

**CS 574**  
**Final Exam**  
**December 5, 2005**

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 social security number, but not your name, on each sheet of paper you turn in
- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 10:00 to finish the exam.

1. In a computer system using Alewife's LimitLESS protocol, support variable  $x$  is homed (*ie* has its directory) on processor A, and variable  $y$  is homed on processor B. There are two valid copies of  $x$ , on processors C and D, and one valid copy of  $y$ , on processor C.

Now suppose processor D executes the following line of code:

$x = y;$

List all the messages that need to pass between the processors to accomplish this. Be sure to give the source processor, destination processor, message type, and name of the variable for each message.

2. One computer wants to send the message 10110101 to another computer using a cyclic redundancy check to assure correctness. Give the resulting message (including both the original message and the frame check sequence) if a CRC generating polynomial of  $x^3 + x + 1$  is used.
3. It is much easier to share memory regions between processes in a computer using conventional page tables than one using either inverted or hashed page tables.
  - (a) Why?
  - (b) When a computer uses inverted or hashed page tables, there has to be some scheme for resolving hash collisions. Can something similar to a hash collision resolution scheme be used to make it easier to do shared memory?
4. Would it be possible to implement an NFS server that used Google FS internally?
5. How are block devices different from character devices in ways that (1) make it possible for Linux to use a different structure for block devices than for character devices, and (2) make it a good idea. I'm specifically thinking of the separation between the interaction with user processes from the strategy used to schedule reads and writes in block devices.
6. In a uniprocessor system, if one process sends a signal to another one, it isn't necessary to make sure the signal is delivered to the receiving process immediately; instead, some signal-related data structures are modified and the sending process is able to resume immediately. Why is this sufficient to ensure the receiving process will receive the signal (assuming it isn't blocked or ignored) before it performs any additional work?
7. The addresses of Intel page tables, as specified by CR3 (the PDBR) and by the Physical Frame Numbers in the tables themselves, as physical (as opposed to virtual). Since Linux runs with virtual memory enabled, how is it possible for the kernel to manipulate the contents of these tables?