Chapter 1 – 8 Essay Question Review

1. Explain why an operating system can be viewed as a resource allocator.

Ans: A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources. Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly.
Feedback: 1.1.2

2. Explain the purpose of an interrupt vector.

Ans: The interrupt vector is merely a table of pointers to specific interrupt-handling routines. Because there are a fixed number of interrupts, this table allows for more efficient handling of the interrupts than with a general-purpose, interrupt-processing routine.
Feedback: 1.2.1

3. What is a bootstrap program, and where is it stored?

Ans: A bootstrap program is the initial program that the computer runs when it is powered up or rebooted. It initializes all aspects of the system, from CPU registers to device controllers to memory contents. Typically, it is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term firmware, within the computer hardware.
Feedback: 1.2.1

4. What role do device controllers and device drivers play in a computer system?

Ans: A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus. Each device controller is in charge of a specific type of device. The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage. Typically, operating systems have a device driver for each device controller. This device driver understands the device controller and presents a uniform interface for the device to the rest of the operating system.
Feedback: 1.2.1

5. Why are clustered systems considered to provide high-availability service?

Ans: Clustered systems are considered high-availability in that these types of systems have redundancies capable of taking over a specific process or task in the case of a failure. The redundancies are inherent due to the fact that clustered systems are composed of two or more individual systems coupled together.
Feedback: 1.3.3

6. Describe the differences between physical, virtual, and logical memory.

Ans: Physical memory is the memory available for machines to execute operations (i.e., cache, random access memory, etc.). Virtual memory is a

method through which programs can be executed that requires space larger than that available in physical memory by using disk memory as a backing store for main memory. Logical memory is an abstraction of the computer's different types of memory that allows programmers and applications a simplified view of memory and frees them from concern over memory-storage limitations.
Feedback: 1.4

7. Describe the operating system's two modes of operation.

Ans: In order to ensure the proper execution of the operating system, most computer systems provide hardware support to distinguish between user mode and kernel mode. A mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfill the request.
Feedback: 1.5.1

8. Explain cache coherency.

Ans: In multiprocessor environments, two copies of the same data may reside in the local cache of each CPU. Whenever one CPU alters the data, the cache of the other CPU must receive an updated version of this data. Cache coherency involves ensuring that multiple caches store the most updated version of the stored data.
Feedback: 1.8.3

9. Why is main memory not suitable for permanent program storage or backup purposes? Furthermore, what is the main disadvantage to storing information on a magnetic disk drive as opposed to main memory?

Ans: Main memory is a volatile memory in that any power loss to the system will result in erasure of the data stored within that memory. While disk drives can store more information permanently than main memory, disk drives are significantly slower.
Feedback: 1.2

10. Describe the compute-server and file-server types of server systems.

Ans: The compute-server system provides an interface to which a client can send a request to perform an action (for example, read data from a database); in response, the server executes the action and sends back results to the client. The file-server system provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a Web server that delivers files to clients running Web browsers.
Feedback: 1.12.2

11. Computer systems can be divided into four approximate components. What are they?

Ans: Hardware, operating system, application programs, and users.

Feedback: 1.1

12. Distinguish between system and application programs.

Ans: System programs are not part of the kernel, but still are associated with the operating system. Application programs are not associated with the operating of the system.
Feedback: 1.1.3

13. Describe why direct memory access (DMA) is considered an efficient mechanism for performing I/O.

Ans: DMA is efficient for moving large amounts of data between I/O devices and main memory. It is considered efficient because it removes the CPU from being responsible for transferring data. DMA instructs the device controller to move data between the devices and main memory.
Feedback: 1.2.3

14. Describe why multi-core processing is more efficient than placing each processor on its own chip.

Ans: A large reason why it is more efficient is that communication between processors on the same chip is faster than processors on separate chips.
Feedback: 1.3.2

15. Distinguish between uniform memory access (UMA) and non-uniform memory access (NUMA) systems.

Ans: On UMA systems, accessing RAM takes the same amount of time from any CPU. On NUMA systems, accessing some parts of memory may take longer than accessing other parts of memory, thus creating a performance penalty for certain memory accesses.
Feedback: 1.3.2

16. There are two different ways that commands can be processed by a command interpreter. One way is to allow the command interpreter to contain the code needed to execute the command. The other way is to implement the commands through system programs. Compare and contrast the two approaches.

Ans: In the first approach, upon the user issuing a command, the interpreter jumps to the appropriate section of code, executes the command, and returns control back to the user. In the second approach, the interpreter loads the appropriate program into memory along with the appropriate arguments. The advantage of the first method is speed and overall simplicity. The disadvantage to this technique is that new commands require rewriting the interpreter program which, after a number of modifications, may get complicated, messy, or too large. The advantage to the second method is that new commands can be added without altering the command interpreter. The disadvantage is reduced speed and the clumsiness of passing parameters from the interpreter to the system program.
Feedback: 2.2

17. Describe the relationship between an API, the system-call interface, and the operating system.

Ans: The system-call interface of a programming language serves as a link to system calls made available by the operating system. This interface intercepts function calls in the API and invokes the necessary system call within the operating system. Thus, most of the details of the operating-system interface are hidden from the programmer by the API and are managed by the run-time support library.
Feedback: 2.3
Difficulty: Hard

18. Describe three general methods used to pass parameters to the operating system during system calls.

Ans: The simplest approach is to pass the parameters in registers. In some cases, there may be more parameters than registers. In these cases, the parameters are generally stored in a block, or table, of memory, and the address of the block is passed as a parameter in a register. Parameters can also be placed, or pushed, onto the stack by the program and popped off the stack by the operating system.
Feedback: 2.3

19. What are the advantages of using a higher-level language to implement an operating system?

Ans: The code can be written faster, is more compact, and is easier to understand and debug. In addition, improvements in compiler technology will improve the generated code for the entire operating system by simple recompilation. Finally, an operating system is far easier to port — to move to some other hardware — if it is written in a higher-level language.
Feedback: 2.6.3

20. Describe some requirements, or goals, when designing an operating system.

Ans: Requirements can be divided into user and system goals. Users desire a system that is convenient to use, easy to learn, and to use, reliable, safe, and fast. System goals are defined by those people who must design, create, maintain, and operate the system: The system should be easy to design, implement, and maintain; it should be flexible, reliable, error-free, and efficient.
Feedback: 2.6.1

21. What are the advantages and disadvantages of using a microkernel approach?

Ans: One benefit of the microkernel approach is ease of extending the operating system. All new services are added to user space and consequently do not require modification of the kernel. The microkernel also provides more security and reliability, since most services are running as user — rather than kernel — processes. Unfortunately, microkernels can suffer from performance decreases due to increased system function overhead.
Feedback: 2.7.3

22. Explain why a modular kernel may be the best of the current operating system design techniques.

Ans: The modular approach combines the benefits of both the layered and microkernel design techniques. In a modular design, the kernel needs only to have the capability to perform the required functions and know how to communicate between modules. However, if more functionality is required in the kernel, then the user can dynamically load modules into the kernel. The kernel can have sections with well-defined, protected interfaces, a desirable property found in layered systems. More flexibility can be achieved by allowing the modules to communicate with one another.
Feedback: 2.7.4

23. Describe two approaches to provide direct sharing of resources in a virtual-machine concept.

Ans: First, it is possible to share a minidisk, and thus to share files. This scheme is modeled after a physical shared disk, but is implemented by software. Second, it is possible to define a network of virtual machines, each of which can send information over the virtual communications network. Again, the network is modeled after physical communication networks, but is implemented in software.
Feedback: 2.8.2

24. In what ways does the JVM protect and manage memory?

Ans: After a class is loaded, the verifier checks that the .class file is valid Java bytecode and does not overflow or underflow the stack. It also ensures that the bytecode does not perform pointer arithmetic, which could provide illegal memory access. The JVM also automatically manages memory by performing garbage collection — the practice of reclaiming memory from objects no longer in use and returning it to the system.
Feedback: 2.8.6.2

25. What are two faster alternatives to implementing the JVM in software on top of a host operating system?

Ans: A faster software technique is to use a just-in-time (JIT) compiler. The first time a Java method is invoked, the bytecodes for the method are turned into native machine language for the host system, and then cached for subsequent invocations. A potentially faster technique is to run the JVM in hardware on a special Java chip that executes the Java bytecode operations as native code.
Feedback: 2.8.6.2

26. Distinguish between virtualization and simulation

Ans: Virtualization is the process whereby the system hardware is virtualized, thus providing the appearance to guest operating systems and applications that they are running on native hardware. In many virtualized environments, virtualization software runs at near native speeds. Simulation is the approach whereby the actual system is running on one set of hardware, but the guess system is compiled for a different set of hardware. Simulation software must simulate — or emulate — the hardware that the guest system is expecting. Because each instruction for

the guest system must be simulated in software rather than hardware, simulation is typically much slower than virtualization.
Feedback: 2.8.3

27. Describe Solaris 10 containers.

Ans: Solaris containers — or zones — provides a virtual layer between the operating system and its applications. Only one kernel is present and the hardware is not virtualized. However, the operating system and its devices are virtualized, providing processes within a container that they are the only running application on the system.
Feedback: 2.8.4

28. Name and describe the different states that a process can exist in at any given time.

Ans: The possible states of a process are: new, running, waiting, ready, and terminated. The process is created while in the new state. In the running or waiting state, the process is executing or waiting for an event to occur, respectively. The ready state occurs when the process is ready and waiting to be assigned to a processor and should not be confused with the waiting state mentioned earlier. After the process is finished executing its code, it enters the termination state.
Feedback: 3.1.2

29. Explain the main differences between a short-term and long-term scheduler.

Ans: The primary distinction between the two schedulers lies in the frequency of execution. The short-term scheduler is designed to frequently select a new process for the CPU, at least once every 100 milliseconds. Because of the short time between executions, the short-term scheduler must be fast. The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next. The long-term scheduler controls the degree of multiprogramming. Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.
Feedback: 3.2.2

30. Explain the difference between an I/O-bound process and a CPU-bound process.

Ans: The differences between the two types of processes stem from the number of I/O requests that the process generates. An I/O-bound process spends more of its time seeking I/O operations than doing computational work. The CPU-bound process infrequently requests I/O operations and spends more of its time performing computational work.
Feedback: 3.2.2

31. Explain the concept of a context switch.

Ans: Whenever the CPU starts executing a new process, the old process's state must be preserved. The context of a process is represented by its process control block. Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a

context switch occurs, the kernel saves the context of the old process in its PCB and loads the saves context of the new process scheduled to run.
Feedback: 3.2.3

32. Explain the fundamental differences between the UNIX fork() and Windows CreateProcess() functions.

Ans: Each function is used to create a child process. However, fork() has no parameters; CreateProcess() has ten. Furthermore, whereas the child process created with fork() inherits a copy of the address space of its parent, the CreateProcess() function requires specifying the address space of the child process.
Feedback: 3.3.1

33. Name the three types of sockets used in Java and the classes that implement them.

Ans: Connection-oriented (TCP) sockets are implemented with the Socket class. Connectionless (UDP) sockets use the DatagramSocket class. Finally, the MulticastSocket class is a subclass of the DatagramSocket class. A multicast socket allows data to be sent to multiple recipients.
Feedback: 3.6.1

34. What is a loopback and when is it used?

Ans: A loopback is a special IP address: 127.0.0.1. When a computer refers to IP address 127.0.0.1, it is referring to itself. When using sockets for client/server communication, this mechanism allows a client and server on the same host to communicate using the TCP/IP protocol.
Feedback: 3.6.1

35. Explain the purpose of external data representation (XDR).

Ans: Data can be represented differently on different machine architectures (e.g., little-endian vs. big-endian). XDR represents data independently of machine architecture. XDR is used when transmitting data between different machines using an RPC.
Feedback: 3.6.2

36. Explain the term marshalling.

Ans: Marshalling involves the packaging of parameters into a form that can be transmitted over the network. When the client invokes a remote procedure, the RPC system calls the appropriate stub, passing it the parameters provided to the remote procedure. This stub locates the port on the server and marshals the parameters. If necessary, return values are passed back to the client using the same technique.
Feedback: 3.6.2

37. Explain the terms "at most once" and "exactly once" and indicate how they relate to remote procedure calls.

Ans: Because a remote procedure call can fail in any number of ways, it is important to be able to handle such errors in the messaging system.

The term "at most once" refers to ensuring that the server processes a particular message sent by the client only once and not multiple times. This is implemented by merely checking the timestamp of the message. The term "exactly once" refers to making sure that the message is executed on the server once and only once so that there is a guarantee that the server received and processed the message.
Feedback: 3.6.2

38. Describe two approaches to the binding of client and server ports during RPC calls.

Ans: First, the binding information may be predetermined, in the form of fixed port addresses. At compile time, an RPC call has a fixed port number associated with it. Second, binding can be done dynamically by a rendezvous mechanism. Typically, an operating system provides a rendezvous daemon on a fixed RPC port. A client then sends a message containing the name of the RPC to the rendezvous daemon requesting the port address of the RPC it needs to execute. The port number is returned, and the RPC calls can be sent to that port until the process terminates (or the server crashes).
Feedback: 3.6.2

39. Why should a web server not run as a single-threaded process?

Ans: For a web server that runs as a single-threaded process, only one client can be serviced at a time. This could result in potentially enormous wait times for a busy server.
Feedback: 4.1.1

40. List the four major categories of the benefits of multithreaded programming. Briefly explain each.

Ans: The benefits of multithreaded programming fall into the categories: responsiveness, resource sharing, economy, and utilization of multiprocessor architectures. Responsiveness means that a multithreaded program can allow a program to run even if part of it is blocked. Resource sharing occurs when an application has several different threads of activity within the same address space. Threads share the resources of the process to which they belong. As a result, it is more economical to create new threads than new processes. Finally, a single-threaded process can only execute on one processor regardless of the number of processors actually present. Multiple threads can run on multiple processors, thereby increasing efficiency.
Feedback: 4.1.2

41. What are the two different ways in which a thread library could be implemented?

Ans: The first technique of implementing the library involves ensuring that all code and data structures for the library reside in user space with no kernel support. The other approach is to implement a kernel-level library supported directly by the operating system so that the code and data structures exist in kernel space.
Feedback: 4.3

42. Describe two techniques for creating Thread objects in Java.

Ans: One approach is to create a new class that is derived from the Thread class and to override its run() method. An alternative — and more commonly used — technique is to define a class that implements the Runnable interface. When a class implements Runnable, it must define a run() method. The code implementing the run() method is what runs as a separate thread.
Feedback: 4.3.3

43. In Java, what two things does calling the start() method for a new Thread object accomplish?

Ans: Calling the start() method for a new Thread object first allocates memory and initializes a new thread in the JVM. Next, it calls the run() method, making the thread eligible to be run by the JVM. Note that the run() method is never called directly. Rather, the start() method is called, which then calls the run() method.
Feedback: 4.3.3

44. Some UNIX systems have two versions of fork(). Describe the function of each version, as well as how to decide which version to use.

Ans: One version of fork() duplicates all threads and the other duplicates only the thread that invoked the fork() system call. Which of the two versions of fork() to use depends on the application. If exec() is called immediately after forking, then duplicating all threads is unnecessary, as the program specified in the parameters to exec() will replace the process. If, however, the separate process does not call exec() after forking, the separate process should duplicate all threads.
Feedback: 4.4.1

45. How can deferred cancellation ensure that thread termination occurs in an orderly manner as compared to asynchronous cancellation?

Ans: In asynchronous cancellation, the thread is immediately cancelled in response to a cancellation request. There is no insurance that it did not quit in the middle of a data update or other potentially dangerous situation. In deferred cancellation, the thread polls whether or not it should terminate. This way, the thread can be made to cancel at a convenient time.
Feedback: 4.4.2

46. What is a thread pool and why is it used?

Ans: A thread pool is a collection of threads, created at process startup, that sit and wait for work to be allocated to them. This allows one to place a bound on the number of concurrent threads associated with a process and reduce the overhead of creating new threads and destroying them at termination.
Feedback: 4.4.4

47. Describe the difference between the fork() and clone() Linux system calls.

Ans: The fork() system call is used to duplicate a process. The clone() system call behaves similarly except that, instead of creating a copy of

the process, it creates a separate process that shares the address space of the calling process.
Feedback: 4.5.2

48. Multicore systems present certain challenges for multithreaded programming. Briefly describe these challenges.

Ans: Multicore systems have placed more pressure on system programmers as well as application developers to make efficient use of the multiple computing cores. These challenges include determining how to divide applications into separate tasks that can run in parallel on the different cores. These tasks must be balanced such that each task is doing an equal amount of work. Just as tasks must be separated, data must also be divided so that it can be accessed by the tasks running on separate cores. So that data can safely be accessed, data dependencies must be identified and where such dependencies exist, data accesses must be synchronized to ensure the safety of the data. Once all such challenges have been met, there remains considerable challenges testing and debugging such applications.
Feedback: 4.1.3

49. Distinguish between coarse-grained and fine-grained multithreading.

Ans: There are two approaches to multithread a processor. (1) Coarse-grained multithreading allows a thread to run on a processor until a long-latency event, such as waiting for memory, to occur. When a long-latency event does occur, the processor switches to another thread. (2) Fine-grained multithreading switches between threads at a much finer-granularity, such as between instructions.
Feedback: 5.5.4

50. Explain the concept of a CPU–I/O burst cycle.

Ans: The lifecycle of a process can be considered to consist of a number of bursts belonging to two different states. All processes consist of CPU cycles and I/O operations. Therefore, a process can be modeled as switching between bursts of CPU execution and I/O wait.
Feedback: 5.1.1

51. What role does the dispatcher play in CPU scheduling?

Ans: The dispatcher gives control of the CPU to the process selected by the short-term scheduler. To perform this task, a context switch, a switch to user mode, and a jump to the proper location in the user program are all required. The dispatch should be made as fast as possible. The time lost to the dispatcher is termed dispatch latency.
Feedback: 5.1.4

52. Explain the difference between response time and turnaround time. These times are both used to measure the effectiveness of scheduling schemes.

Ans: Turnaround time is the sum of the periods that a process is spent waiting to get into memory, waiting in the ready queue, executing on the

CPU, and doing I/O. Turnaround time essentially measures the amount of time it takes to execute a process. Response time, on the other hand, is a measure of the time that elapses between a request and the first response produced.
Feedback: 5.2

53. What effect does the size of the time quantum have on the performance of an RR algorithm?

Ans: At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy. If the time quantum is extremely small, the RR approach is called processor sharing and creates the appearance that each of n processes has its own processor running at 1/n the speed of the real processor.
Feedback: 5.3.4

54. Explain the process of starvation and how aging can be used to prevent it.

Ans: Starvation occurs when a process is ready to run but is stuck waiting indefinitely for the CPU. This can be caused, for example, when higher-priority processes prevent low-priority processes from ever getting the CPU. Aging involves gradually increasing the priority of a process so that a process will eventually achieve a high enough priority to execute if it waited for a long enough period of time.
Feedback: 5.3.3

55. Explain the fundamental difference between asymmetric and symmetric multiprocessing.

Ans: In asymmetric multiprocessing, all scheduling decisions, I/O, and other system activities are handled by a single processor, whereas in SMP, each processor is self-scheduling.
Feedback: 5.5.1

56. Describe two general approaches to load balancing.

Ans: With push migration, a specific task periodically checks the load on each processor and — if it finds an imbalance—evenly distributes the load by moving processes from overloaded to idle or less-busy processors. Pull migration occurs when an idle processor pulls a waiting task from a busy processor. Push and pull migration are often implemented in parallel on load-balancing systems.
Feedback: 5.5.3

57. What is deterministic modeling and when is it useful in evaluating an algorithm?

Ans: Deterministic modeling takes a particular predetermined workload and defines the performance of each algorithm for that workload. Deterministic modeling is simple, fast, and gives exact numbers for comparison of algorithms. However, it requires exact numbers for input, and its answers apply only in those cases. The main uses of deterministic modeling are describing scheduling algorithms and providing examples to indicate trends.
Feedback: 5.7.1

58. Describe how trace tapes are used in distribution-driven simulations.

Ans: In a distribution-driven simulation, the frequency distribution indicates only how many instances of each event occur; it does not indicate anything about the order of their occurrence. Trace tapes can correct this problem. A trace tape is created to monitor the real system and record the sequence of actual events. This sequence then drives the simulation. Trace tapes provide an excellent way to compare two algorithms on exactly the same set of real inputs.
Feedback: 5.7.3

59. What three conditions must be satisfied in order to solve the critical section problem?

Ans: In a solution to the critical section problem, no thread may be executing in its critical section if a thread is currently executing in its critical section. Furthermore, only those threads that are not executing in their critical sections can participate in the decision on which process will enter its critical section next. Finally, a bound must exist on the number of times that other threads are allowed to enter their critical state after a thread has made a request to enter its critical state.
Feedback: 6.2

60. Explain two general approaches to handle critical sections in operating systems.

Ans: Critical sections may use preemptive or nonpreemptive kernels. A preemptive kernel allows a process to be preempted while it is running in kernel mode. A nonpreemptive kernel does not allow a process running in kernel mode to be preempted; a kernel-mode process will run until it exits kernel mode, blocks, or voluntarily yields control of the CPU. A nonpreemptive kernel is essentially free from race conditions on kernel data structures, as the contents of this register will be saved and restored by the interrupt handler.
Feedback: 6.2

61. Write two short functions that implement the simple semaphore wait() and signal() operations on global variable S.

Ans:
```
wait (S) {
  while (S <= 0);
  S--;
}

signal (S) {
  S++;
}
```
Feedback: 6.5

62. Explain the difference between the first readers–writers problem and the second readers–-writers problem.

Ans: The first readers-writers problem requires that no reader will be kept waiting unless a writer has already obtained permission to use the shared database; whereas the second readers-writers problem requires that

once a writer is ready, that writer performs its write as soon as possible.
Feedback: 6.6.2

63. Describe the dining-philosophers problem and how it relates to operating systems.

Ans: The scenario involves five philosophers sitting at a round table with a bowl of food and five chopsticks. Each chopstick sits between two adjacent philosophers. The philosophers are allowed to think and eat. Since two chopsticks are required for each philosopher to eat, and only five chopsticks exist at the table, no two adjacent philosophers may be eating at the same time. A scheduling problem arises as to who gets to eat at what time. This problem is similar to the problem of scheduling processes that require a limited number of resources.
Feedback: 6.6.3

64. How can write-ahead logging ensure atomicity despite the possibility of failures within a computer system?

Ans: Write-ahead logging records information describing all the modifications made by the transaction to the various data it accessed. Each log record describes a single operation of a transaction write. Upon a failure of the computer system, the log can be used to recover using both undo and redo procedures.
Feedback: 6.9

65. What overheads are reduced by the introduction of checkpoints in the log recovery system?

Ans: Whenever a system failure occurs, the log, if checkpoints are not included, must be consulted in its entirety to determine those transactions that need to be redone and undone. The entire log must be processed to make these determinations. This is time consuming. Most of the transactions that are redone are also not necessary due to idempotency. Checkpoints reduce both of these overheads.
Feedback: 6.9

66. Describe the turnstile structure used by Solaris for synchronization.

Ans: Solaris uses turnstiles to order the list of threads waiting to acquire either an adaptive mutex or a reader–writer lock. The turnstile is a queue structure containing threads blocked on a lock. Each synchronized object with at least one thread blocked on the object's lock requires a separate turnstile. However, rather than associating a turnstile with each synchronized object, Solaris gives each kernel thread its own turnstile.
Feedback: 6.8.1

67. Define the two-phase locking protocol.

Ans: This protocol ensures serializability through requiring that each transaction issue lock and unlock requests in two phases: a growing phase and a shrinking phase. In the growing phase, a transaction may obtain locks but not release a lock; whereas the shrinking phase allows the

release of locks but not the obtaining of new locks. Initially, a transaction is in the growing phase.
Feedback: 6.9

68. Describe how an adaptive mutex functions.

Ans: An adaptive mutex is used in the Solaris operating system to protect access to shared data. On a multiprocessor system, an adaptive mutex acts as a standard semaphore implemented as a spinlock. If the shared data being accessed is already locked and the thread holding that lock is running on another CPU, the thread spins while waiting for the lock to be released, and the data to become available. If the thread holding the lock is not in the run state, the waiting thread sleeps until the lock becomes available. On a single processor system, spinlocks are not used and the waiting thread always sleeps until the lock becomes available.
Feedback: 6.8.1

69. Describe a scenario when using a reader–writer lock is more appropriate than another synchronization tool such as a semaphore.

Ans: A tool such as a semaphore only allows one process to access shared data at a time. Reader–writer locks are useful when it is easy to distinguish if a process is only reading or reading/writing shared data. If a process is only reading shared data, it can access the shared data concurrently with other readers. In the case when there are several readers, a reader–writer lock may be much more efficient.
Feedback: 6.6.2

70. Explain what has to happen for a set of processes to achieve a deadlocked state.

Ans: For a set of processes to exist in a deadlocked state, every process in the set must be waiting for an event that can be caused only be another process in the set. Thus, the processes cannot ever exit this state without manual intervention.
Feedback: 7.1

71. Describe the four conditions that must hold simultaneously in a system if a deadlock is to occur.

Ans: For a set of processes to be deadlocked: at least one resource must remain in a nonsharable mode, a process must hold at least one resource and be waiting to acquire additional resources held by other processes, resources in the system cannot be preempted, and a circular wait has to exist between processes.
Feedback: 7.2.1

72. What are the three general ways that a deadlock can be handled?

Ans: A deadlock can be prevented by using protocols to ensure that a deadlock will never occur. A system may allow a deadlock to occur, detect it, and recover from it. Lastly, an operating system may just ignore the problem and pretend that deadlocks can never occur.
Feedback: 7.3

73. What is the difference between deadlock prevention and deadlock avoidance?

Ans: Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions for deadlock cannot hold. Deadlock avoidance requires that the operating system be given, in advance, additional information concerning which resources a process will request and use during its lifetime.
Feedback: 7.4

74. Describe two protocols to ensure that the hold-and-wait condition never occurs in a system.

Ans: One protocol requires each process to request and be allocated all its resources before it begins execution. We can implement this provision by requiring that system calls requesting resources for a process precede all other system calls. An alternative protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, however, it must release all the resources that it is currently allocated.
Feedback: 7.4.2

75. What is one way to ensure that a circular-wait condition does not occur?

Ans: One way to ensure that this condition never holds is to impose a total ordering of all resource types, and to require that each process requests resources in an increasing order of enumeration. This can be accomplished by assigning each resource type a unique integer number to determine whether one precedes another in the ordering.
Feedback: 7.4.4

76. What does a claim edge signify in a resource-allocation graph?

Ans: A claim edge indicates that a process may request a resource at some time in the future. This edge resembles a request edge in direction, but is represented in the graph by a dashed line.
Feedback: 7.5.2

77. Describe a wait-for graph and how it detects deadlock.

Ans: If all resources have only a single instance, then we can define a deadlock-detection algorithm that uses a variant of the resource-allocation graph, called a wait-for graph. We obtain this graph from the resource-allocation graph by removing the resource nodes and collapsing the appropriate edges. To detect deadlocks, the system needs to maintain the wait-for graph and periodically invoke an algorithm that searches for a cycle in the graph.
Feedback: 7.6.1

78. What factors influence the decision of when to invoke a detection algorithm?

Ans: The first factor is how often a deadlock is likely to occur; if deadlocks occur frequently, the detection algorithm should be invoked frequently. The second factor is how many processes will be affected by deadlock when it happens; if the deadlock-detection algorithm is invoked

for every resource request, a considerable overhead in computation time will be incurred.
Feedback: 7.6.3

79. Describe two methods for eliminating processes by aborting a process.

Ans: The first method is to abort all deadlocked processes. Aborting all deadlocked processes will clearly break the deadlock cycle; however, the deadlocked processes may have to be computed for a long time, and results of these partial computations must be discarded and will probably have to be recomputed later. The second method is to abort one process at a time until the deadlock cycle is eliminated. Aborting one process at a time incurs considerable overhead, since, after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.
Feedback: 7.7.1

80. Name three issues that need to be addressed if a preemption is required to deal with deadlocks.

Ans: First, the order of resources and processes that need to be preempted must be determined to minimize cost. Second, if a resource is preempted from a process, the process must be rolled back to some safe state and restarted from that state. The simplest solution is a total rollback. Finally, we must ensure that starvation does not occur from always preempting resources from the same process.
Feedback: 7.7.2

81. Describe how a safe state ensures deadlock will be avoided.

Ans: A safe state ensures that there is a sequence of processes to finish their program execution. Deadlock is not possible while the system is in a safe state. However, if a system goes from a safe state to an unsafe state, deadlock is possible. One technique for avoiding deadlock is to ensure that the system always stays in a safe state. This can be done by only assigning a resource as long as it maintains the system in a safe state.
Feedback: 7.5.1

82. What is the advantage of using dynamic loading?

Ans: With dynamic loading a program does not have to be stored, in its entirety, in main memory. This allows the system to obtain better memory-space utilization. This also allows unused routines to stay out of main memory so that memory can be used more effectively. For example, code used to handle an obscure error would not always use up main memory.
Feedback: 8.1.4

83. What is the context switch time, associated with swapping, if a disk drive with a transfer rate of 2 MB/s is used to swap out part of a program that is 200 KB in size? Assume that no seeks are necessary and that the average latency is 15 ms. The time should reflect only the amount of time necessary to swap out the process.

Ans: 200KB / 2048 KB per second + 15 ms = 113 ms

Feedback: 8.2

84. When does external fragmentation occur?

Ans: As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request, but the available spaces are not contiguous; storage is fragmented into a large number of small holes. Both the first-fit and best-fit strategies for memory allocation suffer from external fragmentation.
Feedback: 8.3.3

85. Distinguish between internal and external fragmentation.

Ans: Fragmentation occurs when memory is allocated and returned to the system. As this occurs, free memory is broken up into small chunks, often too small to be useful. External fragmentation occurs when there is sufficient total free memory to satisfy a memory request, yet the memory is not contiguous, so it cannot be assigned. Some contiguous allocation schemes may assign a process more memory than it actually requested (i.e. they may assign memory in fixed-block sizes). Internal fragmentation occurs when a process is assigned more memory than it has requested and the wasted memory fragment is internal to a process.
Feedback: 8.3.3

86. Explain the basic method for implementing paging.

Ans: Physical memory is broken up into fixed-sized blocks called frames while logical memory is broken up into equal-sized blocks called pages. Whenever the CPU generates a logical address, the page number and offset into that page is used, in conjunction with a page table, to map the request to a location in physical memory.
Feedback: 8.4

87. Describe how a transaction look-aside buffer (TLB) assists in the translation of a logical address to a physical address.

Ans: Typically, large page tables are stored in main memory, and a page-table base register points are saved to the page table. Therefore, two memory accesses are needed to access a byte (one for the page-table entry, one for the byte), causing memory access to be slowed by a factor of 2. The standard solution to this problem is to use a TLB, a special, small fast-lookup hardware cache. The TLB is associative, high speed memory. Each entry consists of a key and value. An item is compared with all keys simultaneously, and if the item is found, the corresponding value is returned.
Feedback: 8.4.2

88. How are illegal page addresses recognized and trapped by the operating system?

Ans: Illegal addresses are trapped by the use of a valid-invalid bit, which is generally attached to each entry in the page table. When this bit is set to "valid," the associated page is in the process's logical address space and is thus a legal (or valid) page. When the bit is set to

"invalid," the page is not in the process's logical address space. The operating system sets this bit for each page to allow or disallow access to the page.
Feedback: 8.4.3

89. Describe the elements of a hashed page table.

Ans: A hashed page table contains hash values which correspond to a virtual page number. Each entry in the hash table contains a linked list of elements that hash to the same location (to handle collisions). Each element consists of three fields: (1) the virtual page number, (2) the value of the mapped page frame, and (3) a pointer to the next element in the linked list.
Feedback: 8.5.2

90. Briefly describe the segmentation memory management scheme. How does it differ from the paging memory management scheme in terms of the user's view of memory?

Ans: Segmentation views a logical address as a collection of segments. Each segment has a name and length. The addresses specify both the segment name and the offset within the segment. The user therefore specifies each address by two quantities: a segment name and an offset. In contrast, in a paging scheme, the user specifies a single address, which is partitioned by the hardware into a page number and an offset, all invisible to the programmer.
Feedback: 8.6

91. Describe the partitions in a logical-address space of a process in a Pentium architecture.

Ans: The logical-address space is divided into two partitions. The first partition consists of up to 8 KB segments that are private to that process. The second partition consists of up to 8 KB segments that are shared among all the processes. Information about the first partition is kept in the local descriptor table (LDT); information about the second partition is kept in the global descriptor table (GDT).
Feedback: 8.7

92. How is a limit register used for protecting main memory?

Ans: When the CPU is executing a process, it generates a logical memory address that is added to a relocation register in order to arrive at the physical memory address actually used by main memory. A limit register holds the maximum logical address that the CPU should be able to access. If any logical address is greater than or equal to the value in the limit register, then the logical address is a dangerous address and an error results.
Feedback: 8.1.1

93. Using Figure 8.11, describe how a logical address is translated to a physical address.

Ans: A logical address is generated by the CPU. This logical address consists of a page number and offset. The TLB is first checked to see if the page number is present. If so, a TLB hit, the corresponding page frame is extracted from the TLB, thus producing the physical address. In

the case of a TLB miss, the page table must be searched according to page number for the corresponding page frame.
Feedback: 8.4