Bartlett, Tyler
April 29, 2019
CS 365 – Operating Systems

Homework Assignment 2: Chapter 3

3.6 Describe the differences among short-term, medium-term, and longterm scheduling.

3.7 Describe the actions taken by a kernel to context-switch between processes.

3.8 Construct a process tree similar to Figure 3.9. To obtain process information for the UNIX or Linux system, use the command ps -ael. Use the command man ps to get more information about the ps command. The task manager onWindows systems does not provide the parent process id,yet the processmonitor tool available fromtechnet.microsoft.com provides a process tree tool.

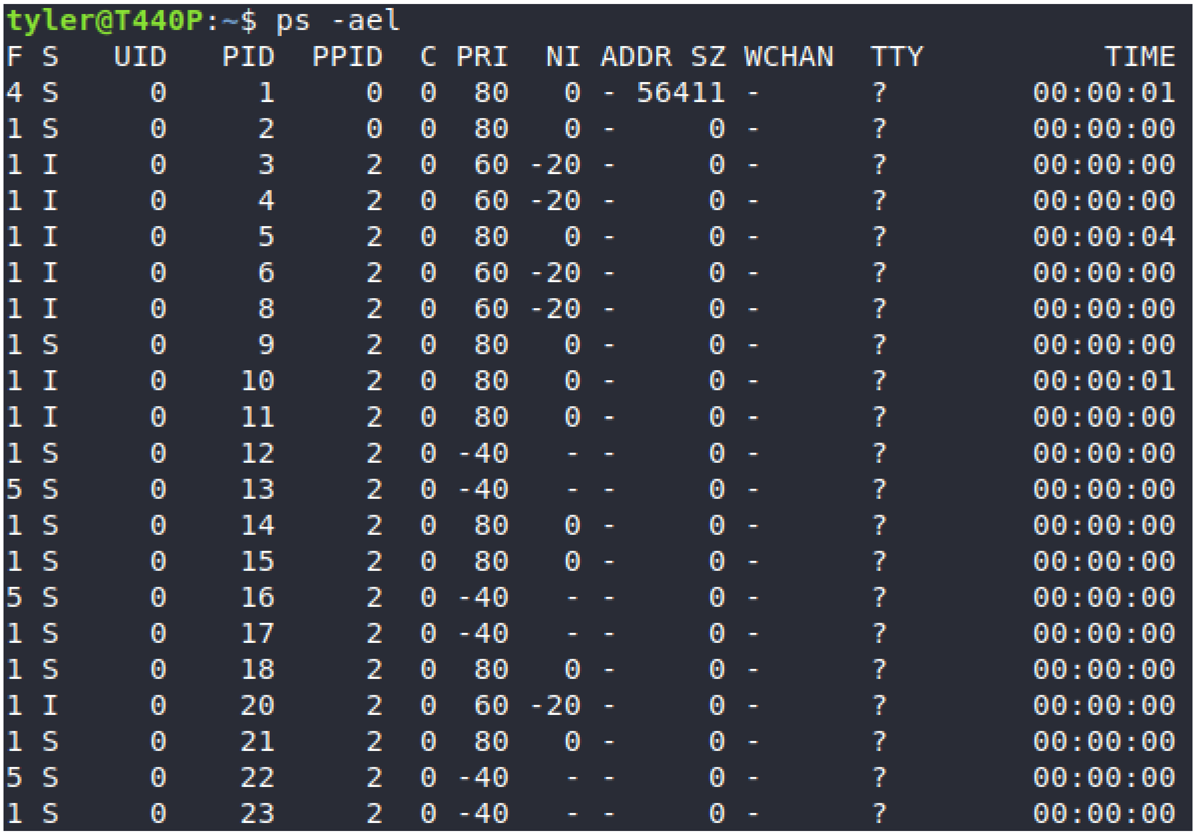
3.9 What are the pid values? Run the following code to prove your guess. Submit the result of the screen shot.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

        pid = fork();

        if (pid<0){
                printf(stderr, “fork failed”);
                return 1;
        }
        else if (pid == 0){
                pid1 = getpid();
                printf(“child: pid = %d”, pid);
                 printf(“child: pid1 = %d”, pid1);
        }
        else {
                pid1 = getpid();
                printf(“parent: pid = %d”, pid);
                 printf(“parent: pid1 = %d”, pid1);
                 wait(NULL);
        }
        return 0;
}
```

3.6. The long term scheduler chooses which programs are sent to the system to be processed and uses a queue to load them into memory for execution. The midterm scheduler controls process swapping by taking the processes out of memory. The short term scheduler changes the ready state of processes to running and allocates the CPU to the processes.

3.7. The Operating System saves the user stack pointer of the process currently being executed and transfers control to the kernel clock interrupt handler which then saves the states of the registers of the process. The Operating System calls the scheduler to determine the next process to execute then retrieves state from the Process Control Block and restores its registers to the state before the interrupt.

3.8.



```
tyler@T440P:~$ ps -ael
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY           TIME
4 S     0     1     0  0  80   0 - 56411 -      ?         00:00:01
1 S     0     2     0  0  80   0 -     0 -      ?         00:00:00
1 I     0     3     2  0  60 -20 -     0 -      ?         00:00:00
1 I     0     4     2  0  60 -20 -     0 -      ?         00:00:00
1 I     0     5     2  0  80   0 -     0 -      ?         00:00:04
1 I     0     6     2  0  60 -20 -     0 -      ?         00:00:00
1 I     0     8     2  0  60 -20 -     0 -      ?         00:00:00
1 S     0     9     2  0  80   0 -     0 -      ?         00:00:00
1 I     0    10     2  0  80   0 -     0 -      ?         00:00:01
1 I     0    11     2  0  80   0 -     0 -      ?         00:00:00
1 S     0    12     2  0 -40   - -     0 -      ?         00:00:00
5 S     0    13     2  0 -40   - -     0 -      ?         00:00:00
1 S     0    14     2  0  80   0 -     0 -      ?         00:00:00
1 S     0    15     2  0  80   0 -     0 -      ?         00:00:00
5 S     0    16     2  0 -40   - -     0 -      ?         00:00:00
1 S     0    17     2  0 -40   - -     0 -      ?         00:00:00
1 S     0    18     2  0  80   0 -     0 -      ?         00:00:00
1 I     0    20     2  0  60 -20 -     0 -      ?         00:00:00
1 S     0    21     2  0  80   0 -     0 -      ?         00:00:00
5 S     0    22     2  0 -40   - -     0 -      ?         00:00:00
1 S     0    23     2  0 -40   - -     0 -      ?         00:00:00
```

3.9. child: pid = 0child: pid1 = 23735parent: pid = 23735parent: pid1 = 23734