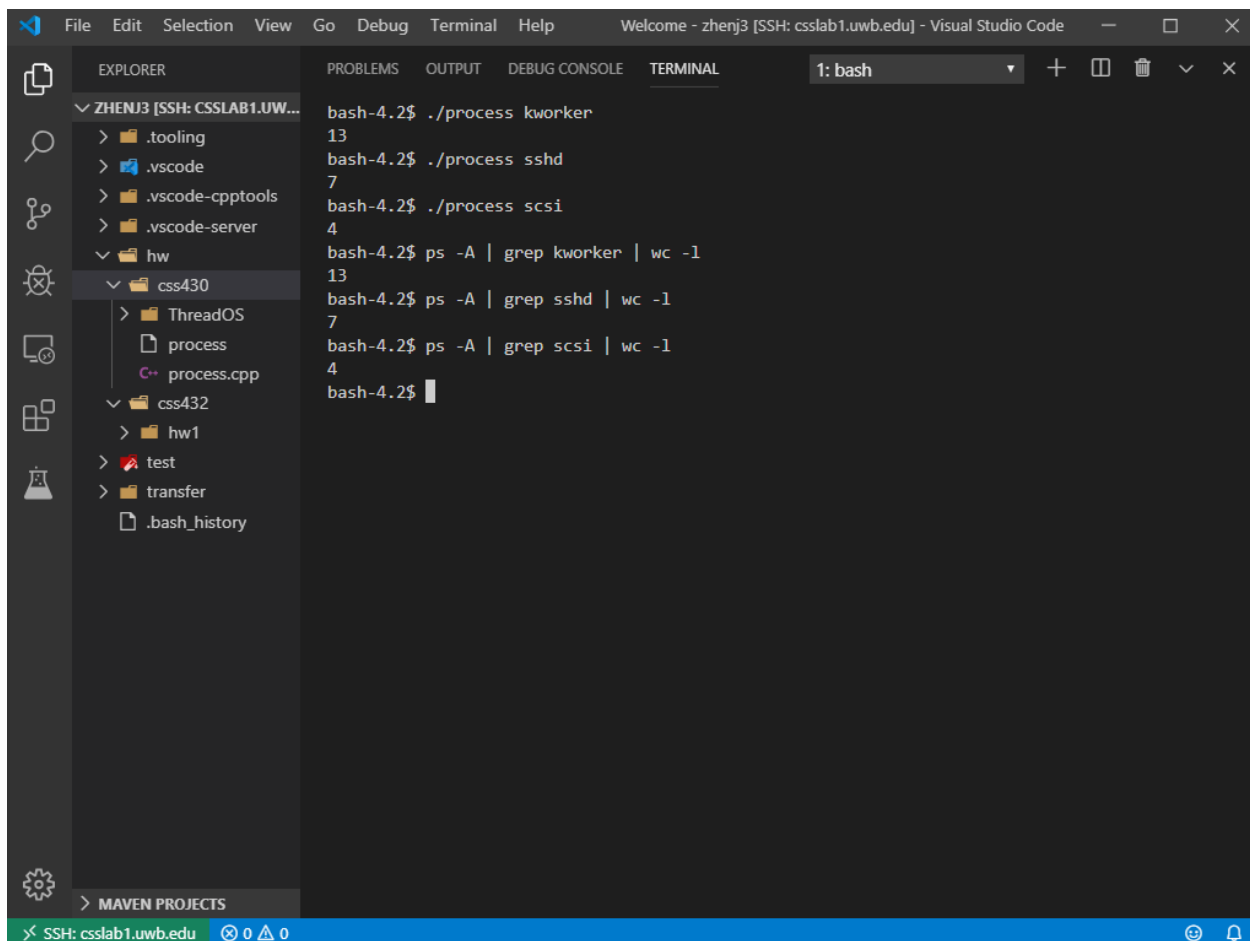


JUN ZHEN – Assignment 1 - CSS430

Report

Part 1: Linux System Programming

Process.cpp is a program that take in a process name as argument and the amount of that process is currently running. begins by initialing two separate pipes, one pipe for the grand child and the child, the other pipe is for parent and child. Next the parent will fork a new process and wait. When the child process spawns, it too will fork another child (grandchild) and wait. The grand child will redirect the standard output to the write side of pipe 2. Then it will call `execlp ("ps")` and close unused pipes. After the grandchild finish running, child will redirect its standard input to the read side of pipe 2 and also redirect its standard output to write side of pipe 1. It will then call `execlp ("grep")` close off all unused pipes. After child finish processing, the parent will redirect its standard input to the read side of pipe 1. Once the parent has the hold of the data from grandchild to child, it will call `execl (wc)`. The output will come out to the standard output, which will be the terminal.



```
File Edit Selection View Go Debug Terminal Help Welcome - zhenj3 [SSH: csslab1.uwb.edu] - Visual Studio Code
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
ZHENJ3 [SSH: CSSLAB1.UW...
  .tooling
  .vscode
  .vscode-cpptools
  .vscode-server
  hw
  css430
    ThreadOS
    process
    process.cpp
  css432
    hw1
  test
  transfer
  .bash_history
MAVEN PROJECTS
bash-4.2$ ./process kworker
13
bash-4.2$ ./process sshd
7
bash-4.2$ ./process scsi
4
bash-4.2$ ps -A | grep kworker | wc -l
13
bash-4.2$ ps -A | grep sshd | wc -l
7
bash-4.2$ ps -A | grep scsi | wc -l
4
bash-4.2$
```

Part 2: ThreadOS Shell Design

The Shell is a program that runs other programs by spawning new threads within the process. It begins by waiting for user input. Then it will begin parsing the data by the user. If there appear an ampersand, it will spawn a new thread, running concurrently with all the threads. If there is a semicolon then no threads will begin before the current thread finishes. This part will be handled by two for-each loop that will parse the data. By using the string method `split()`, the data split apart separating them from each other. This continues until the data has no more special character it will then begin spawning new threads. Once we have our commands, we put them back into an array. The `SysLib.exec()` will take the array and initialize a new thread and execute the arguments. A final while loop at end to wait for all the threads to finish running before the program is exit.

[illegible]