This project has taught me how to take input from the user and develop an efficient way to carry out processes through a command prompt. Major subjects include multiprocessing, execv functions, and memory allocation when handling information. **To execute the code, simply run "make" and then execute with "./shell"**. I designed my project to first check if the user has input a customized prompt when the program executes. If so, the prompt is saved and then goes into an infinite loop. The infinite loop reads anything the user inputs, executes it, and then prompts the user for another command.

After taking an input from the user, the readInfo() function is used to store each argument in char pointer array. This makes determining the user's command more easily. From there, a set of if statements determine if the user input a built-in command. If so, the corresponding command is executed and prints the results. Otherwise, the program forks off and and waits for the child process to finish. The child process uses the execv() function with the command (first value in the array) and the array of arguments as parameters. If the the command exists, it the child process executes, prints the results, and then terminates. Meanwhile, the parent process waits for the child process to terminate and prints the exit results. If there is an "&" at the end of the argument of a process that isn't built-in; the parent process does not block the prompt, allowing the user to continue executing commands while the last child process runs in the background. This is done by using the WNOHANG parameter in the parent process' waitpid function instead of waiting for the corresponding child process