

CS 574
Midterm Exam
October 12, 2007

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your Banner ID number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 11:20 to finish the exam. The questions are equally weighted.

1. Consider a process running under Linux, in a Xen virtual machine, on a processor that provides only User and Kernel modes of operation (no numbered protection rings like on Intel). If the process wants to read a single byte from a device, what is the sequence of transitions across protection levels from when the process calls `read()` to when the call returns? Assume a trivially simple device in which all that has to be done to read a byte is to just read it from the device (no device setup, no checking for a Done condition, no interrupts, etc).
2. Would it be possible to put a JFFS2 file system on a conventional (rotating magnetic media) disk? Discuss why it would or would not be a good idea.
3. A system using start-time fair queuing is executing three processes, one with a weight of 1, a second with a weight of 2, and a third with a weight of 3.
 - (a) Over time, what share (proportion) of the CPU will the each of the processes get? You should get an answers like 0.25, 0.33, etc.
 - (b) Now suppose a fourth process becomes runnable with a weight of 2. Adjust the weight of the third process (the one that previously had a weight of 3) so it continues to get the same share of the CPU that it did before.
4. A problem with allowing user-level processes to do input and output in architectures that use memory-mapped IO is granularity: since protection is on a page level, it's all too possible that more than one device will have its control registers on a single page. How could a scheme similar to the lock bits in IBM 801 page table entries be used to control IO access at a finer granularity?