**CS 33: Homework 3**

8.15. This question essentially deals with understanding processes and how to create new processes. The main takeaway message is the fork() and what happens to our multiple programs when fork() returns. For this question, the program prints out nine “Example.”

I shall now attempt to draw my process tree.

12.17.

a. This is probably due to the fact that our main thread exited the program before the tid thread was able to run the thread function. A possible fix for this is for the main thread to wait for the tid thread to finish before terminating the program.

b. Like I said earlier for part a, we need to wait on the tid thread to finish before leaving the program. The appropriate pthread function would be pthread\_join.

Syntax: ptherad\_join(tid, NULL);

12.29. No the following program cannot deadlock because given the total ordering of all mutexes, each thread acquires its mutexes in order and releases them in that reverse order. This is also because given the defined ordering of locks, when the threads are acquiring, that total ordering is followed. This sequence of posts and waits will not result in one thread depending on resources on another thread and that other thread on our initial thread.

9.11

virtual address: 0x027c

a.

0 0 0 0 1 0 0 1 1 1 1 1 0 0

13.. 12.. 11. 10. 9.. 8.. 7.. …6 . 5…. 4…3… 2… 1.. 0

b.

VPN: 0x09

TLB index: 0x01

TLB tag: 0x02

TLB hit? : No

Page fault: Y

PPN: --

c. Can skip

d. Can skip

9.15

Request Block sizes(decimal bytes) Block header(hex)

Malloc(4) 8 0x09

Malloc(7) 16 0x11

Malloc(19) 24 0x19

Malloc(22) 32 0x21