

Operating Systems Comprehensive Examination
Spring Quarter, 2003

ID _____

NOTE: Please write your ID on every page of the exam.
Answer each question on its own page; you may use the back if you need extra space.
Answers on any other pages of the exam will NOT be graded.

Question	Score	Points
1		10
2		10
3		10
4		10
5		10
Total		50

1. (10 pts) **Performance optimization.** You are hired as a systems consultant for a large company that would like your advice. The company uses a Web proxy cache to reduce network utilization and reduce response time, and it wants to improve on both metrics if possible.

The proxy cache has two levels, a memory cache and a disk cache. Requests are first checked against the memory cache. Those that miss in memory are checked against the disk cache; requests that miss on disk are sent over the network to the origin server. From workload monitoring, the company's IT department tells you the following about the performance of the current proxy cache configuration:

- CPU utilization is 50%.
- Hit rate in the memory cache is 15%.
- Hit rate in the disk cache is 40%.
- Disk cache capacity miss rate is 5%.
- Disk utilization is 75% (handling I/O requests, not capacity).
- Average request latency of a proxy cache hit is 20ms.
- Average request latency of a proxy cache miss is 100ms.

- I) The company wants to increase performance by improving its hardware, and it has the following options for hardware upgrades. For each, specify whether you expect the change would be **useless** or **useful** in addressing each of the following goals and briefly explain why.

- (i) reducing network utilization
- (ii) reducing response time

- a) Replace the older CPU of the proxy cache with the very latest CPU on the market. The new CPU has twice the performance of the old CPU (it executes twice as many instructions per second

- b) Double the existing amount of memory.

- c) Double the existing amount disk space (using the same kind of disks).

- d) Replace the disks with newer models whose seek time is half that of the old disks.

- II) Which option do you think will have the most impact on improving the overall performance of the proxy cache, and why?

2. (10pts) **Synchronization.** Old cruffy Unix programmers occasionally implement mutual exclusion using a shell script like the following (which is written in `csh`, another favorite of the old and cruffy):

```
#!/bin/csh -f
while (-e foo.lock)
end
echo $$ > foo.lock
```

If this was in a file called `lock`, and you wished a program `foo` to run in mutual exclusion, then you would invoke the program as:

```
lock
foo
rm foo.lock
```

The guard in the `while` statement evaluates to true when the file `foo.lock` already exists, and so the effect of the `while/end` pair is to hang as long as the file `foo.lock` exists. The `echo` statement writes the parent's shell process ID to the file `foo.lock`.

- a) This is not a safe implementation of mutual exclusion. Explain why.
- b) Is this a starvation-free implementation? Explain.
- c) Would it be possible to implement a safe and starvation-free shell script for mutual exclusion? Explain. A simple argument will suffice; you don't need, for example, to give us such a script (assuming that one exists).

3. (10pts) **Layering.** The architectures of THE (by Edsger W. Dijkstra, described in “The structure of the THE Multiprogramming system”) and Swift (by David D. Clark, described in “The structuring of systems using upcalls”) are quite different: THE imposed a strict layering hierarchy such that code at layer i could only invoke code at a layer lower than i , while Swift allowed code at layer i to synchronously invoke code both above and below i .

a) What benefits do systems like THE derive from their strict layering?

b) What motivated Clark to violate the layering of Swift using upcalls?

c) How did Clark address the problems that Dijkstra avoided by having this limitation?

4. (10pts) **File Systems.** There are many trade-offs to consider when designing a distributed, network file system. For each of the various design decisions listed below, describe the relative merits (pros and cons) of each of the two options suggested and describe under what conditions each approach is superior to the given alternative.

a) Block sizes: large *vs.* small

b) Caching: local *vs.* remote

c) Server state: stateless *vs.* stateful

d) Access granularity: file level *vs.* block level

