

Operating Systems Comprehensive Examination
Spring Quarter, 2002

ID _____

NOTE: Please write your ID on every page of the exam.

1	
2	
3	
4	
5	
Total	

1. (10 pts) **Synchronization.**

Consider Peterson's algorithm for mutual exclusion, given below:

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

Suppose the critical section code for each process is a short operation, such as an increment instruction. If you were to run this multiprocess program on operating systems (such as Unix) that use a time sliced round robin scheduler, you would find that initially the rate that the processes enter their critical sections would be fast, but after some time elapsed their rate would suddenly become dramatically slower.

- (a) Explain this behavior by giving an execution sequence that results in the slower rate of entrance into critical sections.
- (b) How could you predict the amount by which the rate would be decreased?
- (c) Can you think of a way to fix the problem so that the rate the processes enter the critical sections remain fast no matter what their execution sequence is? Be as creative as you need to be, but do not spend too much time on this part of the question; if no idea immediately occurs to you, say so and move on.

2. (10 pts) **Virtual Memory.** You are given a machine architecture whose hardware support for paged virtual memory is limited to a TLB. No hardware translation based on page tables in memory is supported. When a virtual address is submitted to the hardware, if there is a match in the TLB, a successful virtual-to-physical address translation occurs. If there is no match, a fault occurs, causing a trap to a VM address translation handler in the operating system which is provided the virtual address. Describe (a) what this handler does, (b) what kind of data structure(s) it manages, and (c) operationally, how the virtual address is ultimately translated.

3. (10 pts) **Protection.**

	Domain 1	Domain 2	Domain 3	File 1	File 2	Process 1
Domain 1	*owner control	*owner control	*call	*owner *read *write		
Domain 2			call	*read	write	wakeup
Domain 3			owner control	read	*owner	

Figure 1: Portion of an access matrix (* denotes that copy flag is set)

Recall that in Lampson's *Protection*, A is the access control matrix and an entry $A_{d,x}$ is the set of access permissions domain d has on object x . He further defines the following rules for manipulation the entries:

- (a) d_1 can remove access attributes from $A_{d_2,x}$ if it has 'control' access to d_2 . Example: domain 1 can remove attributes from rows 1 and 2.
- (b) d_1 can copy to $A_{d_2,x}$ any access attributes it has for x which has the copy flag set, and can say whether the copied attribute shall have the copy flag set or not. Example: domain 1 can copy 'write' to $A_{2,File\ 1}$.
- (b) d_1 may add any access attribute to $A_{d_2,x}$, with or without the copy flag, if it has 'owner' access to x . Example: domain 2 can add 'write' to $A_{2,File\ 2}$.

Your company is building an application where an application has to give access to an object x to a second program that is untrusted and may be potentially malicious, where only that second program should have access and no others. The operating system your application is hosted upon directly implements Lampson's protection mechanism. Your colleague Bob claims that the way the copy flag works completely prevents a subdomain with access to x from inappropriately giving away access to x to some other domain. Alice claims that he is wrong, and they have a \$10 bet on this. What is your opinion? Give detailed reasons, because otherwise Alice and Bob won't shut up.

4. (10 pts) **Page Replacement.** Consider the following virtual memory page replacement string:

1, 2, 3, 1, 4, 5, 1, 5, 2, 6, 4, 5

Assume that there are four page frames in physical memory, and that they are initially empty.

- (a) What is the sequence of page faults that occur if FIFO replacement is used?
- (b) What is the sequence of page faults that occur if LRU replacement is used?
- (c) What is the sequence of page faults that occur if MRU replacement is used?
- (d) What is an optimal sequence of evictions that results in the minimum number of page faults?

5. (10 pts) **File Systems.** The V Kernel and Sprite were two early operating systems that supported networked file systems. Recognizing that remote operations are slower than local operations, performance in these new file systems was a primary concern. One method both systems employed to improve performance was caching, although they each approached the problem slightly differently.
- (a) Describe how and where each system employed caching, and the motivation for their design choices.
 - (b) Which do you think would have been more effective at the time when these systems were implemented, and why?
 - (c) Since those operating system were implemented, processor speed has improved by 100x, network bandwidth has improved by 100x and latency by 10x, and disk bandwidth has improved by 25x and latency by 4x. Assuming similar workloads, have these technology trends made caching more or less important, and do these trends change your answer to question (b)?