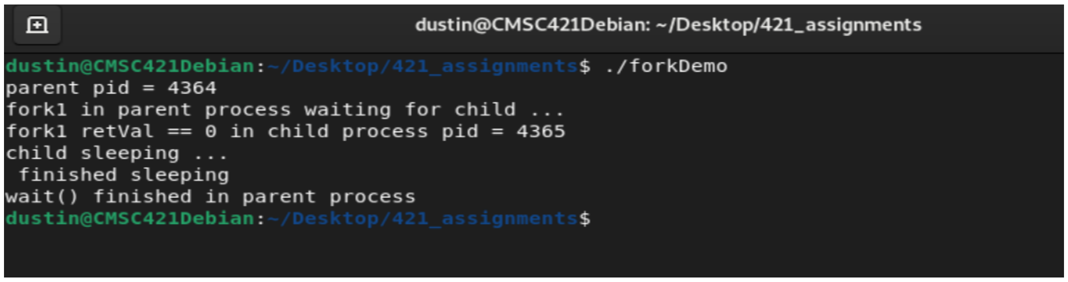
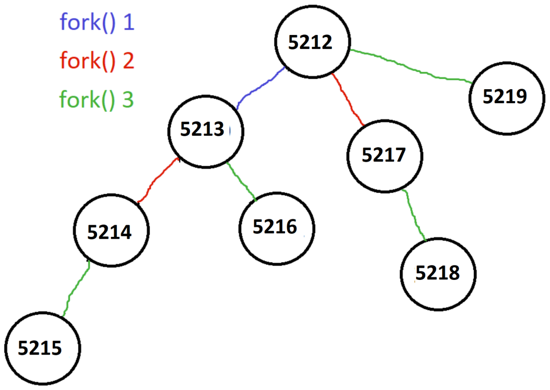
Dustin Mahmot
421 Fork Assignment

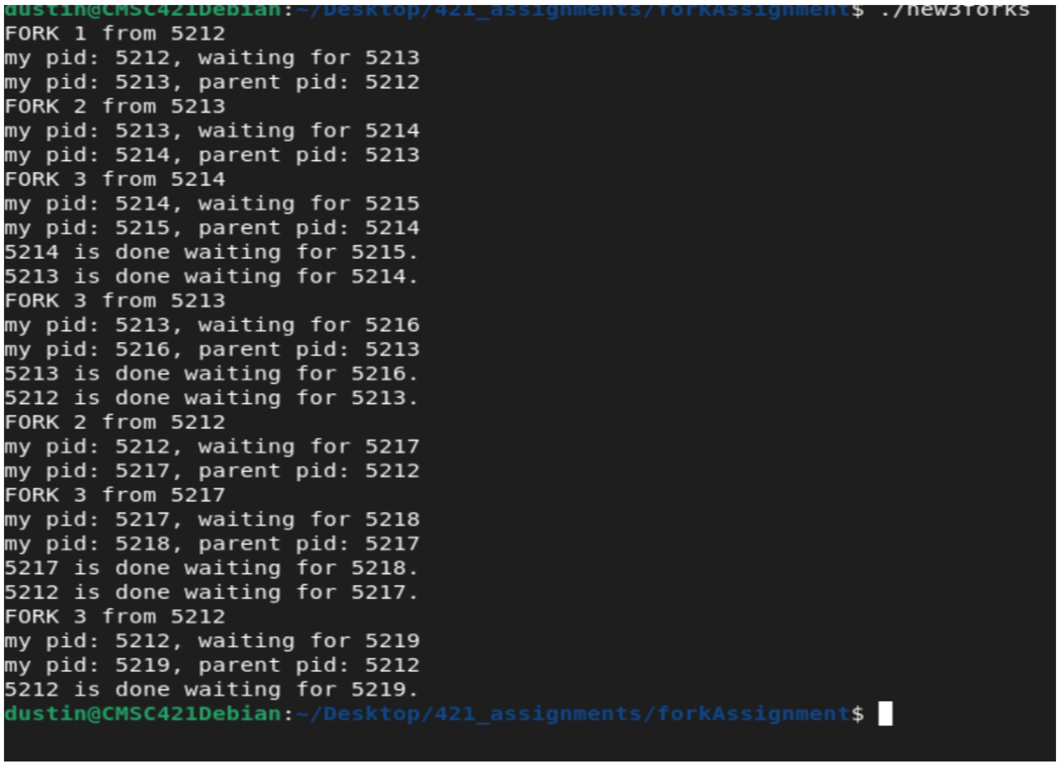
<u>Part 1</u>
Here's the demo output



```
dustin@CMSC421Debian:~/Desktop/421_assignments$ ./forkDemo
parent pid = 4364
fork1 in parent process waiting for child ...
fork1 retVal == 0 in child process pid = 4365
child sleeping ...
 finished sleeping
wait() finished in parent process
dustin@CMSC421Debian:~/Desktop/421_assignments$
```

<u>Part 2</u>

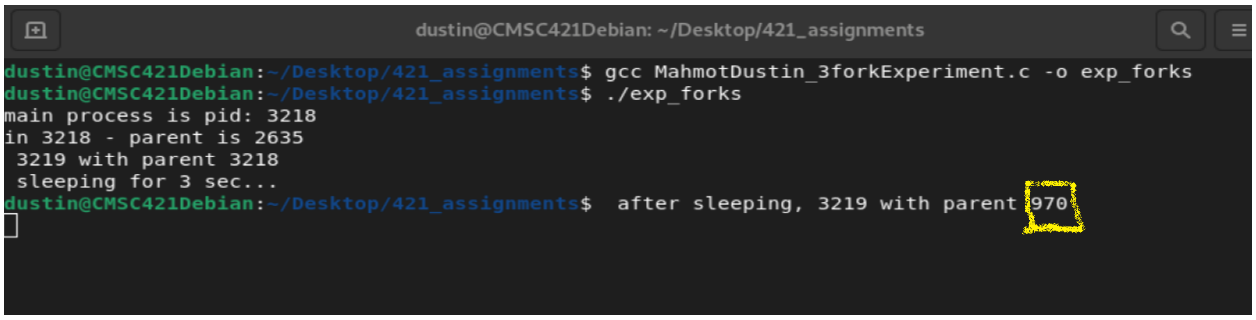




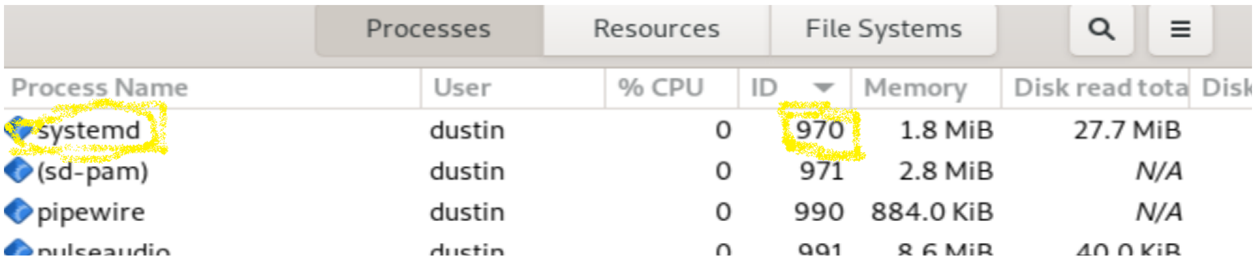
```
dustin@CMSC421Debian:~/Desktop/421_assignments/forkAssignment$ ./new3forks
FORK 1 from 5212
my pid: 5212, waiting for 5213
my pid: 5213, parent pid: 5212
FORK 2 from 5213
my pid: 5213, waiting for 5214
my pid: 5214, parent pid: 5213
FORK 3 from 5214
my pid: 5214, waiting for 5215
my pid: 5215, parent pid: 5214
5214 is done waiting for 5215.
5213 is done waiting for 5214.
FORK 3 from 5213
my pid: 5213, waiting for 5216
my pid: 5216, parent pid: 5213
5213 is done waiting for 5216.
5212 is done waiting for 5213.
FORK 2 from 5212
my pid: 5212, waiting for 5217
my pid: 5217, parent pid: 5212
FORK 3 from 5217
my pid: 5217, waiting for 5218
my pid: 5218, parent pid: 5217
5217 is done waiting for 5218.
5212 is done waiting for 5217.
FORK 3 from 5212
my pid: 5212, waiting for 5219
my pid: 5219, parent pid: 5212
5212 is done waiting for 5219.
dustin@CMSC421Debian:~/Desktop/421_assignments/forkAssignment$
```

<u>Part 3</u>

For my “experiment”, I decided to drop back down to just one fork() and try to create an orphan/zombie process. I did this by forking once, and making the child process sleep for longer than the parent process. Therefore the parent process would terminate while the child process was still running. I wanted to see what the parent process of the child would be after its original parent terminated. Before sleeping, I had both processes print their own pid along with their parent’s pid. The parent (main) process had a parent pid of 2635, which turned out to be bash. The child process had a parent pid matching the main pid, which made sense. After the parent process terminated but before the child terminated, I had the child print its parent pid which turned out to be 970 (systemd). I would have thought 2635 would become the new parent, but after some googling and remembering back to class, that makes sense because systemd has a daemon thread running all the time to take care of processes whose parent terminates first.



```
dustin@CMSC421Debian:~/Desktop/421_assignments$ gcc MahmotDustin_3forkExperiment.c -o exp_forks
dustin@CMSC421Debian:~/Desktop/421_assignments$ ./exp_forks
main process is pid: 3218
in 3218 - parent is 2635
 3219 with parent 3218
 sleeping for 3 sec...
dustin@CMSC421Debian:~/Desktop/421_assignments$  after sleeping, 3219 with parent 970
```



Process Name	User	% CPU	ID	Memory	Disk read tota	Disk
systemd	dustin	0	970	1.8 MiB	27.7 MiB	
(sd-pam)	dustin	0	971	2.8 MiB	N/A	
pipewire	dustin	0	990	884.0 KiB	N/A	
pulseaudio	dustin	0	991	8.6 MiB	40.0 KiB	