Operating Systems

Instructions:
For the **BREADTH** exam: Answer any 4 of questions 1-6.
For the **DEPTH** exam: Answer any 4 of questions 1-6 *and* 3 of questions 7-10. You are expected to spend about two hours on each half of the exam.

BREADTH QUESTIONS

Breadth 1. Process Switching:

Context switching (moving a CPU from one process to another) can be very time consuming.

(1a) Outline some of the hardware and software features in current systems that contribute to the cost of context switching.

(1b) If the cost of a context switch is sufficiently high, it may be better for a processor to spin, waiting for an event (such as I/O completion) to occur than to switch to another process. Outline an algorithm for deciding whether to spin or switch assuming that the time for an I/O operation can be accurately determined when it is started.

(1c) How does your answer to part (1b) change if the time for an I/O operation is a random variable?

Breadth 2. File Systems: The Unix file system organizes all directories into a tree rooted at a unique *root directory*. By contrast, the file system outlined by Dennis and van Horn (*Programming Semantics for Multiprogrammed Computations*) is a general directed graph, with a *home directory* defined for each user. How is the Dennis and van Horn approach more convenient for users? Why do you suppose Unix imposes the more rigid structure? How do symbolic links in Unix mitigate these restrictions?

Breadth 3. Network File Systems:

The NSF file server provides distributed access to files by allowing you to mount a subtree from the file system on a remote host into your local file tree (in any place in the local tree that you wish).

(3a) Compare this design to that of the Locus file system.

(3b) Describe the advantages and disadvantages of the file naming scheme used in each system.

Breadth 4. Process Creation:

UNIX creates new processes by using the `fork()` operation. Another type of process creation operation supported by other operating systems looks like:

```
createproc(filename)
```

This operation causes a new process to be created, and this new process will execute the program (binary or a.out) found in file `filename`.

(4a) Compare this functionality to that provided by the UNIX `fork()`.

(4b) Describe the advantages and disadvantages of each design.

Breadth 5. Working Set:

The Working Set (WS) model is the basis of a nice page replacement algorithm. How would you extend the WS model to handle a shared-memory parallel computer where the processes can share pages?

Breadth 6. Synchronization:

A *Conditional Critical Region* is a synchronization operation of the form:

region *v* **when** *B* **do** *S*.

where *v* is the shared variable for that region and *B* is a boolean expression. If *B* is true, then statement *S* is executed. If *B* is false, then execution of this process is delayed until *B* becomes true and no other process is in the region associated with *v* (there can be several **region** statements with the same *v*).

Show how to implement Conditional Critical Regions using only semaphores (P, V, and Init) and ordinary data structures (NO busy waiting!).

DEPTH QUESTIONS

Depth 7. Distributed Checkpoints:

Condor is a system developed at the University of Wisconsin that runs cpu-intensive programs on otherwise idle workstations. Condor periodically checkpoints the core image of a job so that it can be moved to another workstation if the one it is running on is preempted for interactive use. Currently, Condor supports only one-process programs, partly because of the difficulty of obtaining a consistent checkpoint of a multi-process program. Outline some of the difficulties of checkpointing a program that may consist of several processes on one workstation, connected by pipes. (Assume you are not allowed to modify the Unix kernel). What additional problems arise if the processes are distributed over a local-area network? Outline an algorithm for checkpointing in this environment.

Depth 8. Long Addresses:

Several new processors have been announced that generate 64-bit virtual addresses. It has been proposed that such a huge virtual address space should be managed in a new way. In "traditional" systems, each process is given its own address space. Sharing is supported (if at all) by mapping pages of physical memory to two or more address spaces, perhaps at different virtual addresses. A *flat* scheme, by contrast, would have one huge virtual address space shared by all processes. Each process would only be authorized to access certain portions of the space.

What advantages does the flat scheme offer over the traditional approach? What new problems arise?

Depth 9. Global Time:

(9a) Assume that you have a computer with processors that communicate via messages. These processors each have their own clock. Can you use these clocks to record enough information so that you can recover the true global ordering of all events on the different processors?

(9b) Is it important to determine the true global ordering of all events? Argue for and/or against with examples.

Depth 10. Networking

(10a) Compare 802.3 (CSMA/CD) and 802.5 (Token Ring) networks with respect to

- fairness in bandwidth allocation among users,
- maximum delay before a frame can be sent,
- error detection, and
- retransmission and acknowledgement of frames.

(10b) Would problems arise if the bandwidth of an 802.3 network was increased by a factor of 10? If so, precisely describe the problems.

(10c) Same question as (10b) except for increasing the length or span of an 802.3 network by a factor of 10.

(10d) If 802.5 were to be modified so that recipients remove frames from the medium, what impact would that have on your answer to (10a)?