**Introduction**

This is an operating systems project to implement an alternative Linux shell in the C programming language. The shells command prompt should consist of the username and the hostname of the system it is currently running on. The shell should be able to accept commands from the user and return the expected results to the user.

The project task was completed by Tom Gallagher, Maximilian Mandel, and Dennis Waswa Simiyu.

**Approach**

The approach to the system was to create a command line by using a while loop and creating a command prompt with the username and the hostname, where the user can input the commands.

The inputted commands were read using fgets and stored into a char array.    
The commands then needed to be split into parts so that they could be interpreted further in the system, this was done using strtok with a blank space as a delimiter (a value that denotes where the input should be separated). The inputs could then be checked using if/else statements and strcmp checks to compare the inputs with our predetermined calls.

Exec: The exec command was implemented by using fork, execv and wait functions. Forks were used to allow the child processes to be executed alongside the main process. Execv was used to execute the child process, as an input it takes the function to be executed as well as a list of further parameters. We used a simple for loop to create this list of parameters. The wait function is used to make the parent function wait for the child process to finish. It does this by not allowing the parent process to continue until its child process wpid returns a specified value, notifying that it has terminated.

Globalusage: The globalusage command was implemented just using a printf statement.

Modifiers &,>: The modifiers & and > were implemented by splitting the arguments in specific ways. The & modifier is implemented by checking the last element of the inputted commands, if the & modifier was present it then the exec function was called without the wait function present, therefore allowing the parent process to instantly continue without the child process terminating.     
Furthermore, working with files was necessary for the “>” modifier to work. To check whether the > modifier was present a for loop was used to iterate through the elements of the array of inputted arguments, if the modifier was found the loop was broken to allow the process to continue. The file name from the commands was parsed by taking the element immediately after the > modifier and opening the file using C fopen command. Fprintf was used to write the outputs into the text files.

Quit: The quit command needed to list all the running processes, which can be achieved using fork and by requiring a yes or no input if the program is exited or not.

**Files:**

There is only one main file “imcsh.c”, which contains all of the code and the corresponding comments for it.

Some additional files were used to perform informal testing for the functions, these have also been included. “test.c” “exec\_test.c”

**Code Platform:**   
Github was used to store the code, every member was responsible for uploading their individual code during the building process so that other members could view their progress, also allowing us to code in a more standardized fashion by making sure our code would work when integrated together.