Rory Reidy
QUASH Project Report
October 22, 2021

This is a report on my version of Quite A Shell. Here I will describe my implementation of the shell, including the building process, any difficulties I had, and the ways I tested the shell.

The first feature I implemented was the job system, which seemed rather easy. I designed a class called JobList, which contained a vector of “Job” structs. Each Job contained a process ID, job ID, name, and foreground/background status. I made it so that whenever the child executed a command, the parent added it to the jobs list. Any command that failed was removed from the JobList.

Next, I worked on parsing. This was the most difficult part of the process. I designed helper functions to parse strings based on characters, and for recognizing pipes, ampersands, spaces, etc. Originally everything was using strings, but I ran into difficulty when I tried to exec with strings.

So I changed the parsing from string to char*. This took some time, but when I tried to implement this with execle(), no arguments could be recognized. Only single commands were being run. I adapted to this by parsing everything into char**, arrays containing each argument, then called execvp().

I started off handling environment variables PATH and HOME through a proxy class which altered the envp argument, but eventually switched to directly altering the environment with getenv() and setenv(). The execvp() call passes the path to the child, which has seemed to work.

Exit and quit perform an exit(). If there are no arguments, cd sends the user to HOME. If there are arguments and it is a valid location, cd works normally.

My QUASH only allows for one pipe. If there is a pipe, a special function is called which calls the normal execute twice, piping the IO.

Background/foreground execution may or may not work as well as it should. I was not sure exactly what was expected, so it got put on the back-burner. Try putting an & at the end of your command.

File redirection works well in a shell setting. When each command is physically typed, it can output whatever is needed into a file, or read from a file. When commands are fed into QUASH from an input file, output redirection is inconsistent. Input redirection works from a source file. For example,
“grep path < main.cpp”
returns
“//cout<< path <<endl; CORRECT” and “cout<<”Arg not recognized. Try ‘set HOME=/filepath…./filepath””
This output is the same result as searching main.cpp directly.
However, “ls > out.txt” does not write anything to out.txt if it is called from a text file. It is an enigma.

This is a general summary of my project and its implementations and limitations. QUASH was a difficult project, and had I started it sooner I feel I could have made a fuller implementation.

My most current test file:

```
ls -l | grep r > out.txt
ls -l > ls.txt
cd ..
find -name *.pdf   ////takes a long time to execute
grep path < main.cpp
quit
```

Output:
out.txt:  partially as expected: redirects correct output to “out.txt1”
ls.txt: as expected
hi.txt: prints 8 pdfs
grep: (standard input): Bad file descriptor// to reiterate, this is inconsistent. sometimes it works, sometimes it doesnt