

The exam is worth 100 points. There are 10 questions. Answer all questions.
You are to use ONLY the space provided, NOT the back of the page.
Please be brief, succinct, and to-the-point in your answers.
You will lose points for incorrect statements--even when your answer is correct!

(32 pts) **1) Circle the correct statements.**

- (1) For most operating systems, the physical memory is larger than the logical memory.
- (2) The main problem with using a set of dedicated registers as holding the page table is the dynamic register indexing problem.
- (3) Unix operating system prefetches the page table entries of a process in groups dynamically and selectively into the main memory as they are needed.
- (4) Deadlock prevention techniques prevent deadlocks at process execution time by maintaining cycle-free wait-for graphs where a wait-for graph is defined as a graph whose nodes are processes, and whose edges represent resource requests from the process requesting the resource to the process holding the resource.
- (5) In any thread package implementation, regardless of user-level or kernel-level package, each thread has its own separate program counter, stack space, and address space.
- (6) Global variables are handled by RPCs using the DGVM (Deferred Global Variable Management) mechanism.
- (7) Client and server stubs of RPCs always execute with supervisory/kernel privileges.
- (8) Sockets are in general implemented by employing the RPC mechanism underneath.
- T** (9) Happened-before relation is irreflexive and partial ordering.
- T** (10) Sockets have no location independence in that both the client and the server know each other's machine addresses and port numbers.
- T** (11) If processes could be preempted from the resources they hold, there would not be any deadlocks.
- T** (12) Given a wait-for graph with n nodes, detecting cycles (for the purpose of detecting deadlocks) takes $C \cdot n^2$ comparisons where C is a constant independent of n (i.e., $O(n^2)$ time).
- T** (13) Fork system call provides the only mechanism in UNIX where new processes are created.
- T** (14) Text segment of a Unix process never contains data, only contains code, and does not change during the lifetime of the process.
- T** (15) Unix NFS (Network File System) trusts the UNIX OSs involved in distributed file manipulation in that it accepts and never questions or verifies independently the access rights information provided by the OS representing the client with the file manipulation request.
- T** (16) Unix has no job scheduler as it does not allow batch processing.

(26 pts) **2) Define/explain in few words the terms**

- (1) LRU page replacement policy: *When there is a page fault, the new page that is brought from the secondary storage replaces the page that is least recently used among the pages that are in the main memory.*
- (2) mounting a directory in Unix: *The process of making a remote directory on another machine visible as a local directory in a local machine. Implemented by NFS.*
- (3) local frame allocation: *When a main memory frame is needed to place a new page brought in due to a page fault, only the currently allocated frames of the process are candidates.*
- (4) frame versus page in memory management: *Frame is a physical memory unit, and page is a logical memory unit. Frames house pages.*
- (5) dirty bit in page table entries: *A bit that indicates whether the page is modified, and the modifications are not yet copied to the secondary storage image of the page.*
- (6) throughput: *The number of jobs/user processes completed within a given unit of time (e.g., an hour).*
- (7) orphan extermination: *During RPC execution, the client passes a request successfully, the server starts the service execution, and the client dies or the client site goes down. In this case, the server is an orphan, and its computation needs to be stopped/terminated. Terminating the server processes is referred to as orphan extermination.*
- (8) Socket: *A special file that serves as the endpoint of process communication, possibly at different sites.*
- (9) RPC: *The mechanism through which the execution of remote procedures are requested, and the results are received back.*
- (10) Session Layer: *ISO/OSI network protocol layer responsible from session setup/management, dialog control, and party synchronization.*
- (11) process control block: *A data structure that contains, for each process, the process state, program counter, process id, registers, open files, etc..*
- (12) UDP protocol: *Universal Datagram Protocol, which is a connectionless transport layer protocol for reliable communication.*
- (13) Flattening in RPCs: *In RPCs, the process of transforming the RPC call parameters with data structures into a byte stream.*

(7 pts) 3) List the seven layers of the ISO/OSI network protocol, and the purpose of each of the layers.

Physical Layer: bit transmission.

Datalink Layer: reliable point-to-point frame (packet) communication: error checking and correction with checksums and retransmissions.

Network Layer: Routing and congestion Control; e.g., IP protocol (connectionless) and X.25 protocol (connection-oriented).

Transport Layer: Reliable end-to-end communication (includes data transportation) at the packet level; e.g., TCP protocol (connection-oriented) and UDP protocol (connectionless).

Session Layer: session setup/management, dialog control, and party synchronization.

Presentation Layer: Incorporating data semantics into messages (records, fields, types, etc.)

Application Layer: Protocol for the application at hand, e.g., mail, ftp, telnet, etc.

(9 pts) 4) (a) Define the "Happened-Before" relation.

A relation that defines a partial ordering of events in multiple sites using the two rules:

(i) If A and B are events in the same site, and, using the clock of the site, A happens before B then $A \rightarrow^{\text{Happened-Before}} B$.

(ii) Let A be the event of sending message M by one process, and B be the event of receiving M by another process. Then $A \rightarrow^{\text{Happened-Before}} B$.

(iii) Transitivity: Let A, B and C be events. If $A \rightarrow^{\text{Happened-Before}} B$ and $B \rightarrow^{\text{Happened-Before}} C$ then $A \rightarrow^{\text{Happened-Before}} C$.

(b) Define when two events A and B are concurrent.

If two events are not related by the relationship $\rightarrow^{\text{Happened-Before}}$ then A and B are concurrent. Or, if in the Space-Time diagram, there are no paths between A and B then A and B are concurrent.

(c) Show that one can impose a total global ordering among events in a distributed system using logical clocks, which does not violate the Happened-Before relation.

Each site s has a logical clock $LC_s()$ that is a monotonically increasing counter that "ticks" (gets incremented) when an event, say A , happens at the site; the event A at site s is assigned a logical clock value $LC_s(A)$. The events at all sites are totally ordered by their logical clock values as follows:

(i) In the same site s , if $A \rightarrow^{\text{Happened-Before}} B$ then $LC_s(A) < LC_s(B)$.

(ii) Let A be the event of sending message M from site s at time $LC_s(A)$, and B be the event of receiving the message M at site r . The message M is attached the logical clock value $LC_s(A)$. When r receives the message M , it first adjusts $LC_r()$ as follows: If $LC_r()$ is less than $LC_s(A)$ then $LC_r()$ is set to $LC_s(A)+1$. Then $LC_r(B)$ is initialized as the current (adjusted) value of $LC_r()$. Thus, we always have $LC_s(A) < LC_r(B)$.

(iii) If two events A and B have the same $LC()$ values then they have happened at two different machines, and they are "concurrent" events. Break the tie between A and B by using the `processId.MachineId` values, and create a total ordering.

(4 pts) 5) Assume n processes use the distributed mutual exclusion algorithm.

(a) What is the maximum number of messages for a process to enter into its critical section?
 $2(n-1)$

(b) State one problem with this algorithm.

Failure of any site breaks the algorithm.

- (4 pts) **6)** List four types of sockets available in OSs.
Stream socket, datagram socket, sequenced packet socket, reliably delivered message socket, raw socket.
- (4 pts) **7)** List the three layers employed in Unix NFS implementation.
System Call layer, virtual file system layer, NFS client
- (4 pts) **8)** List three optimization criteria employed by CPU scheduling algorithms.
Max average CPU utilization, Max average throughput, Min average turnaround time, Min average response time.
- (6 pts) **9)** List how deadlocks can be handled for each of the resource types listed below.
- (a)** Dedicated I/O devices:
 - (i) Deadlock prevention by resource ordering*
 - (ii) Deadlock avoidance through MAX information.*
 - (iii) Deadlock detection and process termination (rollback)*
 - (b)** Main memory:
Deadlock prevention by preemption.
 - (c)** System Files:
 - (i) Deadlock prevention using resource ordering.*
 - (ii) Deadlock detection and process termination (rollback).*
- (4 pts) **10)** What does Unix 4.3 BSD use as its CPU scheduling algorithm?
Priority-based scheduling with dynamically changing (increasing with time and decreasing with CPU usage) priorities. Higher priorities with system processes (negative), and lower priorities with user processes (positive).