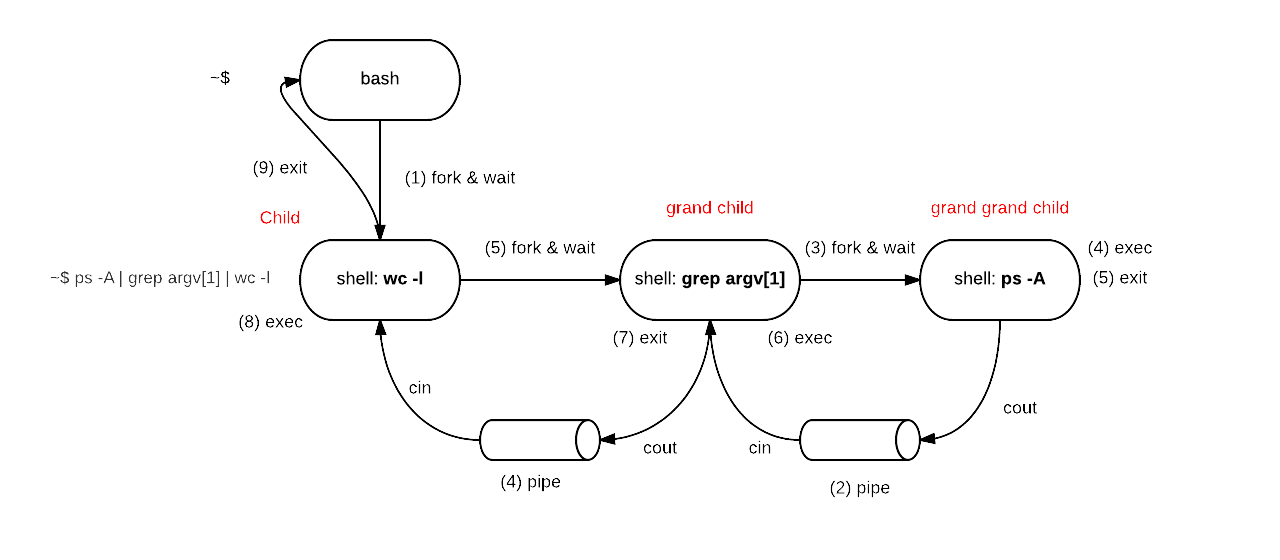
**Report**

1. Algorithm for Processes.cpp

For Processes class which performs same functions as command “ps –A | grep argv[1] | wc -l”, I split individual task as a child of it parent – fork. Being a parent, the fork has to wait for child task to be done. Between each fork, pipes are used to take in child output and deliver to parent connect them. After each child fork complete jobs, returns back to bash and exit.

1. Flowchart for Processes.cpp



1. Test output for Processes.cpp

D-10-156-21-245:processes jingjingdong$ g++ processes.cpp

D-10-156-21-245:processes jingjingdong$ ./a.out tty

9

D-10-156-21-245:processes jingjingdong$ ps -A | grep tty | wc -l

8

D-10-156-21-245:processes jingjingdong$ ./a.out Sys

92

D-10-156-21-245:processes jingjingdong$ ps -A | grep Sys | wc -l

91

D-10-156-21-245:processes jingjingdong$ ./a.out user

3

D-10-156-21-245:processes jingjingdong$ ps -A | grep user | wc -l

2

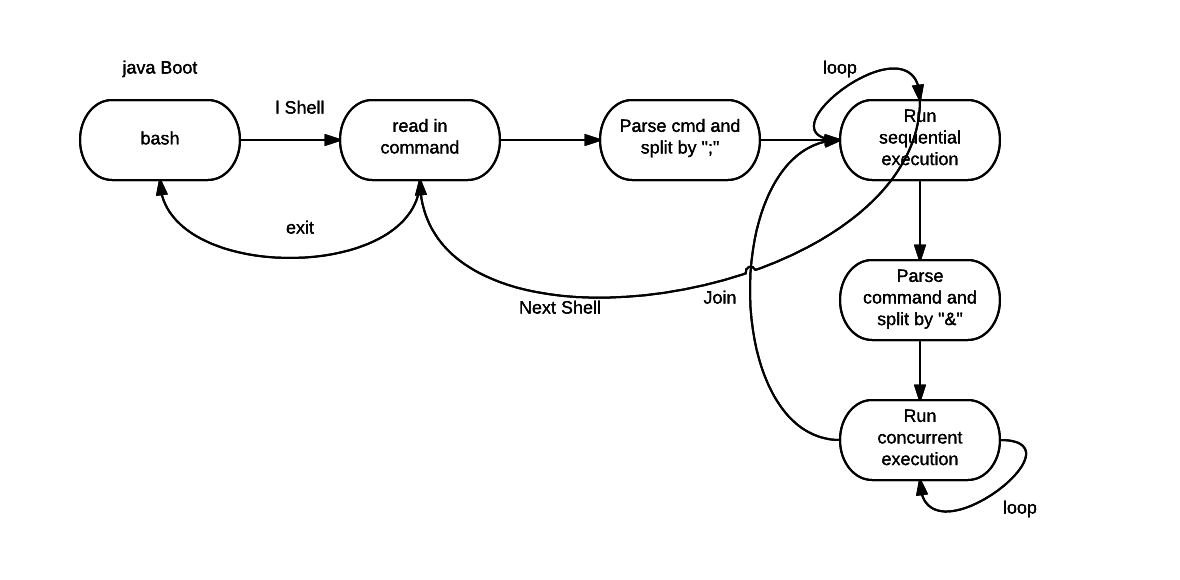
1. Algorithm for Thread OS – Shell

Shell class takes in command and makes concurrent and/or sequential executions.

First, read in command line and then parse and split the line by “;”. Now, perform sequential execution for each command. Before SysLib.exec, split each sequential execution by “&”. After concurrent execution, join the sequential execution. Loops over the same process until reach the last command.

Exit Shell when typing in “exit”.

1. Flowchart for Shell.java



1. Test output for Shell.java

I tested my Shell.java with different cases:

Case 1: empty command

Case 2: one command

Case 3: multiple concurrent commands

Case 4: multiple sequential commands

Case 5: multiple concurrent and sequential commands

Case 6: exit Shell

