

Rojin Atefimehr
WSU ID#: 011799957

Program 1

1) I uploaded the processes.cpp file separately on the Canvas.

2)

```
37 }
38
39 // when pid=0, create the second child
40 if (pid == 0) {
41     pid = fork();
42     if (pid < 0) {
43         perror("error for creating the fork");
44     }
45 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● [ratefime@sig4 CPTS_370]$ g++ -o processes processes.cpp
⊗ [ratefime@sig4 CPTS_370]$ ./processes kworker
28
● [ratefime@sig4 CPTS_370]$ ps -A | grep kworker | wc -l
28
○ [ratefime@sig4 CPTS_370]$
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● [ratefime@sig4 CPTS_370]$ g++ -o processes processes.cpp
⊗ [ratefime@sig4 CPTS_370]$ ./processes sshd
3
● [ratefime@sig4 CPTS_370]$ ps -A | grep sshd | wc -l
3
○ [ratefime@sig4 CPTS_370]$
```

```
20     }
21
22     if (pid < 0) {

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● [ratefime@sig4 CPTS_370]$ g++ -o processes processes.cpp
⊗ [ratefime@sig4 CPTS_370]$ ./processes scsi
7
● [ratefime@sig4 CPTS_370]$ ps -A | grep scsi | wc -l
7
○ [ratefime@sig4 CPTS_370]$ █
```

- 3) This program counts the number of running processor on the device by giving the name which is `argv[1]`. First I created a new processor with `fork()`. Then I created two pipes which is `pip1` and `pip2`. I used `pip1` for `ps` to `grep` and `pip2` for `grep` to `wc -l`. Then I started with checking the fork. If `pid < 0` we will get the fork error. According to the assignment table first I created the parent to wait for a child. Then I created the if loop for creating child and grandchild when `pid == 0`. So, I created the child (`wc -l`). In this step I close the writing in `pip2`, `pip1`, and duplicate the reading from `pip2`. Then executes the command `wc -l`. Second, I created the grandchild. In this step, I delete the reading from `pip2` and writing from `pip1` and duplicate the reading from `pip1` and writing from `pip2`. Then I execute `grep` command with given `argv[1]`. Then I created the great grand child. In here, I only duplicate the writing in `pip1` and then execute the `ps -A` command. Finally I was able to compile my program with `g++ -o processes processes.cpp` and then run it with `./processes kworker`. I got the same output as if I run it with `ps -A | grep kworker | wc -l`.