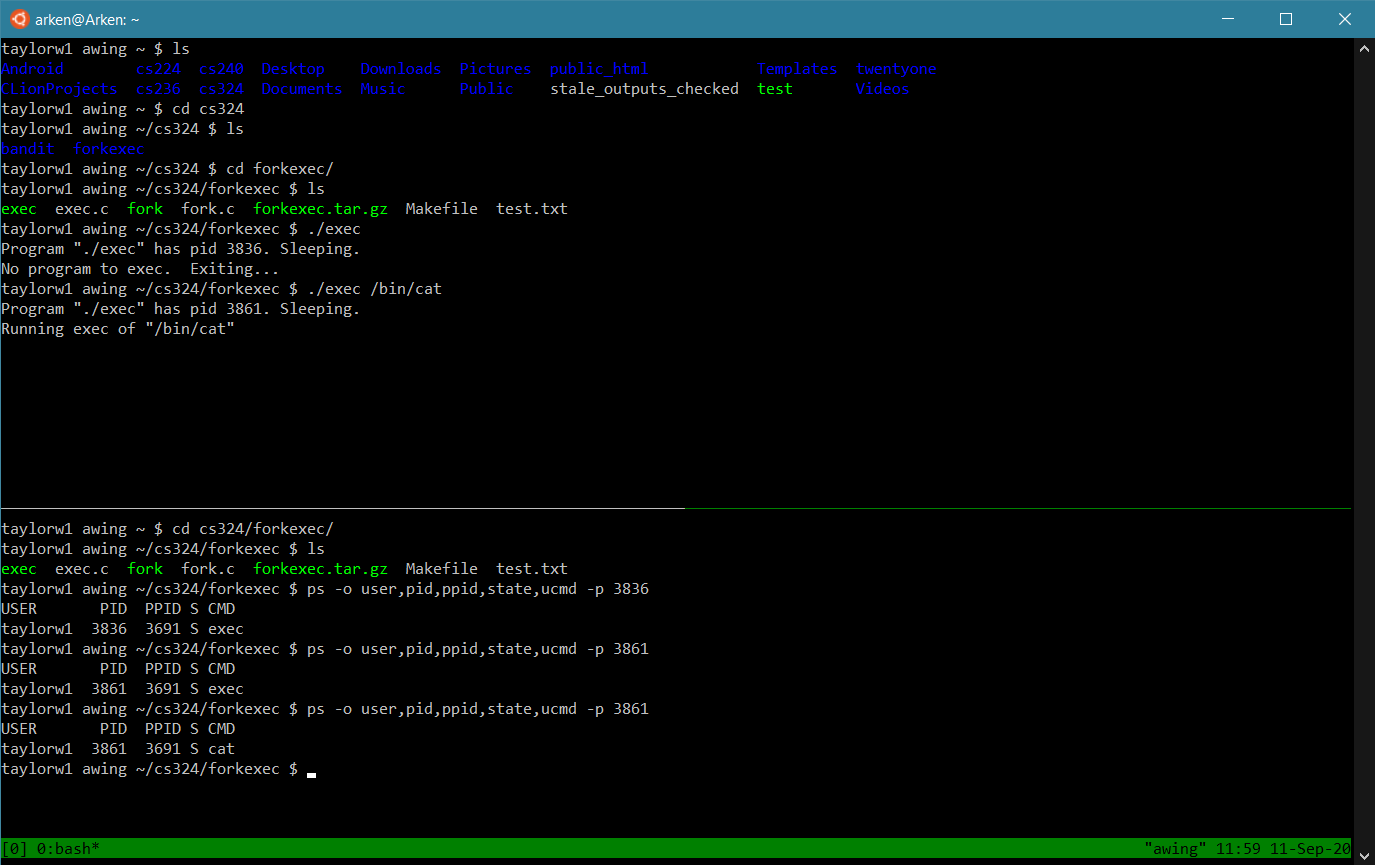
Forkexec

1. exec.c gets the pid, stalls for 30 seconds, then executes the program that was passed in as an argument.

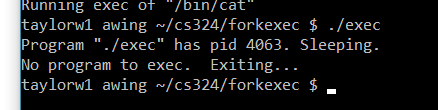
2. The final print statement will be executed when the other program returns/finishes.

3-4.



5. The pid is the exact same, but the command has changed. The reason being it’s still the same program but it is now executing a different command.

6.

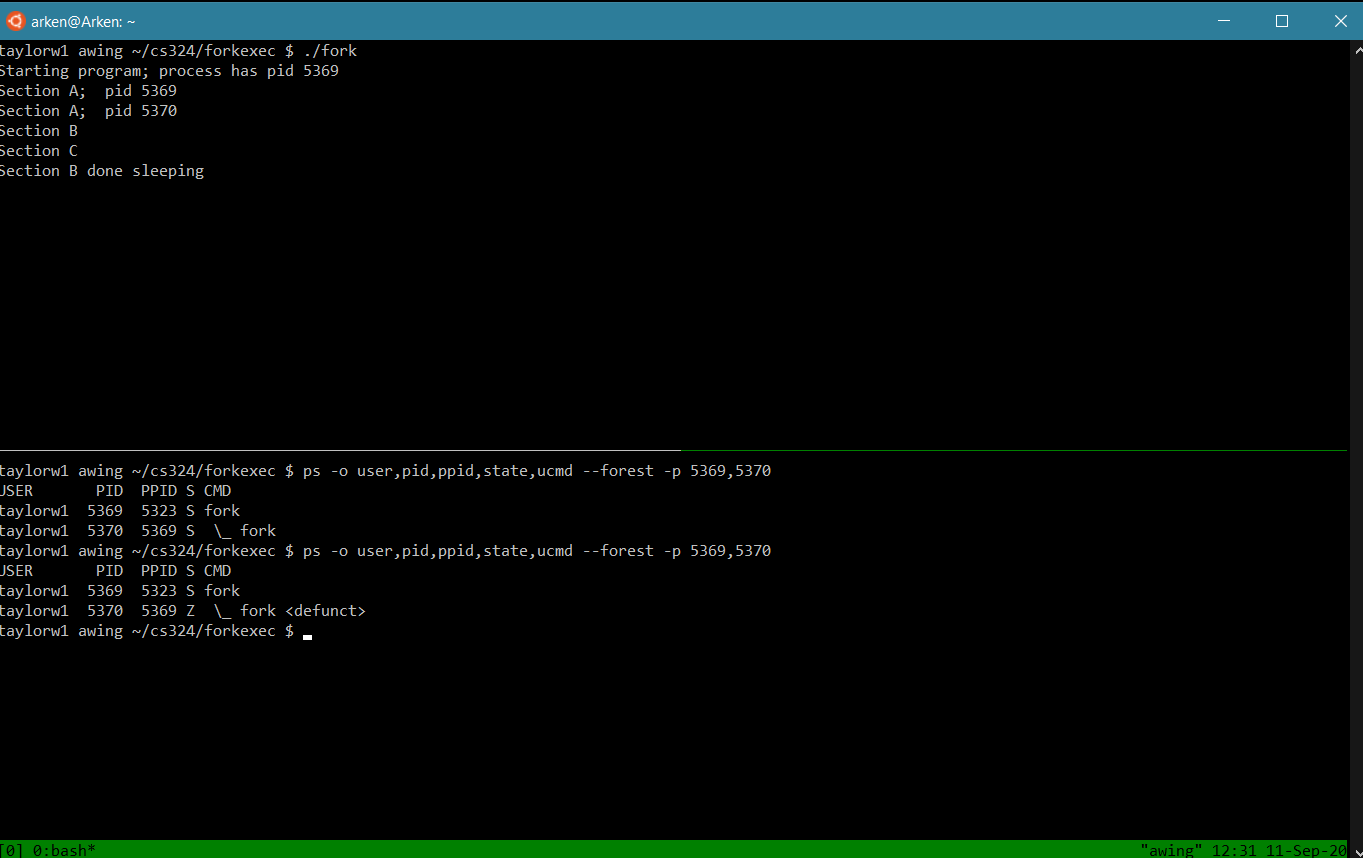


There weren’t any additional command line arguments, so argc was <= 1 and therefore the program exited.

7. fork.c forks and then prints which section it is in depending on the pid, sleeping between sections.

8. Sections A&D are run by both the parent and child processes, section B is only run by the child process, and section C is run only by the parent process.

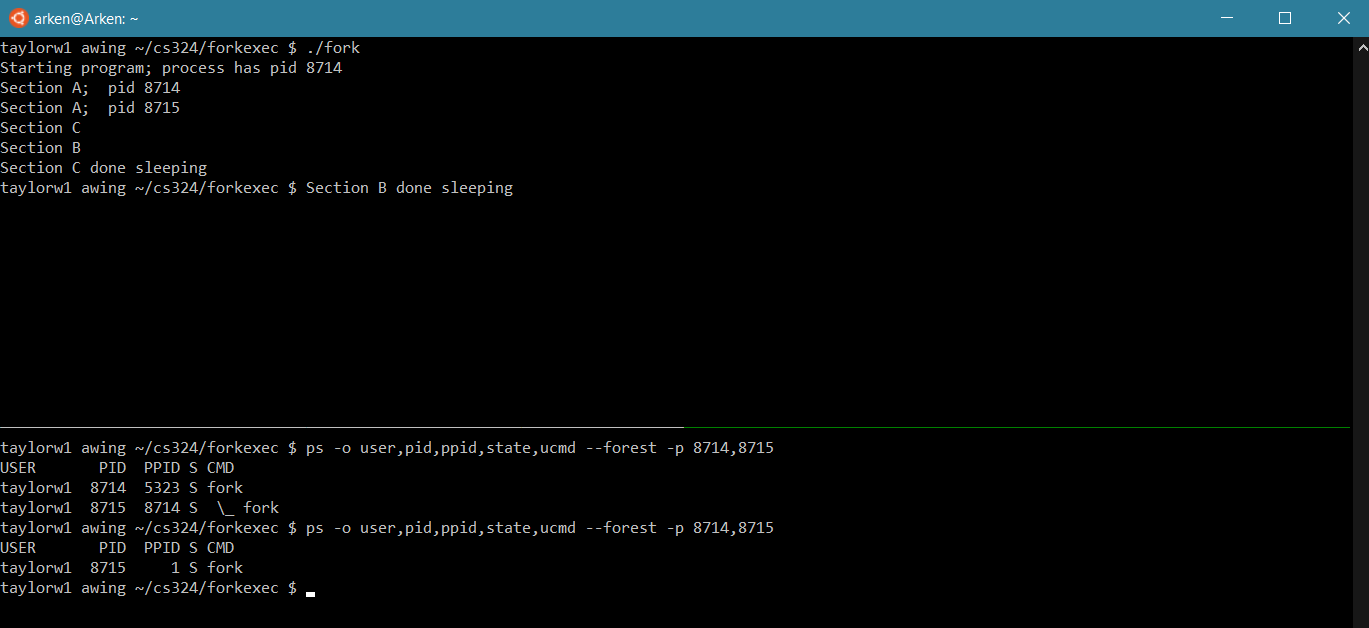
9.



10. Since fork creates another process, the pids are different. The states are the same except at the end when the child process became a zombie.

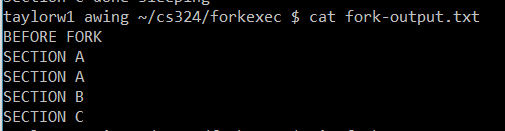
11. You can add either wait() or waitpid() if you have the specific child pid. I put waitpid() at the top of section C.

12.



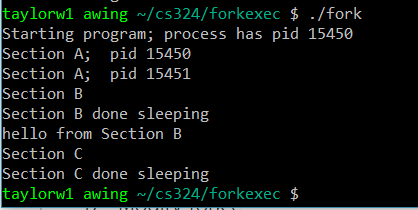
13. The biggest difference between the two ps commands is that the child process terminated after the program finished executing (i.e. after section c was done sleeping), and the reason for that is that the child process was reaped by the system.

14.

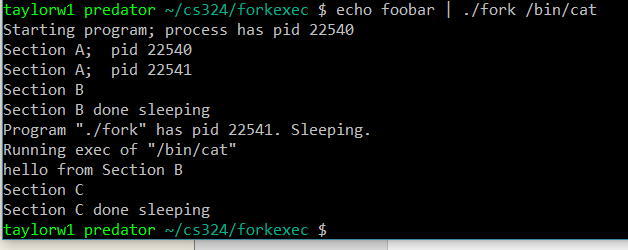


15. The parent/child processes execute in exactly the order that they should. There is no need for sleeps since the parent waits for the child to finish, and then continues executing as normal.

16.



17.



18.

