

CSE 120 Winter 2002 Final Examination

There are six questions in this examination, and the examination is worth 120 points. Please answer all questions. You may refer to your textbook, your notes, and printed copies of the material that I posted on the web site.

You have three hours to complete this examination.

Problem 1 (12 points)

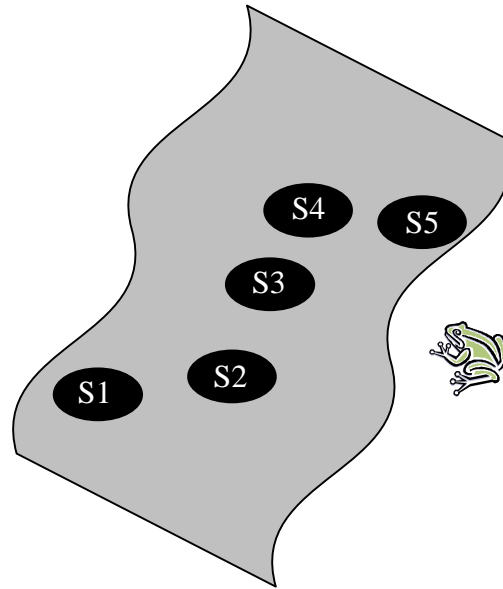
Consider the following proposed solution the two-process mutual exclusion problem:

```
int in0 = 0, in1 = 0;
cobegin
while (1) {
    in0 = 1;
    while (in1) {
        in0 = 0;
        while (in1) ;
        in0 = 1;
    }
    critical section
    in0 = 0;
||
while (1) {
    in1 = 1;
    while (in0) {
        in1 = 0;
        while (in0) ;
        in1 = 1;
    }
    critical section
    in1 = 0;
coend
```

- a) Is this a *safe* solution to the mutual exclusion problem? If so, then give a brief argument why; if not, then give a behavior of the program that violates the safety property of mutual exclusion.
- b) Is this a *live* solution to the mutual exclusion problem? If so, then give a brief argument why; if not, then give a behavior of the program that violates the liveness property of mutual exclusion.

Problem 2 (20 points)

Consider the stream illustrated below. To cross the stream, you can walk across the five stepping stones in order. In particular, if you are crossing from the left side to the right side, then you would walk across the stones in order S1; S2; S3; S4; S5. If you were crossing in the other direction, then you would walk across the stones in order S5; S4; S3; S2; S1. The stones are small, large enough for only one person's foot.



a) If you were to model this system of the stream and the stepping stones, you would model each of the stones as a separate resource of one unit rather than the collection of stones as being a one resource with five units. Why?

b) Using binary semaphores to represent the stones, give a sequence of P and V operations that models a person crossing from the left side to the right. Give another sequence of P and V operations that models a person crossing from the right side to the left. Assume that a person, having decided to cross, does not change her mind, back up, jump into the water, or do other creative acts.

c) Show that a deadlock is possible in this system. Illustrate the deadlock state with a resource allocation graph.

d) For the purposes of this part, you can assume that Antonia and Beppo are on the left side and Carlo is on the right side. Suppose these three people wish to use the Banker's algorithm to avoid deadlock. In practice, how would using the Banker's algorithm constrain how and when they cross?

Problem 3 (28 points)

The shell line

```
echo hello > foo
```

creates a file `foo` (assuming that it doesn't already exist), opens the file, writes the string "hello" to the file, and closes the file.

- a) How many file data blocks are directly written as a result of this execution? Which blocks are they?
- b) What metadata is modified by the execution of this shell line?
- c) Give an instance of a reliability-induced synchronous write that will be generated during the execution of this shell line.
- d) Explain what benefit, if any, the *inode cache* gives during the execution of this shell line.
- d) If you were designing a file system, you would need to decide whether the `close` operation would flush cached data blocks of that file. What are the benefits and drawbacks of doing so?

Problem 4 (20 points)

Here is the Needham-Schroeder secret key authentication protocol. A and B denote the two parties that wish to agree on a session key, and S denotes the authentication server.

1. $A \rightarrow S: A, B, I_A$
2. $S \rightarrow A: \{I_A, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$
3. $A \rightarrow B: \{K_{AB}, A\}_{K_B}$
4. $B \rightarrow A: \{I_B\}_{K_{AB}}$
5. $A \rightarrow B: \{I_B - 1\}_{K_{AB}}$

- a) How does this protocol allow A and B to be confident that it is S , rather than an imposter, that generated the key K_{AB} ?
- b) How does the protocol allow A to be confident that it is B , rather than an imposter, with whom it establishes the session?
- c) The designers of this protocol said that the nonce identifiers I_A and I_B should be randomly generated numbers. Why? How could the security of this protocol be threatened if they were not randomly generated?
- d) Explain why replacing the fifth message with

$$A \rightarrow B: \{I_B\}_{K_{AB}}$$

is not a good idea.

Problem 5 (20 points)

Consider the code in an operating system that services page faults. We'll call this code the *page fault manager*.

- a) Give a list of the tasks that the page fault manager does, starting with a page fault generated when process p faults on page v and terminating with resuming the process that took the page fault. You can assume that there is a free page frame f available for the page fault manager to use.
- b) Typically, the code of the page fault manager and the data it references is *pinned*: the pages of this code and data are permanently assigned page frames. Otherwise, the system can end up in a state in which the page fault manager can make no progress. Explain how this could happen.
- c) One of the tasks of a page fault manager is compute the location of page v in backing store. Briefly explain why this computation is relatively straightforward if the operating system does not support memory mapped files.
- d) This computation is more complex if the operating system support memory mapped files because each page may be associated with a different file (or with an anonymous swap file).

The *Pilot* operating system (an operating system built at Xerox PARC) only supported memory mapped files. It also allowed files to be stored either on remote file servers or on the local disk.¹ For a page that was mapped to a file that was stored on a remote file server, the page fault manager had to reference a fairly large data structure to determine its location. Because of the size of these data structures, both the code that computed the location for pages mapped to files on remote file servers and the data it referenced was not pinned.

Recalling your answer to part (b) above, not pinning this code and data structure can result in the page fault manager not being able to make progress. The Pilot architects addressed this problem by enforcing the rule that *the swappable code of the page fault manager must use the local disk for backing store*.

Why does enforcing this rule ensure that the page fault manager can always eventually make progress?

¹ This description considerably undersimplifies the *Pilot* architecture.

Problem 6 (20 points)

Some short questions on VM:

a) If a large number of programs is kept in main memory, then there is almost always another ready program when a page fault occurs. Thus, CPU utilization can be kept high. If, however, we allocate a large memory space to each of a few programs, then each program produces a small number of page faults. Thus, CPU utilization is kept high.

Are these two arguments correct? Which, if either, of the two policies should be preferred?

b) Show a page reference string and a number of page frames such that a FIFO page replacement policy generates fewer page faults than an LRU policy.

c) Why is an exact LRU page replacement policy difficult to implement?

d) The designers of the Unix virtual memory system added a free page pool that maintained a list of free page frames. The clock algorithm would run when the number of page frames in this pool dropped too low. Why did they add this feature? After all, implementing a free page pool removes page frames from use, thereby reducing the physical memory available for paging.

e) In the same virtual memory system as the previous question, how did the designers address the problem of thrashing? In particular, how was thrashing detected and what would the system do to stop thrashing?