Dustin Horvath
Nicholas Roudebush
[EECS678] Project 1
Spring 2016

Overview:

The purpose of the QUASH or Quite a Shell project was to understand the concepts that the UNIX interface implements to parse user input to process into commands. Through the first few labs we learned how Fork(), Dup2(), and PIPE() each play a part in executing user input. QUASH tied all of these together with signal handling, process groups / id’s, and environment handling.

Problems:

Most of the problems that we encountered in this project involved the lack of string handling libraries of the C language. We feel that the language of choice added unnecessary complexity. The biggest issue we faced was the handling of the environment variables. The semantics of each function that handles a new task is extremely important; for example, calling fork() vs exec() vs pthread_create(). As a student these can seem very similar at first, especially when implementing a function that seems to work.

Experience:

We are most definitely comfortable reading and interpreting the “man pages” applicable to “application” level system programming. We learned a lot about GDB and how to debug intricate problems, but we also feel that using a modern compiler would make the debugging much more user friendly and would’ve removed a considerable amount of necessary debugging. QUASH was mostly tested against the output and behavior of the shell running on the cycle servers. Most of the debugging was done with carefully placed print statements and GDB’s ability to display variables while stepping in and out of functions. This was especially a big help while figuring out why we were not able to use “export” to change the environment variables for the parent from the child process.

Implementation:

Most of the implementation of QUASH is memory handling and string handling to pass a “valid” command to “./sh -c”. The piping was handled by creating three cases. The cases are set up to handle the first pipe, the last pipe, and any number of pipes in between. These cases also depend of the STDIN/STDOUT requirements of each command. The environment variable handling was managed by the creation of a local environment data structure that was safe to pass from the child to the parent. This had the consequence of nullifying any command run using the form “./quash arg1 arg2” from the cli that utilizing environment variables, because those variables are explicitly designed not to damage the parent process.

For commands that could not be run by /bin/sh and those that require special handling within the program, the first words were compared to handle exceptional cases. These include ‘cd’, ‘set’, and ‘jobs’. These commands were handled internally, rather than being passed to the fork/exec function for execution.