Name: Riley Lawson
Section:
University ID: 116555487

Lab 2 Reports

Summary:

Throughout this lab there are many experiment that help you identify and evaluate how a process works, how to identify a parent and child node along with its process state, and different ways to execute process tree including time slicing, process synchronization, system calls, and signal kills. Each experiment, you use, tweak, and experiment with the idea of what a fork is, how it works (and different ways to execute them), and how to use them for various purposes.

Lab Questions:

3.1:

6pts In the report, include the relevant lines from "ps -l" for your programs, and point out the following items:

- output
- process name
- process state (decode the letter!)
- process ID (PID)
- parent process ID (PPID)

```
Output: Process ID is : 18947
                Parent process ID is: 15193
                sleep is S
                ps is R

Process ID is : 18947
Parent process ID is: 15193
Process State: S or Sleeping
Process name: sleep
```

2pts Repeat this experiment and observe what changes and doesn't change.

When doing the same experiment, the only difference is the Process ID and the ADDR. Everything else not mention is left unchanged.

2pts Find out the name of the process that started your programs. What is it, and what does it do?

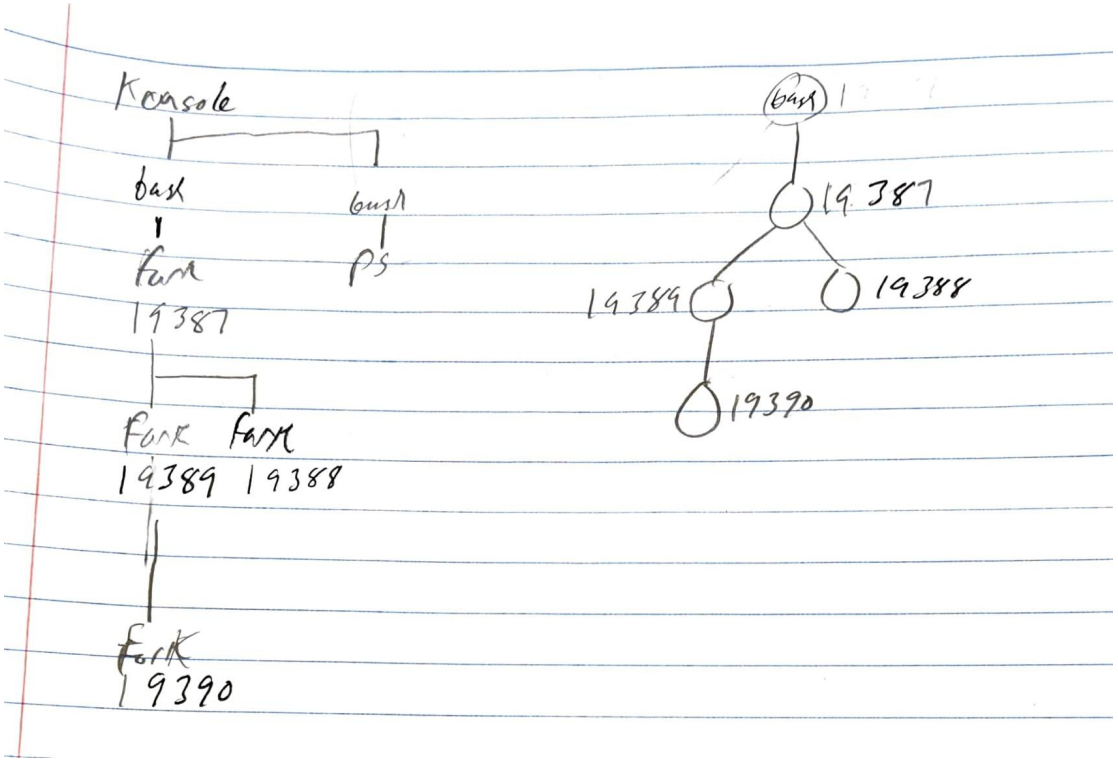
The parent process was bash which also came from konsole which came from systemd. Bash is the command line tool that is able to control and direct files and directories.

3.2:

1pt Include the output from the program

```
Process 19356's parent process ID is 15193
Process 19358's parent process ID is 19356
Process 19357's parent process ID is 19356
Process 19359's parent process ID is 19357
```

4pts Draw the process tree (label processes with PIDs)



3pts Explain how the tree was built in terms of the program code

A parent node (root) had a child node, then that child node had two more child nodes. At that point one those child nodes (process 19380) had an additional child node. This has a depth 3 or a full depth of 5 (if including the konsole).

2pts Explain what happens when the sleep statement is removed. You should see processes reporting a parent PID of 1. Redirecting output to a file may interfere with this, and you may need to run the program multiple times to see it happen.

When it becomes one it means that the parent process is killed before the children processes finish

3.3:

2pts Include the (completed) program and its output

The parent process ID is 22060
The parent's parent process ID is 15193
The child process ID is 22061
The child's parent process ID is 22060

4pts Speculate why it might be useful to have fork return different values to the parent and child. What advantage does returning the child's PID have?

Otherwise, there is no concrete way to be able to retrieve all of child's PID's. The advantage would be to be able to send information to the parent of that process. Since the child's PID is 0 then it always calls the parent function.

3.4:

2pts Include small (but relevant) sections of the output

```
Parent: 499980
Parent: 499981
Parent: 499982
Parent: 499983
Parent: 499984
Parent: 499985
Parent: 499986
Parent: 499987
Parent: 499988
Parent: 499989
Parent: 499990

and

Parent: 499397
Child: 499544
Parent: 499398
Child: 499545
Parent: 499399
Child: 499546
Parent: 499400
Child: 499547

and

Child: 499543
Parent: 499397
Child: 499544
Parent: 499398
Child: 499545
Parent: 499399
Child: 499546
Parent: 499400
Child: 499547
Parent: 499401
Child: 499548
Parent: 499402
Child: 499549
Parent: 499403
Child: 499550
```

4pts Make some observations about time slicing. Can you find any output that appears to have been cut off? Are there any missing parts? What's going on (mention the kernel scheduler)?

For a short period of time it stopped around the 10000 mark and then just rapidly finished at 499999. This makes since since the task isn't very long the kernel scheduler will take more time processing information instead of spending that extra energy doing every single output. Also, at the end there are only parent outputs instead of both parent and child outputs, and this also makes since with the sentence mentioned above.

3.5:

6pts Explain the major difference between this experiment and experiment 4.
Be sure to look up what wait does (*man 2 wait*).

The main difference between both of them is that during time slicing it goes between a PID of 0 and 1 and alternates for loops until both are completed. However, during process synchronous it has a wait(NULL); uses system calls to wait for state changes and doesn't start the parent until after its completed going through the child loop.

3.6:

2pts The program appears to have an infinite loop. Why does it stop?

It stops because the main parent was killed and the child no longer has a parent killing all further processes from happening and killing the existing ones.

4pts From the definitions of *sleep* and *usleep*, what do you expect the child's count to be just before it ends?

1000

2pts Why doesn't the child reach this count before it terminates?

Due to the .1 second delay of each child loop it adds up over 900 times and thus doesn't meet its initial goal.

3.7:

8pts Read the man page for *execl* and relatives (*man 3 exec*). Under what conditions is the *printf* statement executed? Why isn't it always executed?
(consider what would happen if you changed the program to execute something other than */bin/ls*.)

The printf statement wouldn't execute if the default environment does not contain the variable PATH. It's not always executed because they replace the current task with a new process; also if it's replaced with a different directory then it might not have access to that directory or have nothing to execute within it.

3.8:

2pts What is the range of values returned from the child process?

255 to -1

2pts What is the range of values sent by child process and captured by the parent process?

A value of 1

2pts When do you think the return value would be useful?
Hint: look at the commands *true* and *false*.

A return value that would tell you if it was a success or a failure which could be 1 or 0