or the number of commit, we can see how far each member to complete his or her task and what kind of problems he or she faces. These are the problems of our project.

Another challenge is that it was difficult to combine four functionalities in one source. At the beginning of the project, we worked individually and wrote code in different files, and later we decided to maintain those four functionalities in one source. It is because those four functionalities are related to each other and it is easier to test the codes for demo. We combined them during meeting, so when some errors occur when compiling, we could fix them easily. Finally, Yiying and Yucheng took a responsibility for the combined source and fixed small bugs before demo.

In terms of communications, we did well through email and meetings. When someone was absent in the meeting, we let him or her know what we discussed in the meeting. We also tried to distribute tasks, coding and documentation, equally. Although we did not have a leader in our team, each member tried to help each other and contribute to our project.