**SPRING 1997**
**COMPUTER SCIENCES DEPARTMENT**
**UNIVERSITY OF WISCONSIN—MADISON**
**PH. D. QUALIFYING EXAMINATION**

Operating Systems
Qualifying Examination

Monday, February 3, 1997
3:00 – 7:00 PM
2365 Computer Sciences and Statistics

**GENERAL INSTRUCTIONS:**

1. Answer each question in a separate book.
2. Indicate on the cover of *each* book the area of the exam, your code number, and the question answered in that book. On *one* of your books list the numbers of *all* the questions answered. *Do not write your name on any answer book.*
3. Return all answer books in the folder provided. Additional answer books are available if needed.

**SPECIFIC INSTRUCTIONS:**

Answer any *six* of the *seven* questions. Before beginning to answer a question, be sure to read it carefully. If you are confused about what the question means, state any assumptions you have made in formulating your answer. Good Luck!

**POLICY ON MISPRINTS AND AMBIGUITIES:**

The Exam Committee tires to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and ambiguities. If you are convinced a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the *first hour* of the exam. In any case, you should indicate your interpretation of the problem in your written answer. Your interpretation should be such that the problem is nontrivial.

# Operating Systems Exam

**Instructions:**
Answer any six of the following seven questions.

**Question 1. Global Snapshots:**

Chandy and Lamport's global snapshot algorithm is based on detecting stable properties.

(1a) What is a stable property and why does the algorithm contain this restriction?

(1b) What guarantees are given on the termination time of the C&L's snapshot algorithm?

**Question 2. Web Servers:**

As the popularity of Web grows, the performance of Web servers becomes an important issue. To improve performance, many commercial Web servers cache documents in the server's virtual memory instead of relying on the operating system's file caching.

(2a) Explain why this strategy might improve the performance of a Web server.

(2b) How should this cache be managed? Would LRU (least recently used) replacement be appropriate? Can you suggest a better policy? Justify your answers.

(2c) If Web servers cache documents in the server's virtual memory, what problem arises when the author of a document wants to modify it? Suggest at least two solutions to this problem.

**Question 3. Resource Management:**

There are four major resources in a computer that are under the management of the kernel: CPUs, memory, disks, and network interfaces.

(3a) Describe one good scheduling policy for each of the four resources.

(3b) In traditional UNIX, the scheduling of these resources is not coordinated—each uses its own scheduling algorithms. What are the advantages and disadvantages of such a design? What problems arise when such an operating system is used to support real-time applications?

**Question 4. Networking:**

To secure network communication and reduce bandwidth requirement, many systems apply both encryption and compression to a message before sending it out.

(4a) Should encryption be performed before compression, or the other way around? List two major reasons for the ordering of the operations.

(4b) Should compression be performed before message fragmentation or after? Why? Should encryption be performed before fragmentation or after? If encryption is performed before fragmentation, what problems might arise? If it is performed after fragmentation, what care should be taken to ensure the encrypted message does not reveal repeating patterns in the message?

**Question 5. Deadlock:**

One strategy for combating deadlock is *detection and recovery*.

(5a) A resource allocator accepts three kinds of request from other processes: `allocate(resource_class, number_of_units)`, `release(resource_class, number_of_units)`, and `check_for_deadlock()`. Outline an algorithm for implementing these functions, assuming a strategy of deadlock detection and recovery. What additional information would be necessary for deadlock avoidance?

(5b) Suppose there are several resource allocators, each responsible for some set of resource classes. A client process sends each request to the appropriate allocator. (Assume there is some mechanism for clients to figure out the appropriate allocator for each request.) A `check_for_deadlock` request can be sent to *any* allocator. Allocators may exchange messages with each other to detect deadlock. Outline a deadlock detection algorithm in this environment. You may assume there only one unit of each resource and you may ignore the possibility of process or communications failures.

**Question 6. Migrating Processes:**

Many distributed operating systems allow processes to move from machine to machine while they are running. Each of these systems must solve the forwarding problem: How does process *A* continue to communicate with process *B* while both processes may be moving around? There are many possible algorithms for solving this problem. Describe three such algorithms and compare their performance. Be sure to discuss how the choice of algorithm depends on such issues as processor and communications medium performance, probability of failures, and expected traffic patterns.

**Question 7. Protection:**

An experimental addition to Unix allows a user to connect a "watchdog" program to a file such that the watchdog is invoked whenever a program requests access to the file. The watchdog then either grants or denies access to the file. Discuss the pros and cons of using watchdogs for security.