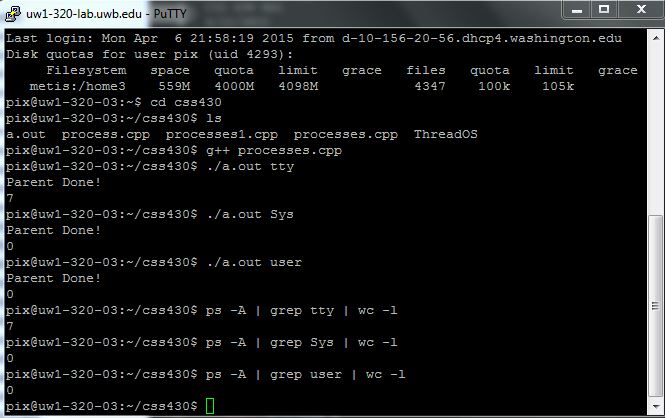
Xueting Pi

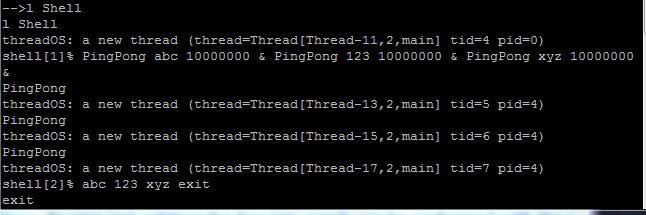
CSS 430 – Assignment 1

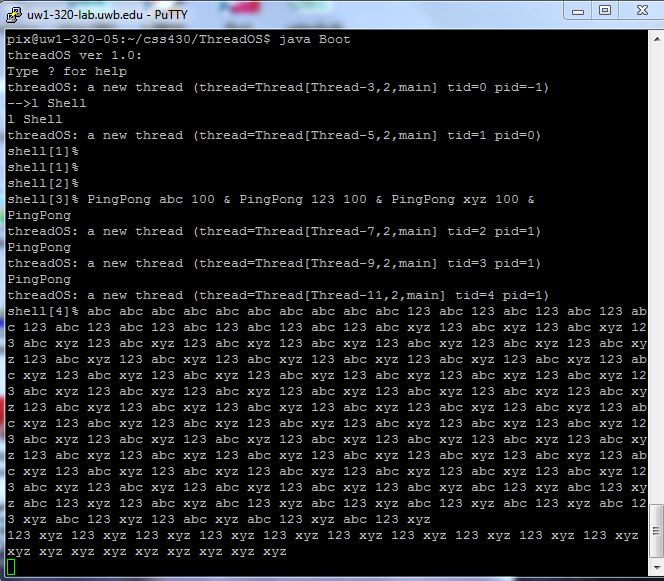
4/15/2015

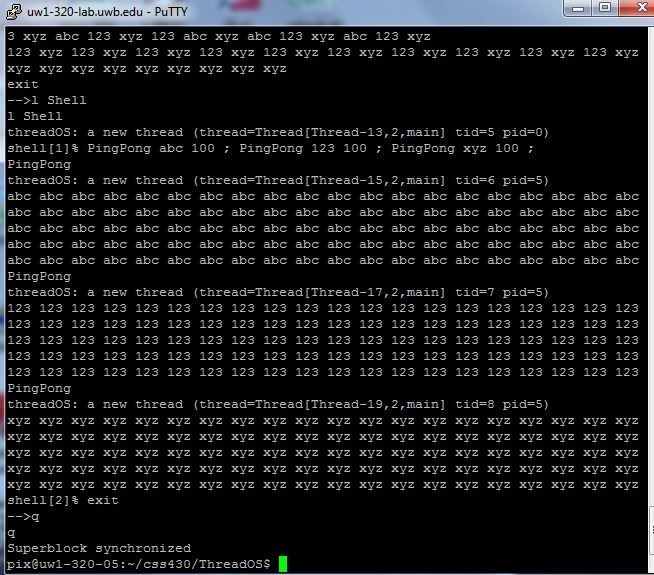
Part 1



Part 2







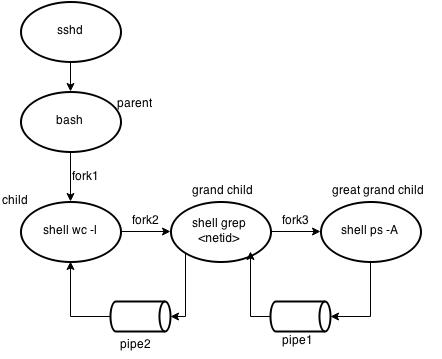
Report

Processes.cpp

Flow chart

As my flow chart shows, the pipeline goes through 3 forks and two pipes. I created two fd for two pipes and handled the opening and closing of them for each forked process respectively. At the beginning, I checked the number of arguments, if less than 2, then not enough argument is provided. I also created the two pipes at the beginning, if either is less than 0, then it would output pipe error, which means the pipes are not created successfully.

I used switch to do three forks, for each fork, if the output is less than 0, it outputs fork error. Otherwise, based on whether it’s a child process or parent process, I controlled the pipe accesses and rewrite the duplicate process images. It’s pretty clear from the flow chart that for child “wc –l”, it closes pipe1’s read and write, and closes pipe2’s write, only allowing writing to pipe2. For grandchild “grep argv[1]”, it closes pipe2’s read and pipe1’s write, and opens pipe2’s write and pipe1’s read. For great grandchild “ps –A”, it only allows to write to pipe1. Note that before parent process waits, we need to close all pipes, but in my case, since I closed every pipe access right after each dup2 call, so I didn’t do it in my parent process.



Shell.java

Testing:

|  |  |
| --- | --- |
| Testing case | result |
| PingPong abc 100 & PingPong xyz 100 & PingPong 123 100 & | Three threads are running concurrently |
| PingPong abc 100 ; PingPong xyz 100 ; PingPong 123 100 ; | Threads are running sequentially |
| PingPong abc 10^7 & PingPong xyz 10^7 & PingPong 123 10^7 & | Each argument for PingPong only outputs once |
| PingPong abc 100 & PingPong xyz 100 & PingPong 123 100 | Even with no &/; in the end, it outputs correctly |
| exit | Get out of Shell and back to ThreadOS |

The way I implemented Shell.java is that in the run method, there is a while loop that runs forever until user types exit, which takes you out of Shell and back to ThreadOS. In the loop, first of all, I prompt user to type commands and read their input stream into a String cmd. Then I called SysLib.stringToArgs(cmd) to handle the messy input and store them into an array with all the arguments.

I created a List<String> to repeatedly adding arguments unless hit “&” or “;” or “exit”. After that, I looped over each element in the array. If args[i] is equal to “exit”, call SysLib.exit() and break out of the loop. If args[i] is equal to “&”, call execution function, which takes in the list containing all the arguments before “&” and convert to String[] to pass in SysLib.exec(). Clear the list after this call. Similar for “;”, except for that after calling execution, I added the call SysLib.join() to ensure next process runs after the termination of previous one. If args[i] is none of the above cases, add it to the list. To handle commands that do not end with “&” or “;”, I executed the rest arguments in the list if the list is not empty.