Operating Systems Exam

Instructions:
Answer any six of the following seven questions.

Question 1. Virtual Memory: Assume you are designing the virtual memory system for a machine with a RISC processor. Your cprocessor has a first-level cache and a TLB (translation look-aside buffer). The cache is indexed by physical address. A memory reference can miss in the cache, the TLB, or the page table (i.e., a page fault). A cache miss usually costs 10 CPU cycles, a TLB miss costs 50 CPU cycles, and a page fault costs 10,000 CPU cycles.

(1a) Which of the three kinds of misses are handled by the operating system?

(1b) What should the kernel do in response to those misses? Describe in detail the steps of the processing in the kernel.

(1c) There are eight possible combinations of those misses. Which of the combinations are possible and which are not? Which combinations, if they happen, affect the correctness of the program? What should the kernel do in response to these combinations?

Question 2. Copy on Write:

Copy-on-write is a way for processes to share pages (or segments) that are logically distinct. When one process sends some data to a second process with copy semantics, nothing needs to be copied immediately, or perhaps ever.

(2a) Explain how standard memory-management hardware can be used to implement copy-on-write cheaply.

(2b) Describe how this method can be used to improve system performance for:

(i) message passing,

(ii) forking a new process, and

(iii) file I/O operations.

Question 3. Datagram Fragmentation

When datagrams are transmitted through a series of networks connected by gateways, fragmentation is sometimes required.

(3a) Why might it be necessary to fragment datagrams at gateways?

(3b) Compare the relative merits of reassembling the datagrams at gateways versus doing the reassembly at the destination host.

Question 4. Distributed Shared Memory:

A single virtual address space can be supported simultaneously across the (homogeneous) processors of a collection of workstations distributed across a local area network.

(4a) Distributed shared memory may be implemented with standard paging hardware. Describe an algorithm for doing so by indicating the sequence of events for page fault handling when the desired page is not physically present. Assume virtual pages can be replicated on several processors, and that multiple processes may be attempting to write to the same page simultaneously. You may ignore the question of secondary storage (paging to or from a disk).

(4b) When a page fault occurs, the data may be moved or copied to the physical location of the faulting process. It is also possible to move the faulting process to the physical location of the data. Discuss the tradeoff between the two choices. When is process migration the appropriate choice? Make explicit your assumptions and expectations about process behavior.

Depth 5. Byzantine Generals Meet Reality

One solution to the Byzantine Generals problem makes the following assumptions about the underlying system:

(5a) Every message that is sent is delivered correctly.

(5b) The receiver of a message can reliably determine the identity of the process from which it was (most recently) sent.

(5c) The receiver of a signed message can reliably determine whether the message has been modified since it was signed.

(5d) Anyone can verify the authenticity of a signature.

Discuss whether each of these assumptions is reasonable in light of the way distributed systems really work. If an assumption is reasonable, describe how a system would support such a feature. Where applicable, provide examples from systems that you know. If an assumption is unreasonable, explain why the feature is difficult to provide.

Question 6. 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.

Question 7. Monitors The operating system usually associates a queue with each monitor *condition* variable. When a process performs a `wait(c)`, the processes is blocked and placed on the queue associated with `c`. When a process attempts to enter the monitor while another process is active in one of its procedures, it is blocked and placed on an "entry" queue. Typically, these queues are maintained in FIFO order.

(7a) What events can cause processes to be unblocked from each kind of queue?

(7b) Is it possible for a programmer *using* monitors to write code that can detect whether a condition variable queue is using FIFO order? It is possible to detect whether the entry queue is maintained in FIFO order? Explain.

(7c) Does your answer depend on whether the monitor is using Hoare or Mesa semantics? Explain.