QUASH

Quite A Shell

Blake Morrell

Matthew Felsen

Wednesday 9AM – 11 AM

Abstract

Quash (Quite a shell) is a Unix based text interface. Its commands and functionality are modeled after popular Linux programs like Bash and Shell.

Features

Command Name	Description
cd	Changes the current working directory
echo	Prints out the value of environment variables or any string following command
exit	Leaves the Quash interface
jobs	Lists all the processes running in the background
pwd	Displays absolute path of the current working directory
set	Sets the value of a environment variable
quit	Leaves the Quash interface

Symbol	Description
&	Background execution
<	Redirect standard input from a file
>	Redirect standard output to a file
|	Pipe - Unilateral communication between processes

Implementation

int main(int argc, char** argv, char** envp)

Responsible for setting up environment. main() begins by masking signals and setting up necessary dependencies for redirecting file I/O. Also passes arguments like environment variables to Quash.

void quash_run(m_command* qcommd, char** envp)

Handles commands and symbols passed in from main and directs the proper function to execute for the given command or symbol.

int command_logic(m_command* qcommd, char* envp[])

Controls command flag logic for pipe, background and file redirection.

bool m_get_command(m_command* qcommd, FILE* in)

Takes command qcommd and splits it into tokens to be easier for other implementations. The tokens are stored in a structure called m_command which contains the raw string, the length of the string, an array of the tokens, and the length of the array.

int prim_cmd(m_command* qcommd, char* envp[])

This function runs commands stated in the table that are free of symbols. This is done using the execvpe() system call. This function also uses execvpe() to pass variables into the environment.

int input_io_cmd(m_command* qcommd, bool i_o, char* envp[])

Same implementation as prim_cmd(), but has additional handling for file I/O redirection symbols. (<, >)

int m_cmd_background(m_command* qcommd, char* envp[])

This function executes commands with the background execution symbol (&). This is done by creating a child process which inherits the same environment as its parent.

int m_command_pipe(m_command* qcommd, char* envp[])

This function runs commands with the pipe symbol (|). In this function, a pipe is created to unilaterally communicate from one process to another.

Structure

The backbone of our project revolves around the m_command and m_job structures initialized in our header file. These structs allow for simple command manipulation and flow. By passing in qcommd to nearly every function, we can easily redirect/buffer data passed between our implemented functions. The straight forward direction leads to enhanced usability and flexibility.

Testing

To test functionality, the given test cases were used. This was done by using file redirection of the .txt files into quash. All test cases passed except command7.txt. New variables could be added to the environment but said variables could not be inherited by the child process. Below is a table consisting of test results.

Test Case	Output
Test 1	Linux
Test 2	Linux matthew-VirtualBox 5.3.0-45-generic #37~18.04.1-Ubuntu SMP Fri Mar 27 15:58:10 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
Test 3	./ /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/ snap/bin:./
Test 5	/home/matthew/Documents/EECS678_ProjectQUASH /usr
Test 6	bash6.txt command3.txt command7.txt ls NOTES quash.h command1.txt command5.txt expected_output ls.c quash quash_test.h command2.txt command6.txt file7.c Makefile quash.cREADME.md My Dummy program is run
Test 7	Command: "./file7" not found.

Conclusion

Similar to Bash and csh, QUASH provides a user-friendly flow that allows for running executables and manipulating files. Commands like cd, set, quit, jobs and pwd were implemented and functioned perfectly. Using execvpe() to filter commands, we can execute similar Bash programs (with or without arguments) not directly handled by our program. Whether you want to read from a file or make a program for you grandma, QUASH can do it all.