<u>Quiet a Shell: Report</u>

1. Executables without arguments: This is the feature where the command has no arguments, i.e a single command like ls or cd. We implemented this feature using execvpe() library call, implemented in our Quash.cpp Run function. The execvpe takes in the program path, the program args which is a NULL terminated char* array, and an environment PATH variable that we keep as a private variable.
2. Executables with arguments: This is the feature where the command can have arguments, i.e. ls -a or ls -l. We implemented this feature in the same way as the executables without arguments were being handled, in this we just passed in the arguments to the execvpe() library call as well.
3. set for HOME and PATH work properly: This feature sets the HOME and PATH environment variable for our Quash. To implement this we made a private path variable that keeps the path environment variable for our Quash. We have setter/getter functions for both the environment variables we support in Quash. The setter functions do not replace the existing path, rather it updates and adds the new path with the old paths and is ':' separated.
4. exit and quit work properly: This is the feature that lets us exit/quit from our command line interface of our Quash. We implemented this feature using exit(0) library call. When we see exit/quit input in our terminal we call exit(0), which exits the current process.
5. cd (with and without arguments) works properly: This is the feature which can let us change our current working directory. We implement this feature using chdir() library call. If there is no argument provided to cd, this will change the current working directory to HOME, else if an argument was provided i.e. a directory was provided. We check if the directory exists, and can be accessed and we change the current working directory to that using chdir().
6. PATH works properly. Give error messages when the executable is not found: This is the feature which lets us have our own path

environment variable that could be different from the main extern path environment variable. We have made a private path variable per Quash, which keeps the Path variable and so we have get/set functions for it. We also pass this environment variable to the exec call using execvpe(). If the executable is not found in the given path variable directories, the exec will give an error.

7. Child processes inherit the environment: This is the feature where the child process forked from our Quash using fork system call can have the PATH environment variable passed to it. We implement this feature using execvpe() library calls, which lets us pass the environment PATH variable, which is a private variable that is different from the main extern PATH variable.
8. Background/Foreground Execution: Jobs can be called either in the foreground or background of QUASH. To run a job in the background, call the cmd with an '&' at the end.
9. To see the jobs that exist in the background, run 'jobs'. To return the last job in the background to the foreground, use 'fg'. To see what is the last command in the background that would be restored with 'fg', use 'bg'. In addition, jobs can be restored with '%<jobid>'.
10. File redirection: This can be done in quash by tying a cmd, either '<' or '>', and the file path. Then for reading in, the command will run with arguments passed line by line on the input file. For writing out, the users command is run, and the output piped to the file rather than standard out.
11. Allow (1) pipe (|): This is the feature where ipc is allowed using anonymous pipes, this lets us pipe the output of our first program (the command on left) to the input of the second program (the command on right). We implement this feature using pipe() library call, and using dup2() to duplicate the file descriptor of the pipe to Standard out/in respectively.
12. Cmds from prompt and file: The user can run commands within QUASH by either starting a QUASH and using its command line interface via "./Quash" and "> cmd". The user can also run

commands from an input file by starting the QUASH via "./Quash" and running "> quash < <infile>".

13. Extra Credit: Killing background processes: using the 'kill' command within quash with either a '%<jobid>' or 'pid'.

Partners:

Saharsh Gupta: 2826224
Blake Rieschick: 2829512