Department of Computer Science
Operating Systems Qualifying Exam
Fall, 2003

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:

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

1. (20 points) The C operator

   ```
   i++
   ```

   increments the value of the variable `i`, and returns its former value. For this problem, assume that `i` is in memory, not a register.

   (a) In most C implementations, this is not an atomic operation. Describe, in terms of the operator's implementation and the hardware operations needed to execute it, why it isn't.

   (b) Suppose we modify the semantics of the `i++` operator to make it an atomic operation (but leave the semantics otherwise unchanged). How could you take advantage of this change in writing new programs in a shared memory environment?

2. (30 points) Suppose you are implementing a high-performance networked server. Clients send messages to your server, and your server processes them. You have adopted a multi-threaded design, so each message to arrive causes a new process to be spawned.

   Messages are delivered without loss, but may be delivered out of order.You can always obtain the sequence number of a message `M` by calling the function `seqno(M)`. You may assume that sequence numbers do not wrap around to 0.

   Even though messages may be delivered out of order, there is a critical section in your code which must process them in-order (so, if a message arrives but one or more of its predecessor messages have not arrived, the process handling the new message must wait until all of the predecessor messages have arrived and been handled).

   Write code, using synchronization primitives of your choice, which will guarantee that processes will execute the critical section in-order.

3. (20 points) In a computer system using Alewife's LimitLESS memory consistency protocol, variable `X` is homed on processor P0, with valid copies on P1 and P2 (but not on P0 itself). Variable `Y` is homed on processor P1, and the only valid copy is on processor P4.

   (a) What are the contents of the pointer sets for `X` and `Y`?

   (b) If processor P3 executes the statement

   ```
   Y = X;
   ```

   what sequence of messages will be transferred between the processors in the system?

4. (30 points) Consider a distributed computer system. Some of the nodes in the system will be specialized for processing; some will be specialized for file serving; some will be specialized for the user interface. In general, a user seated at an interface node will be executing programs running on one or more processing nodes (only a very small number of processes, directly related to the interface, will be running on the interface node); the processing nodes will be obtaining data from the file server nodes. All the nodes are fully featured computers, running an operating system such as Linux or Windows NT which has good support for virtual memory, processes, and any other operations you might find useful in implementing any of the classes of nodes described above.

   In this problem, consider the following questions regarding the configuration of the nodes in this system. Naturally, your reasoning is far more important than the final conclusions you reach.

   (a) Would it be useful for the processing nodes and interface nodes to have local disk drives?

   (b) Which classes of nodes would you expect to need the most memory?

   (c) Would you expect the process scheduling algorithms to be the same on the various node classes? If not, how would they differ?