IRIS NICOLE CARSON
ANTONIO RAFAEL CASTRO
JOSHUA MANUEL LOUISE KEMPIS

4 BS CpE

February 6, 2022

Project 1 Second Progress Report

Rudimentary Shell Interpreter

Creating a Command Parsing system

Using the *getline()* function, *stringstream*, and the *>>* operator, a system was created that reads the user inputs by up to a certain amount of arguments/parameters. The range at which each command is read accounts for up to 8 possible parameters; this can be manipulated by changing the value in the *ARG_MAXX* variable. Each user input is read by the virtual terminal as a stream of this format:

cmd	<parameter1>	. . .	<parameterN>

Where parameterN represents the last parameter read by the system .

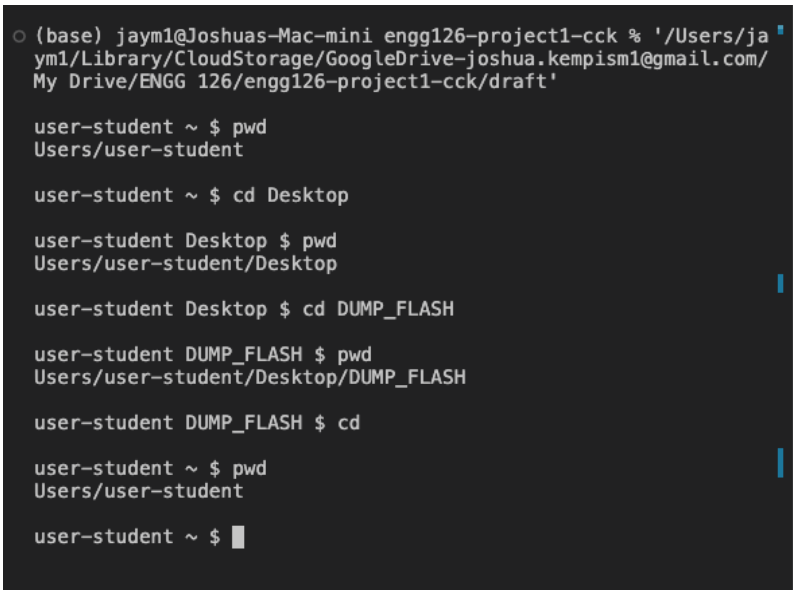
"cmd" or command refers to the first string input read in the stringstream, this determines which function or action is being called in the virtual terminal.

Testing model for a working virtual directory

The virtual terminal runs with a virtual directory that begins at "Users/user-student" this is what is referred to in the code as the home directory. When the user calls on the `cd` command, the directory appends the pathname into the directory and updates the prompt. This virtually moves the working location of the simulated terminal.

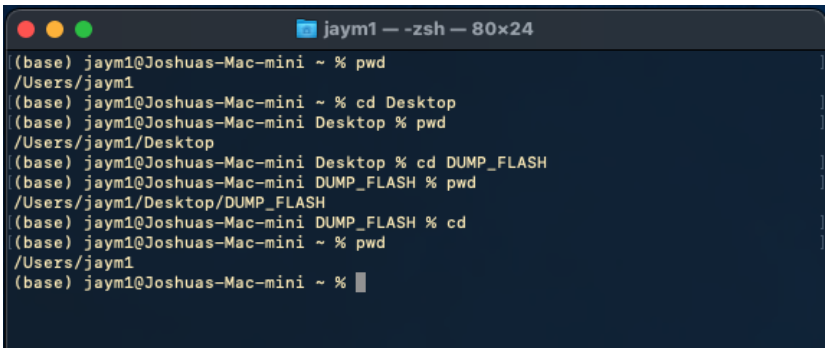
Several implementations such as location verification is yet to be coded into. As of the moment, the code accepts any change directory command input as a valid directory. However, assuming as of the moment that all cd-command inputs are valid directories, the virtual terminal is able to implement the cd and pwd UNIX commands accurately.

By comparison, the images below compare the UNIX command implementation between the virtual terminal and the actual terminal from the Mac operating system.



C++ UNIX Terminal Simulation (LEFT)

MacOS Terminal Application (RIGHT)



Conclusion

The project as of the moment is around 5% complete. There is still a bit more that needs to be understood along UNIX-based commands and how a basic shell interface works.

Task	Description	Status
Project Research	Search and familiarize UNIX based commands and how virtualizing a shell would work in a C++ program	PARTIAL
Preliminary Code Implementation	Begin Coding the basic structure of the project	ONGOING
Basic Input Commands	Recognize basic user inputs in the program loop. Implement a user-input termination command.	ONGOING
Output & Feedback	Produce feedback and outputs in the program	TBD
Debugging	Code polishing	TBD