Lab 3 Answers

kprintf blocks interrupts so printing isn't all over the place.

Problem 3
DEBUG define is commented out in xinu.h

f. The context switcher uses a priority queue so that the highest priority process will be first in the ready list. If there are multiple processes of the same priority, it does a round robin between them. If a lone high priority process terminates quickly, the cpu time is low.

Round robin will switch processes of the same priority after quantum time is up.

Problem 4
4.2
Reused test cases from problem 3.

4.3
1. The process all have the same huge priority and will round robin during context switching. They all finish roughly around the same time with similar preempt, priority, cpu time. The null process isn't caught so it is also mixed in with the processes in the round robin after awhile.

2. The sleepms function gives up its time slot for other processes to run. Depending on the length of sleepms, the process would end all the same time.

3. CPU bounds finishes before IO bounds since it was created first and the sleepms in IO bounds lets the CPU bounds to run while it sleeps so CPU bounds get more time. IO bounds finish a lot later since 5ms is a much longer time than the calculations.

Catching null process in the dynamic prioritization improves the running of all these test cases. The finish times cluster much tighter when null process priority stays 0.

This dynamic fair scheduling implementation is pretty fair since it decreases the priority of the higher priority processes after it has been ran for sometime so that it is able to let other slightly lower priority processes to run before running itself again.

Bonus
To implement RT processes, the procent struct will need to have an

identifier for whether or not the process is RT. The context switcher will check if the process is RT and will not context switch the entire time the process is run. The process will let the context switcher know once it's done so it can run the next process in the ready list. The priorities for both RT and TS process are treated the same in the ready queue. It is only when the context switcher deciding on running the next process where the RT and TS makes a difference.