Ryall Edwards

Jeremy Corbello

COSC 4302 Project

12/3/2018

The purpose of this project is to create a basic shell program. A shell is a program that takes user input in the form of commands and interprets them, allowing the user to perform functions and access kernel services. This particular shell program prompts the user for a command then parses and executes said command in a child process. The child process is created using the fork() function. The parent process waits while the child process executes the command then terminates. Then, the shell itself terminates as well. So, any time a command is given, the program creates a new child process, executes the command, then ends the process. This is necessary so that the initial parent process doesn’t fail causing the machine to crash. The shell interprets commands using a series of functions.

The function printPrompt() creates a string containing whatever information is desired. In this case we used the current directory as the prompt. This prompt is called within a loop in the main method so that after a command is received, it prompts the user for a new command. The command is read and interpreted using the functions readCommand and parseCommand respectively. The readCommand function is called first in the main method to retrieve the contents of the command line. Then the parseCommand function is called to interpret these contents. It also allocates memory using the malloc function. Once the memory has been allocated and the commands interpreted, the program must find the command files in order to execute the command. It can do this in one of two ways.

The shell utilizes the PATH environment variable to locate the command that the user attempts to execute. The PATH variable is an ordered list of path names that tell the shell where to search for the given command.

The program can be run through the LINUX terminal by compiling the code, along with the header file, then executing it. Once the program is running it allows the user to input commands in the terminal, then executes the commands.

There are currently issues with our program, particularly a segmentation fault that we have been unable to diagnose successfully. The shell executes to the point that it requests user input, but upon parsing the command and copying it to a variable for use, I believe we attempt to access memory that is not properly allocated, but cannot prove this theory.

A basic workflow can be seen below:

