3/7/2015

Project 1 Report

Quite a Shell (quash) implementation

Tyler Steiner, Michael Esry

Project 1 Report

Quite a Shell (quash) implementation

Introduction

Problem at hand

The problem at hand that we were given was to implement a shell program similar to bash. A shell program performs basic interaction with a user, including directory navigation and program launching (both in the foreground and as a background process), as well as other basic interactions. The implementation must be done in the language C.

Proposed Solution

The proposed solution will be a single program that will act as a shell program implementing the requirements as given. It will include a README file to inform users about possible commands, proper syntax, as well as other miscellaneous information.

Implementation

Discussion

The project was successfully implemented using a single .c file. As per the requirement, quash successfully implements its own set, cd, jobs, quit, and exit functionality. Each of the built-in functions is further described in the README file for quash and all of the functions are listed out if a user types "help".

Our group very heavily utilized github to split up the work and share the resulting code. While lacking a formal separation of the tasks needing to be implemented, our group used heavy communication, in association with github, to split up, work on, and finish the project.

Each part of the program produced its own set of challenges. For example, an overarching problem of getting and dealing with the string specifying the input command was something that had to be dealt with early on. This was further complicated by a very limited knowledge and experience level with the C language.

Testing

Procedures

There were two main phases for testing this project. The first testing stage was a normal type of testing that is done while coding. The main difference being that it was done by both members of the group. When one member of the group would successfully implement a feature, they then reported that to the other person who, after pulling the code down from github, would test out that feature to see as well.

The second phase of testing was much more involved. It was done at the end of the coding process. At that point, all of the features and requirements of quash were tested one by one, using all possible permutations of a command that could be required. The results of those tests are in the table below.

Results

Command	Additional arguments	Pass/fail (and result)
cd	(blank)	PASS (Moves to specified home directory)
	..	PASS (moves up one directory)
	</directory/>	PASS (moves to specified full directory)
	<directory>	PASS (moves to current working directory plus additional directory and appends '/')
set	(blank)	PASS (does nothing)
	PATH=</directory/>	PASS (Sets PATH to specified directory)
	PATH=</directory/:/directory/>	PASS (Sets PATH to multiple directories)
	HOME=</directory/>	PASS (Sets HOME to specified directory)
	(random string)	PASS (Does nothing)
jobs	(blank)	PASS (lists current background jobs)
	(random string)	PASS (Does nothing)
quit	(blank)	PASS (Exits quash)
	(random string)	PASS (Exits quash)
exit	(blank)	PASS (Exits quash)
	(random string)	PASS (Exits quash)
./	(blank)	PASS (Permission denied by OS)
	(Existing Executable)	PASS (Executes executable w/in quash)
	(Non-Existing Executable)	PASS (Returns not found)
./ &	(blank)	PASS (Does nothing)
	(Existing Executable)	PASS (Executes in background)
	(Non-Existing Executable)	PASS (Does nothing)
<		PASS (Input succeeds, last command of file must be exit or quit)
>	(other command and new file)	FAIL (Does not direct to file, but executes)
> &	(other command and new file)	PASS (Redirects output to file)
|	(2 other commands)	PASS (pipes output of the first command to the second command)

Concluding Remarks

This was a very intense and detailed project. It involved a lot of first learning the C language better, followed by using that knowledge to implement the desired program. The end goal functionality was nearly 100% of desired. The extensive testing that the project underwent should also have caught any bugs that were introduced or any end functionality that was not to specification.

Appendix

README

```
*********************************************
*   #####  #     #    #     #####  #     #  *
*  #     # #     #   # #   #     # #     #  *
*  #     # #     #  #   #  #       #     #  *
*  #     # #     # #     #  #####  #######  *
*  #   # # #     # #######       # #     #  *
*  #    #  #     # #     # #     # #     #  *
*   #### #  #####  #     #  #####  #     #  *
*********************************************
```

I. Introduction
Welcome to the quash, quite a shell. Quash was designed as a programming assignment for EECS 678: Introduction to Operating Systems at the University of Kansas. This README was intended to be opened upon using the provided Makefile to compile the project.

II. Building and launching quash
In order to build quash, please type "make" into the command line of a unix based computer. This README should launch in vi to assist with the syntax of quash. Please see section III for syntax specification. To launch quash, type:

"./quash </home directory/> </path directory/>"

where home directory is the directory that you would like to set as home, and path directory is the directory or directories (separated by ':') that you would like to have as your path(s).

III. Syntax Specification
For all path specification, AN ENDING '/' IS REQUIRED.
The following commands (and their proper syntax) are included with quash.

cd [PATH]
-Changes current working directory to specified.
-PATH can be empty, a folder within the current working directory, or be a full path specification.

set {[PATH/HOME]=/<directory>}
-Sets path or home environment variables accordingly.
-PATH or HOME must be specified.
-if more than one directory is to be added to path, separate by ':'
Example: set PATH=/usr/bin/:/usr/

jobs
-Displays any running jobs.

quit
-Exits quash.

exit
-Exits quash.

./[EXECUTABLE] [OPTIONAL ARGUMENTS]
-Any executable can be given as long as it is either in path, or current working directory. Additional

optional arguments can also be given which are passed to the program.

&
-Appending '&' on to any executable allows for background execution

<
-Can be used to run a list of commands from a file. NOTE: the file must have a final command of either exit or quit.

>
-Used in conjunction with '&', '>' allows all output of the background executing program to be redirected to a file. Does not allow the same with programs not running in the background.

|
-Allows output of one command to be piped as input to the next command. Up to 4 pipes may be used.

kill [pid]
-Allows for termination of ongoing background process.

IV. Conclusion

Thank you for trying quash. As a work in progress, this program is subject to revision. Please contact either Tyler Steiner (tsteiner@ku.edu) or Michael Esry (m716e163@ku.edu) for any questions or comments.