Jared Jones
UIN: 626001380
CSCE 313 – 508

PA 4 Report

In this programming assignment, a simple shell is designed in order to implement some of the basic features of Bash. This shell is able to continually prompt the user for input on a command line, while parsing the input based off of predetermined grammar, and when a user enters a correct command, it will execute it the same way a shell would with fork and exec. This shell should also allow for the use of input and output redirection, command piping, background process running, and have its own custom prompt that supports printing the current directory, user name, current date, and time.

This shell prompts the user for input with a custom prompt that can be changed using the command “togglelong”. The shell prompts the user with a prompt that consists of “user name – computer name : project working directory $”, but when the command togglelong is used the prompt is extended to include the current time and date inserted in between the computer name and directory. The toggle was created in order to provide the user with a shorter shell prompt because the full prompt is extremely long and almost wraps around the terminal once.

The shell implements its own split function separate from the provided starter code, that iterates through the string and breaks it up based on the passed in character separator, but this split function accounts for quotes by not splitting if it finds a character inside a string or char array that it has been instructed to split on.

The shell, after reading in the users single line command will split it up based on pipes and then will fork allowing the child process to handle the actual parsing and execution of the command. The command’s parts are each individually searched for the input and output redirection operators and the background operator (< , > , and & respectively) and then their output is redirected accordingly, or in the case of an &, the PID of the child process is added to a vector that keeps track of the current background tasks so that they can be checked every iteration of the shell using a non blocking implementation of the waitPID() command.

The shell handles pipes using a form of triple pipe. This triple pipe is really actually a single normal pipe with an extra integer that keeps track of the previous pipes output file descriptor so that the current process can take its input form the previous task without overriding the main functions input/ output file descriptors. At the beginning of the pipe loop, the first pipe is created and then on the first iteration only the standard output is redirected to the pipe. Then on the next iteration of the loop the next child process sets its input as the output of the previous pipe and then creates a new pipe. Since it is not the first and not the last pipe then the middle command executed will make its standard input be equal to the previous pipe value and direct its output to the input of the pipe again. This chain will continue until it reaches the final pipe statement where it will only redirect its input form the previous pipe output file directory and the output its sent to standard output where the user can see the product of the pipe.

Through the combination of these implementations, a basic shell system is able to be implemented that is able to fulfill the design requirements stated in the assignment description. Various comments are also available in the program that aid with the understanding of each section or portion of the program letting the user know what parts of the program function and accomplish the various tasks.