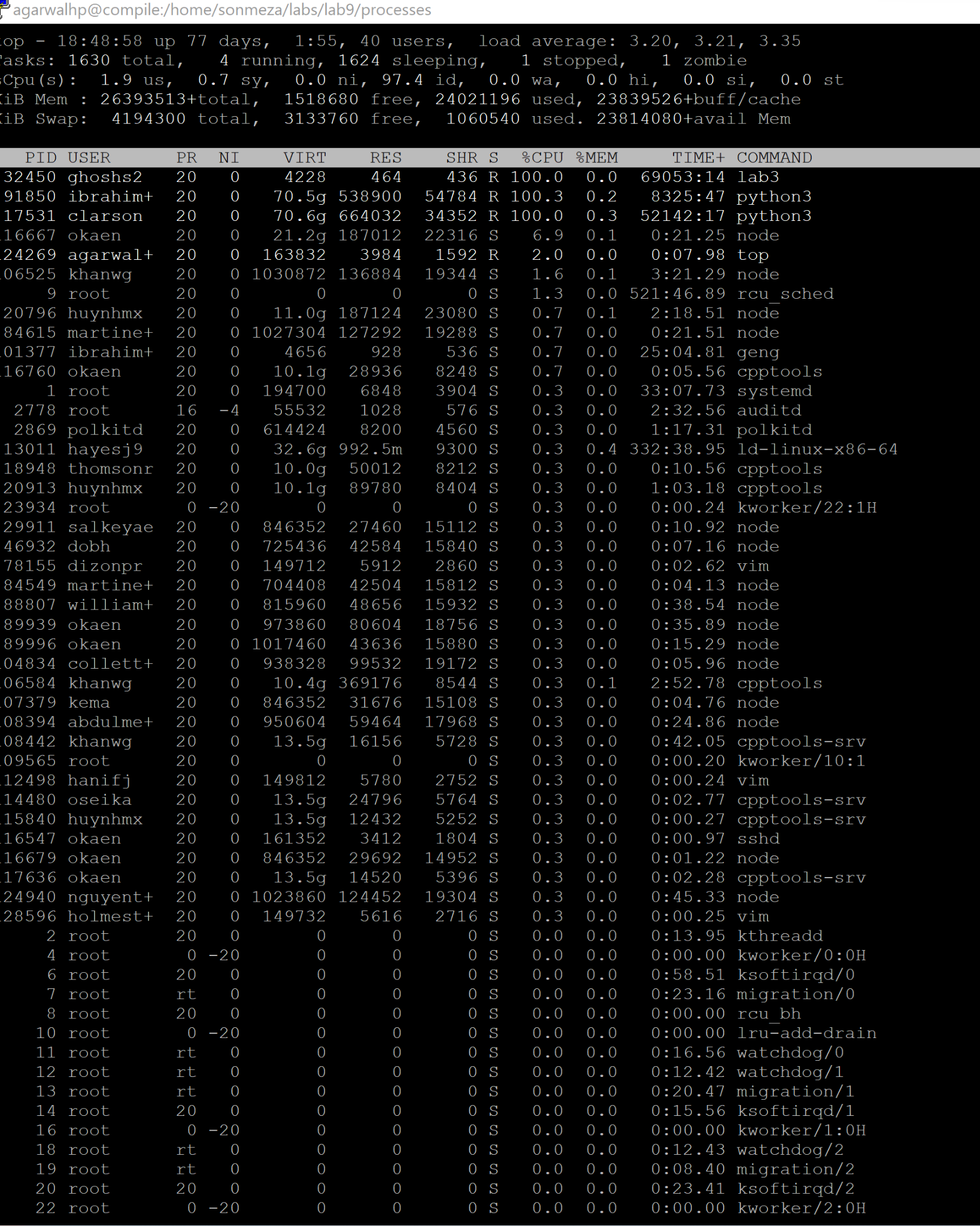
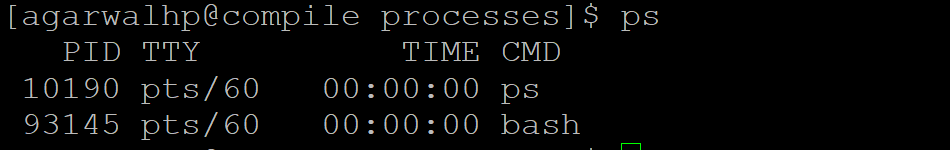
* 1. In the terminal type top, what do you see, describe shortly? (use man pages to understand)

Ans1: It shows you all of the processes and commands but live on which the user is currently processing. 

In the terminal type ps, what do you see, describe shortly? (use man pages to understand,)

Ans1:

ps also shows the checking for the processors like ps and bash.



Now compare and contrast top and ps. What is the difference between top and ps? (Hint: Which one is dynamic, which one is static?)

Ans1: top allows you display of process statistics continuously until stopped vs. ps which gives you a single snapshot. Top is dynamic and the ps is static.

Q2. In the terminal type ps -axu, what do you see, describe? (use man pages to understand, do not copy and paste just explain in a few words)

Ps -aux shows the processes from all users, owner of the process, and shows the processes by others as that's what aux stands for.

Q3.Modify this program and print the parent process id in addition to current process id (Look at the lecture slides). Which function returns parent process id?

/\* GettingStarted/getpid.c \*/

#include <stdio.h>

#include <unistd.h>

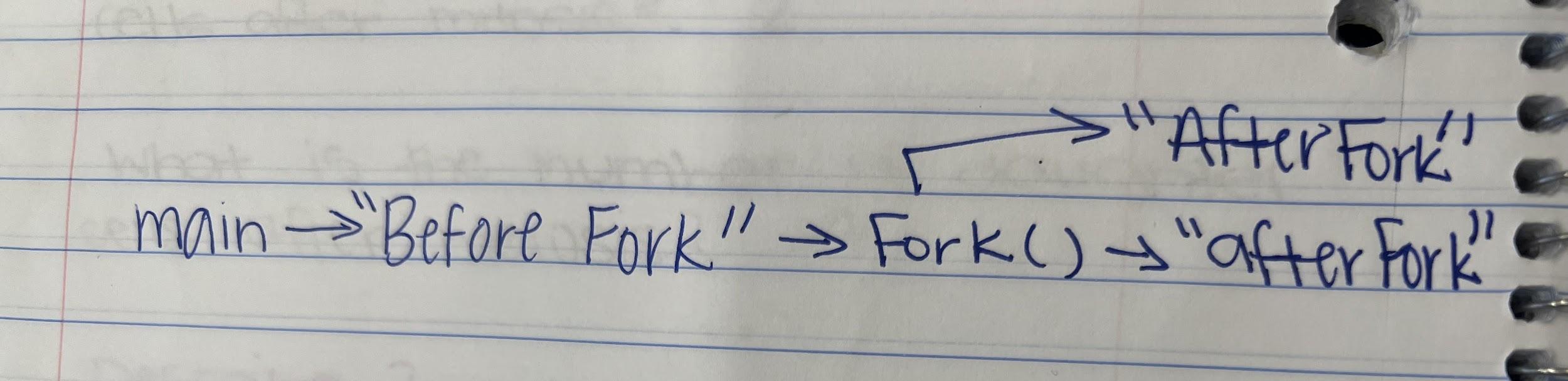
int main(int argc,char \*argv[])

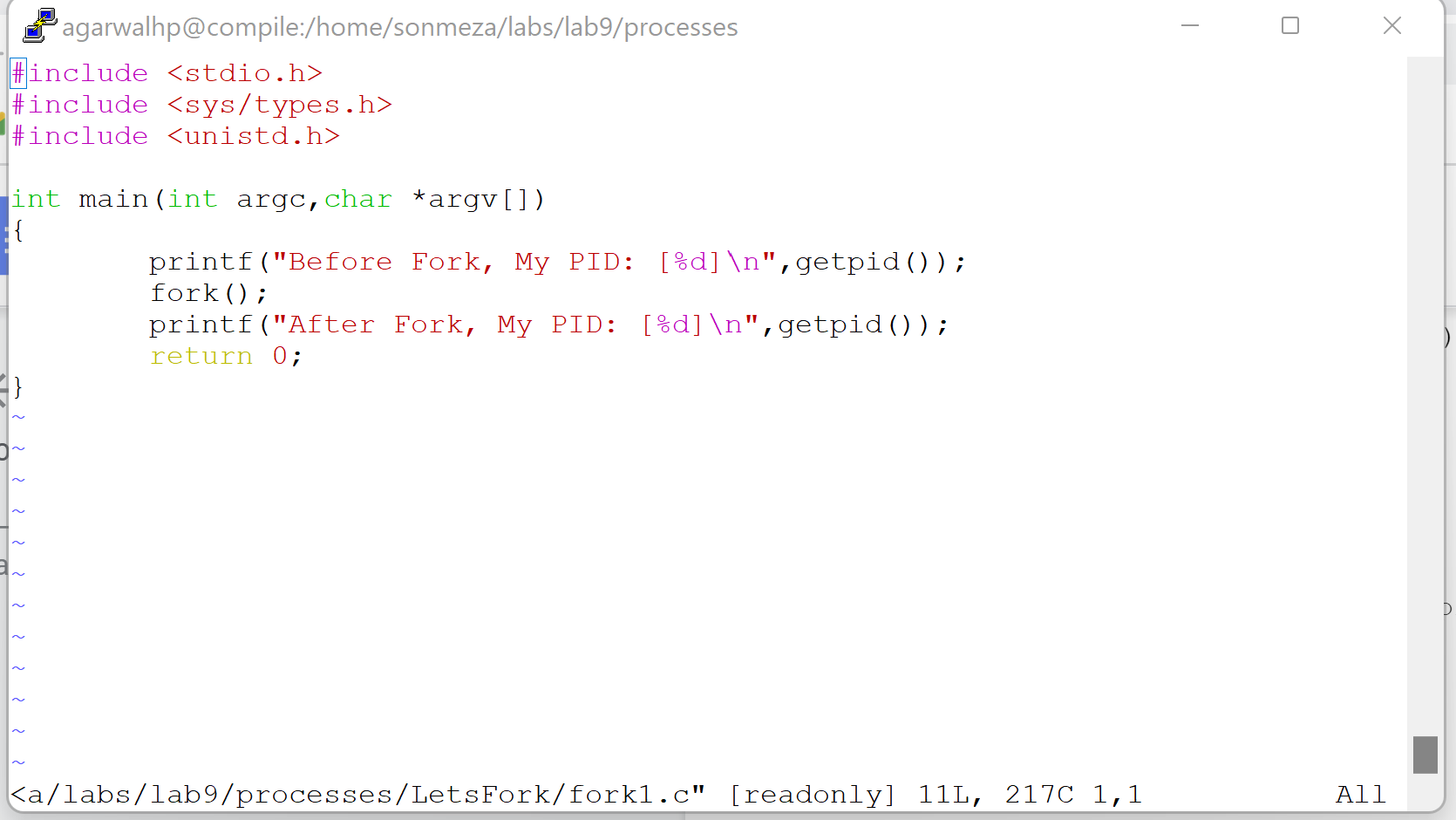
{

printf("I'am the parent with PID %d and PPID %d.\n",getpid(),getppid());

}

Q4. Draw process graph.



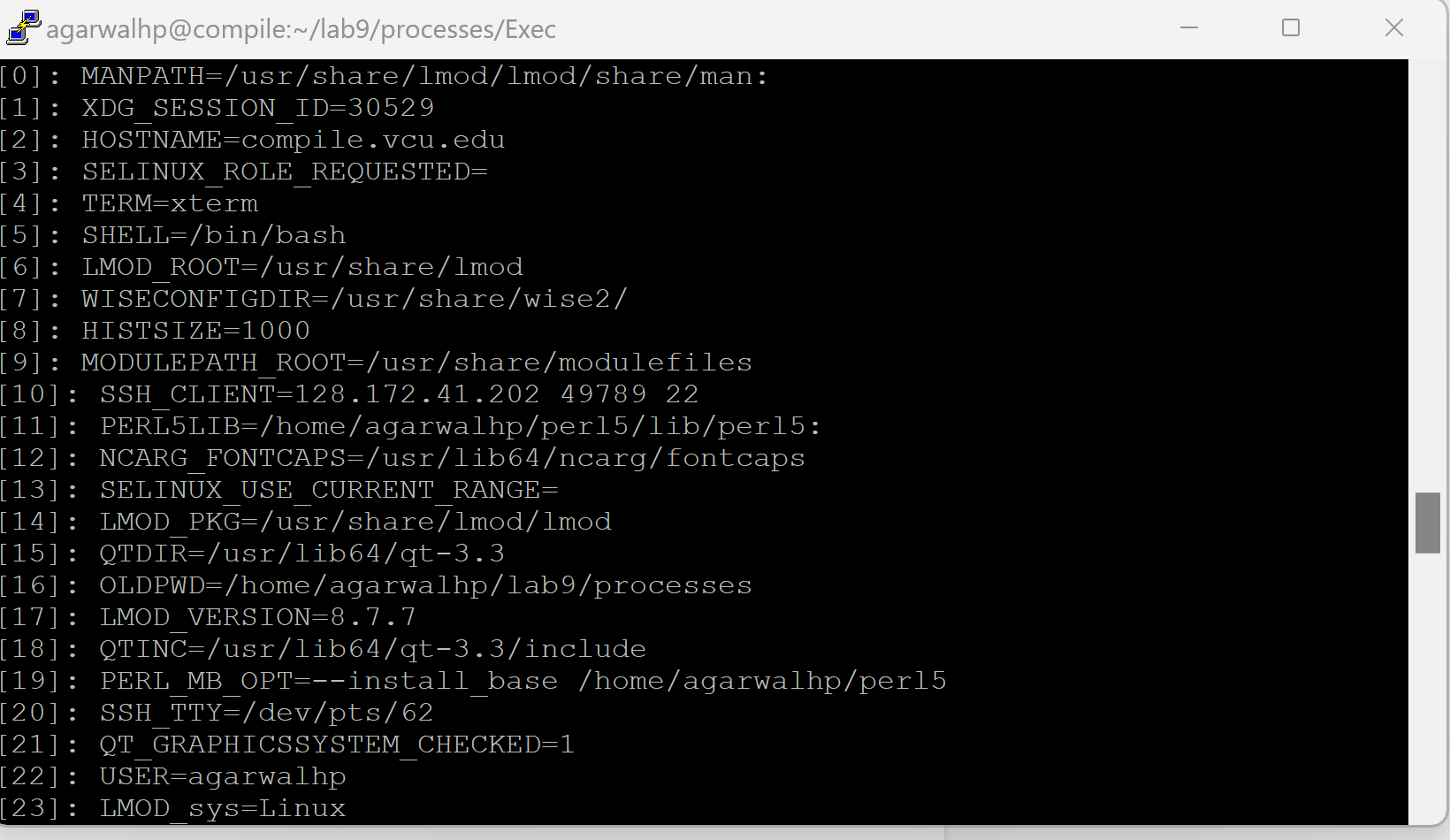


Q5. Look at the program below. It has several fork() and printf() calls. How many A, B and C will be printed? (Use process graph, then run the program)

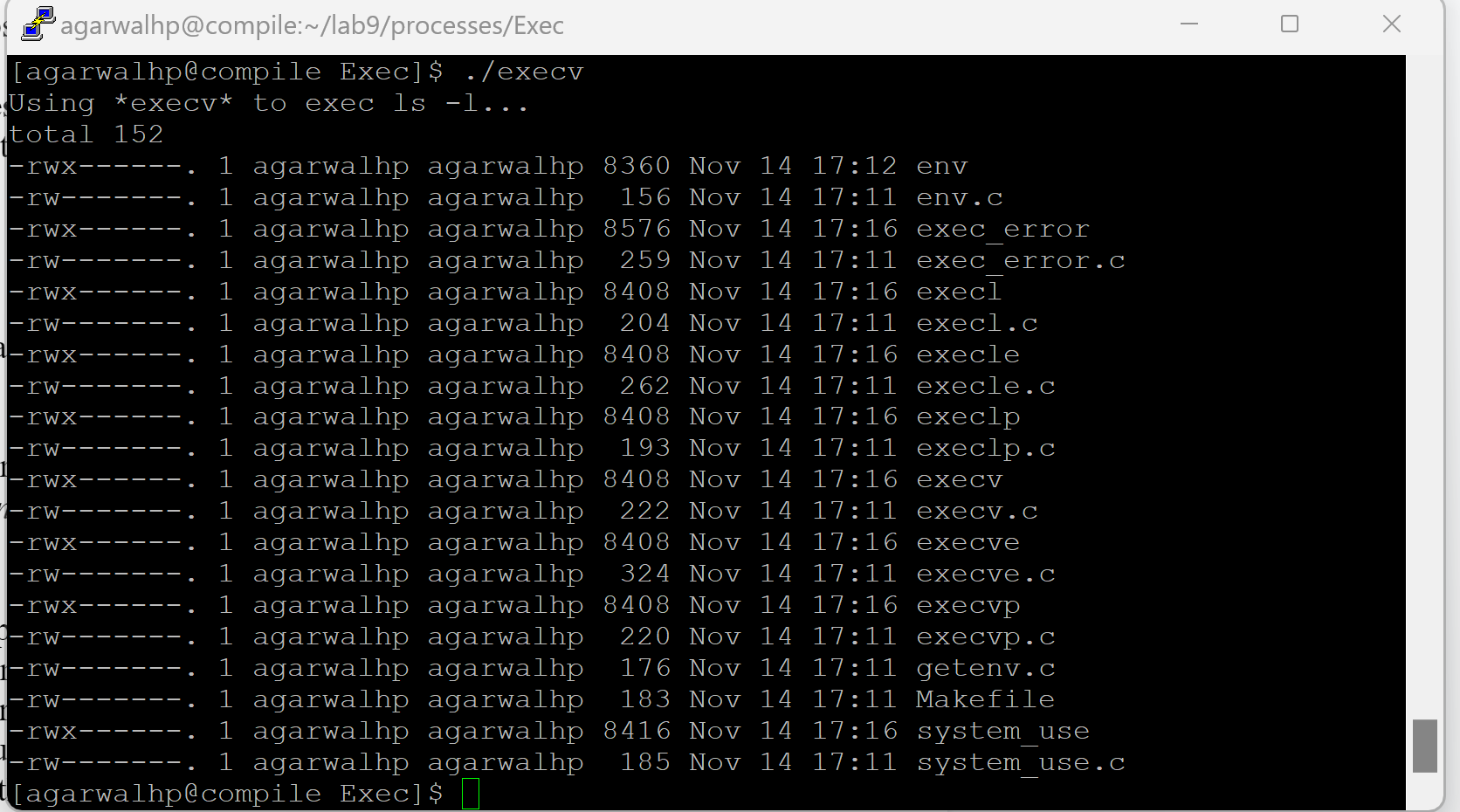
A: 1 time, B: 2 times, C: 4 times

Q6. Get a screenshot of environment variables (first 10 lines), get a screenshot and paste it under Q6. It should look similar to the screenshot below.

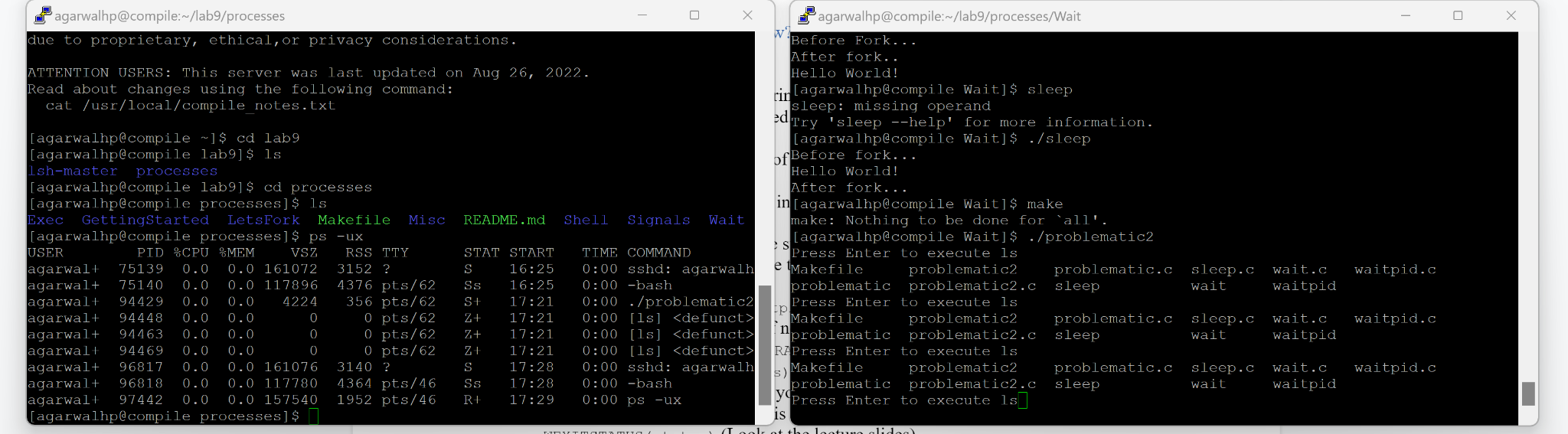
Gcc -env.c -o env



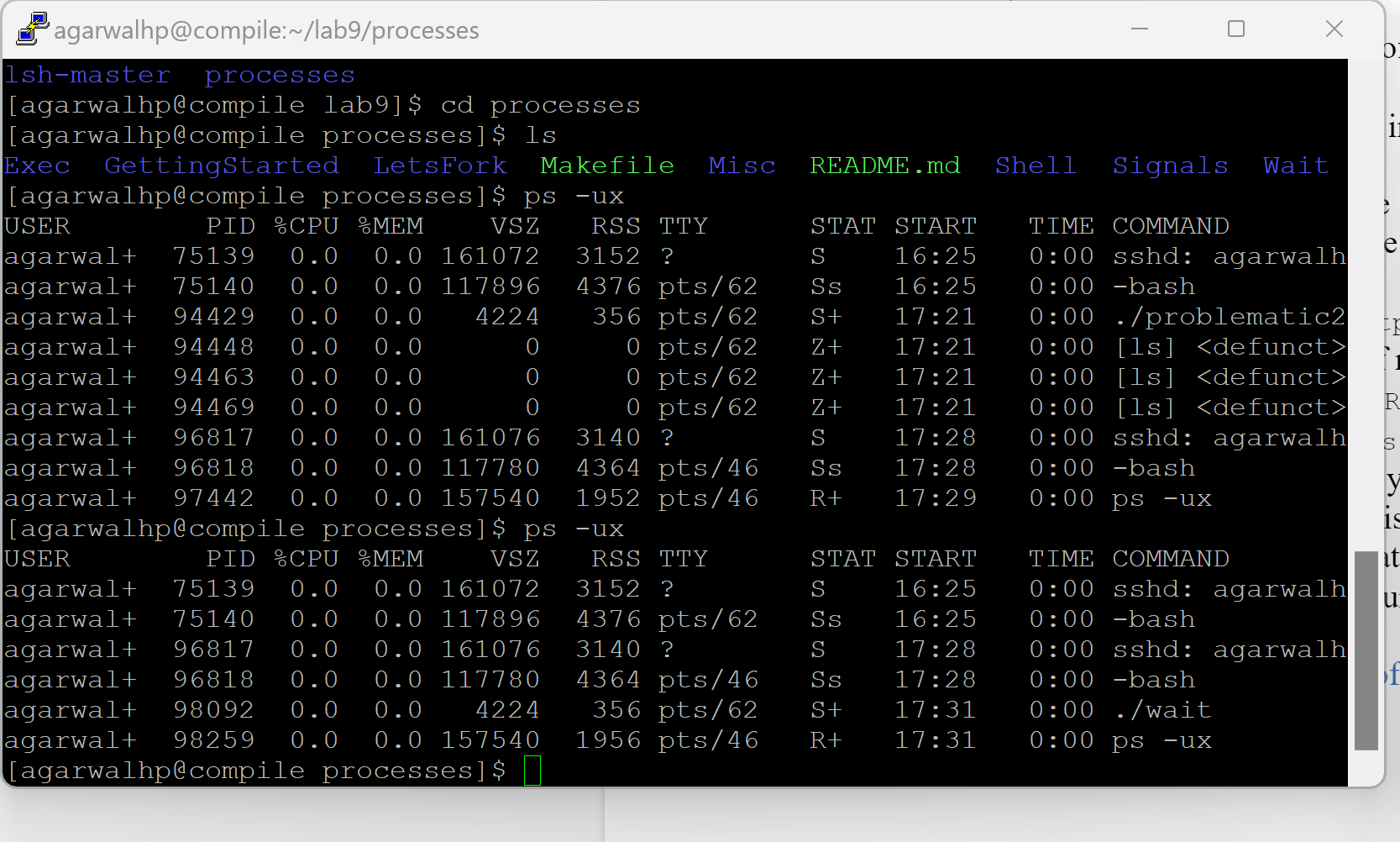
Q7. Get a screenshot of running .execv and paste it under Q7



Q8. Get a screenshot as the sample I attached above, and attach it to the word document under Q8. You should see 3 defunct processes, indicating zombies.



Q9. Do you see any zombies now?



No zombies

Q10. What is the difference of wait and waitpid in your own words?

Wait usually takes only one parameter and stores the status of the child process and blocks the call until the child process doesn’t terminate. Whereas waitpid is a more modernized version of wait where it takes 3 parameters and doesn’t wait for the child that terminates first; it has options that can control the process it waits for.

TEXTBOOK ANSWER: wait() takes one parameter for storing the status of the child process. waitpid() takes 3 parameters. The *first* one is child pid, therefore this is more specific than the previous wait().

The second one stores the status of the child. Whenever the child terminates/suspends, it will store the status code in the middle parameter. Note that it accepts a **pointer** only.

The third is the options for waitpid(). You can change various behaviour, for example, returns immediately if no child has exited: WNOHANG; also returns if the child is stopped (eg: by Ctrl-Z): WUNTRACED