Project 4 Report

by Josh Moore, Maria colon, Theint New Nyein, Leo Goldstein

Source:

For project 4, the implemented the ‘source’ command, which is used to execute shell commands within another file, was approached by having the child process execute the read commands from the sourced file and outputting the results of the child process in the parent process. Specifically, used the popen() function. The hardest part was testing the program. A testing script file to see if it works however, we notice that we were trying to compile a script file with a cpp compiler as if it were a cpp program, leading to compilation errors. Thus, to figure out how to compile the script file separately from the main program, we found a way to use bash, which is a shell interpreter, to execute the script file separately from the compilation of the cpp program.

Shift:

In comparison to the other commands with needed to handle, this command was by far the easiest to implement. The command, relatively speaking, only moved the arguments given when using the source command by the number specified. Because of this, a simple string manipulation was performed by removing the number of arguments based on the input given. The result would be retuned and then used for the next set of commands.

Sub-System:

The sub-system was, as the shift command was, relative easy to incorporate into the program. The simplest method that was used was to have the command be executed using the forking method, thus any data performed within this child would not be saved to the parent. This did, however, cause an error as the previous project would have the commands issued already in a child processes. A simple fix was incorporated into the main, determining first if the command inputted was the sub-system command.

Code

```
Tesla: [~/SP24/Operating_Systems/Project_4] $ make
mkdir object_files
gcc -c implement_files/command_alias.cpp -o object_files/command_alias.o
gcc -c implement_files/command_cd.cpp -o object_files/command_cd.o
gcc -c implement_files/command_help.cpp -o object_files/command_help.o
gcc -c implement_files/runnable.cpp -o object_files/runnable.o
gcc -c implement_files/special_characters.cpp -o object_files/special_characters.o
gcc -c implement_files/placholder_file.cpp -o object_files/placholder_file.o
gcc -c implement_files/command_history.cpp -o object_files/command_history.o
gcc -c implement_files/command_set.cpp -o object_files/command_set.o
gcc -c implement_files/history_manager.cpp -o object_files/history_manager.o
gcc -c implement_files/command_source.cpp -o object_files/command_source.o
gcc -c implement_files/command_shift.cpp -o object_files/command_shift.o
gcc -c implement_files/util.cpp -o object_files/util.o
g++ main.cpp -o gamma ./object_files/command_alias.o ./object_files/command_cd.o
./object_files/command_help.o ./object_files/runnable.o ./object_files/special_characters.o
./object_files/placholder_file.o ./object_files/command_history.o ./object_files/command_set.o
./object_files/history_manager.o ./object_files/command_source.o ./object_files/command_shift.o
./object_files/util.o
Tesla: [~/SP24/Operating_Systems/Project_4] $ make run
./gamma
Entering interactive mode...
/home/josmoor/SP24/Operating_Systems/Project_4 $ source data/testing_script.sh first_arg
second_arg
Hello, this is a testing script.
This script will print the current date and time:
Thu 25 Apr 2024 10:25:14 PM EDT
First argument: first_arg
Second argument: second_arg
Number of provided args is 2
/home/josmoor/SP24/Operating_Systems/Project_4 $ ( ls -ld .?*; )
ls: cannot access '.?*': No such file or directory
/home/josmoor/SP24/Operating_Systems/Project_4 $ ( ls -ld; )
drwxr-xr-x 9 josmoor domain users 4096 Apr 25 22:22 .
/home/josmoor/SP24/Operating_Systems/Project_4 $ exit
```