Project 1 : Implementing a Shell

Team Members: Evan Lee, John Cyr, Abraheem Omari

Problem Statement

Problem Statement: Design and implement a basic shell interface that supports input/output redirection, pipes, background processing, and a series of built in functions as specified below. The shell should be robust (e.g. it should not crash under any circumstance beyond machine failure). The required features should adhere to the operational semantics of the bash shell.

Assumptions

The task was to implement a basic shell that handled a number of user commands and operations. In our shell we assumed that user would enter a shell command followed by a list of parameters

Steps Taken

Many of the smaller parts of the code were handled individually and coded simply on our own. When we encountered larger issues we often scheduled meetings to discuss approaches and discuss structure design and implementation. Many times discussing the code was very helpful and resulted in better code. If there were ways to delegate pieces of the code out to each member we did and tried our best to share the load of the project.

Major bug issues were best resolved in meetings working together to trace errors in the code and brain storm why issues were occurring. Debugging meetings resolved a large number of problems we had with our code and was a very effective use of time.

Typically our process was to enter a test case and if it didn't work trace it in the code. Then attempt to fix and try again until we solved the problem. Then we did this hundreds of times.

Chosen system calls and Libraries

The system calls that we are using are for getting and setting environment variable values and executing non built-in commands in our shell.

```
#include <string.h>        // c-string functions
#include <stdio.h>         // standard print/input
#include <stdlib.h>        // for malloc/realloc
#include <unistd.h>        // major library
#include <fcntl.h>         // check if directoy exisits
#include <sys/types.h>     // for use of pid_t
#include <sys/wait.h>      // for use of wait_pid
#include <sys/stat.h>      // file and directory properties
#include <sys/time.h>      // for etime
#include <errno.h>         // for error catching
```

Problems encountered

Limits outputs the contents of the /proc file, but doesn't print the output of the actual command.

Can't print a processes place in queue when it finishes. Only when it is started.

$PWD is whack, yo

grep doesn't work

Division of labor

John Cyr – parsing implementation, formating and outputting prompt, environmental variable replacement, FILE I/O opening/checking, command struct design and implementation, echo and exit command, Report

Evan Lee – path resolution implementation, Execution of none built-in commands, I/O redirection implementation and design, piping implementation and design, built-in implementation, code commenting, merging and debugging

Abraheem Omari – built-in limit and etime implementation and design, overall structure design, code commenting, debugging and testing, README, working out Zombie process issues, background processes modification and optimization, Makefile creation

Number of slack days used per team member

Evan Lee – Use 1 Slack Day

Abraheem Omari – Use 1 Slack Day

John Cyr – Use 1 Slack Day

Meeting Times, Modifications, Decisions Made, Accomplishments

09-03-2015 – Meeting - Initial Meeting to discuss overall project ideas and code collaboration methods. We decided initially to try working with a shared Dropbox archive.

09-05-2015 – Meeting – Distribution of work to each member to work on basic parts of the project including built-ins, path resolution and environment variable handling. Discussed changing our code sharing method to Github

09-10-2015 – Migrated code to a Github repository and pushed all our code. At this point we have prompt, built-ins and environmental variable handling written.

09-12-2015 – Meeting – Discussed changing structure of project moving towards adding piping to the shell. Major changes to the container for each command to a struct to hold all information of each command to make handling easier.

09-15-2015 – Fork/Exec Implemented and tested. Worked to merge newly written code between group members. Each re-learned how to use Git

09-17-2015 – Meeting – Met at Majors lab to talk about file redirection approaches and issues with code. Decide to move some code out to header files

09-24-2015 – Completed major code that implements piping and wrote a struct building function to make code easier to read and use.

09-27-2015 – Meeting – More code integrating and looking forward towards trying to fix background processing – optimizing code

09-28-2015 – Debugging and working out many formating and processing errors that are coming up now as we try out test cases. Begin commenting and writing both the README and report. Decided to take a slack day and work out issues we couldn't solve in time

09-29-2015 – Finish off as much of the project as we could before packing and submitting our project

Descriptions of additional features – Extra credit

Added built-in to clear terminal while working in the shell

Will handle *n* pipes

Can handle both redirection and piping simultaneously