CPS 210 Final Exam

Spring 2003

Except for the obligatory synchronization problem, this exam is even more "free form" than the last one. Please be concrete and specific and do not waste any words: answers are graded on content, not style. Bullet lists with sentence fragments are acceptable and even preferred. Diagrams and examples are helpful. There are **five** equally weighted questions for a total of 200 points. **Note:** question 5 is on the back.

Synchronization

1. Tweedledum and Tweedledee are separate threads executing the code loop below. The code is intended to cause them to forever take turns exchanging insults through the shared variable x in strict alternation. The `sleep()` and `wakeup()` routines operate as follows: `sleep` blocks the calling thread, and `wakeup` unblocks a specific thread ("ready to run") if that thread is blocked, otherwise it is ignored.

```
loop forever {
        x = insult(x);
        wakeup(other tweedle thread);
        sleep();
}
```

(a) This code illustrates a well-known synchronization flaw. Identify the flaw. Briefly outline a scenario in which this code will fail, and the outcome of that scenario.

(b) Show how to fix the problem by using semaphore P (down) and V (up) operations instead of `sleep` and `wakeup`.

(c) Implement Tweedledum and Tweedledee correctly using a mutex and condition variable.

Isolation

The following questions deal with *isolation* properties for computing systems that host multiple programs or workloads. *Fault isolation* is the property that a failure of any program does not affect the others. *Performance isolation* is the property that each program or workload achieves a stable and predictable level of performance independent of the resource demands of the others.

2. Evaluate the degree to which 'classical' operating system kernels (e.g., Multics, Unix, Nachos) achieve fault isolation. Briefly outline the key mechanisms for fault isolation in these systems.

3. Evaluate the degree to which classical operating systems achieve performance isolation. Outline why they succeed or fail, with reference to the mechanisms and policies for memory management, CPU scheduling, and I/O.

4. This semester we explored several examples of abstractions for performance isolation, i.e., interfaces to express performance isolation goals to the system. Outline a few of your favorite ones and discuss their relative strengths and weaknesses with reference to their semantics and implementation.

Virtual Copies

5. Shadowing and copy-on-write are closely related techniques to make a virtual copy of a logical block space, while deferring the copy operation for each block until that block is actually modified. Outline three examples of how these techniques are used in real systems. For each example, explain briefly how the copy and block write operations are handled, and explain what benefits and/or costs result from using the technique.

Have a great summer!