REPORT

**Name:** Chris Banci

**Date:** February 27, 2017

**Course:** CS433 - Operating Systems

**Assignment:** 2 - Unix Shell & History Feature

-----------------

**Description:**

-----------------

This program is an implementation of a shell which allows the user to enter shell commands and execute them. If the command entered does not include an ampersand symbol at the end, the parent process waits until the child process completes its operation before resuming to its own operation.

The program also has a command history feature, which allows the user to select previous entered commands. Entering *“!!”* selects the most recent command from the command history list and entering *“!n”* selects the nth command from the command history list. The selected command is echoed to the screen and also saved into the history command list.

----------------------

**Implementation:**

----------------------

*To execute a command:*

* The program takes the entered command into a buffer and parses it. The parsing function tokenizes the buffer into separate arguments and checks if the last arguments contain an ampersand. Lastly, the argument list is terminated with a *NULL*.
* Next, the arguments are checked if it is a user command or shell command. If the function is a user command the arguments are passed into a function called execUserCommand. If the function is a shell command, the arguments are passed into a function called exec ShellCommand.
* If a shell command, a child process is created by forking the parent process. The arguments are then passed into the function prototype *execvp(args[0], args)* which executes the program.
* If the command did not contain an ampersand, the parent process waits until the child completes its operation before resuming.

*To execute a user command, type:*

* exit // exits the program.
* history // displays a command history list.
* !! // selects the most recent command from the command history list.
* !n // selects the nth command in the command history lists.

----------------------

**Design Choices**

----------------------

For this assignment, I wanted to simplify the design to make it readable in the main function.

I was able to achieve this by using C++ with a class and header instead of C to organize my code. This allowed me to break down the process into 3 simple steps: 1) get command, 2) parse command, 3) execute command.

For saving the command into the history command list, I chose to use the data structure vector as it is easy to insert new elements inside. The commands are saved as a string type partly due to me not knowing how to save char pointer arrays inside a vector. To execute those string type commands, they would be reparsed into arguments.

--------------------

**Included Files:**

--------------------

Source:

* Main.cpp // contains the driver for the assignment
* Shell.cpp // contains the implementation of shell

Headers:

* Shell.h // contains the header for shell.

Other:

* Makefile // used to compile the source files
* output // the executable to run the program

----------------

**How to run:**

----------------

To run this program, use the Makefile which will compile the source files and create an executable called output.