SHELL Report

(1) Code Organisation :

- I wrote my shell adapting functional programming.
- Every part that could be separated to make debugging and reading of code easy was made.
- So the flow of my code is as follows:
    1) You call the shell from your Unix Terminal.
    2) Depending on whether you decided to make it batch mode or interactive mode it will function accordingly.
    3) My code contains helper methods, Core implementation methods, Main method, Parsing methods, Printing methods, and Validating methods.

(2) Main Functions :

Method	Description
`int areEqual(char* s1, char* s2)`	Checks if the two input strings are the same
`char* getEnvVar(char* envName)`	This method gets the environment variable value recursively
`char** parser(char* input, char split)`	Parses the input command splits on split returns an array of arguments also it handles Environment variables
`char* getCommand(char** src, char* command)`	This method takes the binary files and checks which file has the given command using the access() method
`void signalHandler()`	This method prints into the log file when a child process terminates
`void brain()`	This method handles modes of the shell, Reads a string from stdin ,Checks if the command exists,Executes it or handles errors

`int main(int argc, char * argv[])`	Main method call from terminal *if argument is file the ==>* Batch mode else Interactive mode set mode then run the brain() method

(3) Manual:

(1) Open Terminal.
(2) Locate the folder containing source code.
(3) Set where you want to put your history and log files on your machine.
(4) In your Terminal type -> make
(5) Then ./Shell [arg]
(6) Your in my Shell.

(4) Notes:

- Ksysguard is not supported on my Mac, so I failed to complete this part of course.
- You have to handle path of History/Log and the '~' on your machine in the global variables