Volkan Ozturk 2019400033

Project 1 Report

Implementation:

I implemented the basic shell as required in the project description. I satisfied the MUSTS that are required:

1) My shell is working with fork/exec operations, every command works in this way. Except for the built-in bello() and alias functions.
2) Instead of using execvp(), I implemented search mechanism for the PATH environment variable. I then used execv() for the execution for the given path.
3) My shell does not terminate with an error, if the execv() function experiences with an error, strerror(errno) is printed to the stdout. The shell session can be terminated with the exit command. SIGINT interrupt (Control+C on Macos) does not end the shell session as it also does not with the regular shell.

I also implemented all the CANS and SHOULDS parts as required in the description. The necessary functions can be found in the project code with the explanatory comments.

Assumptions and Clarifications:

The number of processes part of the bello() function is counted with running “ps” command. More explanation can be found on the relevant function’s comments. (get_current_procs())

The lengths of; cwd, hostname, prompt, total_num_processes, and the output of “>>>” operator are all assumed to be maximum of 1024 chars long. This is chosen for easiness; flexible length implementation is also possible.

Difficulties Met:

Implementing the functionality of execvp (i.e., searching through PATH) and other MUSTS were straightforward to implement. But the additional complexity comes from the newly introduced functionalities. Implementing alias was a bit hard, but with the help of the parsing and conversion functions, I implemented them. The most challenging part was to implement newly introduced redirection operator “>>>”. Since we do not know the output of the command, I was needed to use intercommunication mechanisms. I decided to use pipe, and with this approach the implementation of the “>>>” operator was successful.