CSIS 2260 Quiz 1 - Quiz Review

Operating Systems (Douglas College)

CSIS 2260

1. The _____ is the interface that is the boundary between hardware and software.
   ISA
2. A(n) _____ is a set of resources for the movement, storage, and processing of data and for the control of these functions.
   Computer
3. The operating system's _____ refers to its inherent flexibility in permitting functional modifications to the system without interfering with service.
   ability to evolve
4. Operating systems must evolve over time because:
   new hardware is designed and implemented in the computer system
5. A special type of programming language used to provide instructions to the monitor is _____ .
   JCL
6. Hardware features desirable in a batch-processing operating system include memory protection, timer, privileged instructions, and _____ .
   Interrupts
7. A user program executes in a _____ , in which certain areas of memory are protected from the user's use, and in which certain instructions may not be executed.
   user mode
8. Multiprogramming operating systems are fairly sophisticated compared to single-program or _____ systems.
   Uniprogramming
9. One of the first time-sharing operating systems to be developed was the _____ .
   Compatible Time-Sharing System
10. The technique where a system clock generates interrupts, and at each clock interrupt the OS regains control and assigns the processor to another user, is _____ .
    Time slicing
11. An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.
    True
12. The OS masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.
    True
13. The ABI gives a program access to the hardware resources and services available in the system through the user ISA.
    False
14. The OS frequently relinquishes control and must depend on the processor to allow it to regain control.
    TRUE
15. On of the driving forces in operating system evolution is advancement in the underlying hardware technology.

TRUE

16. The processor itself is not a resource so the OS is not involved in determining how much of the processor time is devoted to the execution of a user program.
    FALSE
17. A process consists of three components: an executable program, the associated data needed by the program, and the execution context of the program.
    TRUE
18. Uniprogramming typically provides better utilization of system resources than multiprogramming.
    FALSE
19. A monolithic kernel is implemented as a single process with all elements sharing the same address space.
    TRUE
20. The user has direct access to the processor with batch-processing type OS.
    FALSE
21. The processor controls the operation of the computer and performs its data processing functions
    TRUE
22. It is possible for a communications interrupt to occur while a printer interrupt is being processed.
    TRUE
23. A system bus does not transfer data between the computer and its external environment.
    TRUE
24. Cache memory is invisible to the OS.
    TRUE
25. With interrupts, the processor can not be engaged in executing other instructions while an I/O operation is in progress.
    FALSE
26. Digital Signal Processors deal with streaming signals such as audio and video.
    TRUE
27. The fetched instruction is loaded into the Program Counter.
    FALSE
28. Interrupts are provided primarily as a way to improve processor utilization.
    TRUE
29. The interrupt can occur at any time and therefore at any point in the execution of a user program.
    TRUE
30. Over the years memory access speed has consistently increased more rapidly than processor speed.
    FALSE
31. The four main structural elements of a computer system are:
    Processor, Main Memory, I/O Modules and System Bus
32. The _____ holds the address of the next instruction to be fetched.
    Program Counter (PC)
33. The _____ contains the data to be written into memory and receives the data read from memory.
    memory buffer register
34. Instruction processing consists of two steps:

fetch and execute

35. The _____ routine determines the nature of the interrupt and performs whatever actions are needed.
    interrupt handler
36. The unit of data exchanged between cache and main memory is _____ .
    block size
37. The _____ chooses which block to replace when a new block is to be loaded into the cache and the cache already has all slots filled with other blocks.
    replacement algorithm
38. _____ is more efficient than interrupt-driven or programmed I/O for a multiple-word I/O transfer.
    Direct memory access
39. The _____ is a point-to-point link electrical interconnect specification that enables high-speed communications among connected processor chips.
    QPI
40. Small, fast memory located between the processor and main memory is called:
    Cache memory
41. The processor controls the operation of the computer and performs its data processing functions.
    True
42. It is not possible for a communications interrupt to occur while a printer interrupt is being processed.
    False
43. A system bus transfers data between the computer and its external environment.
    False
44. Cache memory is invisible to the OS.
    True
45. With interrupts, the processor can not be engaged in executing other instructions while an I/O operation is in progress. ,
    False
46. Digital Signal Processors deal with streaming signals such as audio and video.
    True
47. The fetched instruction is loaded into the Program Counter.
    False
48. Interrupts are provided primarily as a way to improve processor utilization.
    True
49. The interrupt can occur at any time and therefore at any point in the execution of a user program.
    true
50. Over the years memory access speed has consistently increased more rapidly than processor speed.
    false
51. What is the purpose of system calls?
    Answer: System calls allow user-level processes to request services of the operating system.
52. What are the five major activities of an operating system with regard to process management?
    The five major activities are:

a. The creation and deletion of both user and system processes
b. The suspension and resumption of processes
c. The provision of mechanisms for process synchronization
d. The provision of mechanisms for process communication
e. The provision of mechanisms for deadlock handling

53. What are the three major activities of an operating system with regard to memory management?

Answer: The three major activities are:

a. Keep track of which parts of memory are currently being used and by whom.

b. Decide which processes are to be loaded into memory when memory space becomes available.

c. Allocate and deallocate memory space as needed.

54. What are the three major activities of an operating system with regard to secondary-storage management?

a. Free-space management.

b. Storage allocation.

c. Disk scheduling

55. What is the purpose of the command interpreter? Why is it usually separate from the kernel?

It reads commands from the user or from a file of commands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes.

56. What system calls have to be executed by a command interpreter or shell in order to start a new process?

In Unix systems, a fork system call followed by an exec system call need to be performed to start a new process. The fork call clones the currently executing process, while the exec call overlays a new process based on a different executable over the calling process.

57. What is the purpose of system programs?

System programs can be thought of as bundles of useful system calls. They provide basic functionality to users so that users do not need to write their own programs to solve common problems.

58. What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?

As in all cases of modular design, designing an operating system in a modular way has several advantages. The system is easier to debug and modify because changes affect only limited sections of the system rather than touching all sections of the operating system. Information is kept only where it is needed and is accessible only within a defined and restricted area, so any bugs affecting that data must be limited to a specific module or layer.

59. List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer

The five services are:

a. Program execution. The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

b. I/O operations. Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device- or controller-specific commands. User-level programs cannot be trusted to access only devices they Practice Exercises 7 should have access to and to access them only when they are otherwise unused.

c. File-system manipulation. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

d. Communications. Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

e. Error detection. Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently processindependent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system

60. Why do some systems store the operating system in firmware, while others store it on disk?

For certain devices, such as handheld PDAs and cellular telephones, a disk with a file system may be not be available for the device. In this situation, the operating system must be stored in firmware.

61. How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?

Consider a system that would like to run both Windows XP and three different distributions of Linux (e.g., RedHat, Debian, and Mandrake). Each operating system will be stored on disk. During system boot-up, a special program (which we will call the boot manager) will determine which operating system to boot into. This means that rather initially booting to an operating system, the boot manager will first run

during system startup. It is this boot manager that is responsible for determining which system to boot into. Typically boot managers must be stored at certain locations of the hard disk to be recognized during system startup. Boot managers often provide the user with a selection of systems to boot into; boot managers are also typically designed to boot into a default operating system if no choice is selected by the user.

62. What are the three main purposes of an operating system?

Answer: The three main puropses are:

• To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.

• To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.

• As a control program it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.

63. What are the main differences between operating systems for mainframe computers and personal computers?

Answer: Generally, operating systems for batch systems have simpler requirements than for personal computers. Batch systems do not have to be concerned with interacting with a user as much as a personal computer. As a result, an operating system for a PC must be concerned with response time for an interactive user. Batch systems do not have such requirements. A pure batch system also may have not to handle time sharing, whereas an operating system must switch rapidly between different jobs.

64. List the four steps that are necessary to run a program on a completely dedicated machine—a computer that is running only that program.

Answer: The four steps are:

a. Reserve machine time.

b. Manually load program into memory.

c. Load starting address and begin execution.

d. Monitor and control execution of program from console.

65. We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?

Answer: Single-user systems should maximize use of the system for the user. A GUI might “waste” CPU cycles, but it optimizes the user’s interaction with the system.

66. We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?

Answer: Single-user systems should maximize use of the system for the user. A GUI might “waste” CPU cycles, but it optimizes the user’s interaction with the system.

67. What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?

Answer: The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time frame, it may cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don’t allow response time to exceed the time constraint.

68. Consider the various definitions of operating system. Next, consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not, and support your answers.

Answer: Point. Applications such as web browsers and email tools are performing an increasingly important role in modern desktop computer systems. To fulfill this role, they should be incorporated as part of the operating system. By doing so, they can provide better performance and better integration with the rest of the system. In addition, these important applications can have the same look-and-feel as the operating system software.

Counterpoint. The fundamental role of the operating system is to manage system resources such as the CPU, memory, I/O devices, etc. In addition, it's role is to run software applications such as web browsers and email applications. By incorporating such applications into the operating system, we burden the operating system with additional functionality. Such a burden may result in the operating system performing a less-thansatisfactory job at managing system resources. In addition, we increase the size of the operating system thereby increasing the likelihood of system crashes and security violations.

69. How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?

Answer: The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices could be accessed only when the program is executing in kernel mode. Control over when interrupts could be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.

70. Which of the following instructions should be privileged?

a. Set value of timer.

b. Read the clock.

c. Clear memory.

d. Issue a trap instruction.

e. Turn off interrupts.

f. Modify entries in device-status table.

g. Switch from user to kernel mode.

h. Access I/O device.

Answer: The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.

71. Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

Answer: The data required by the operating system (passwords, access controls, accounting information, and so on) would have to be stored in or passed through unprotected memory and thus be accessible to unauthorized users.

72. Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?

Answer: Although most systems only distinguish between user and kernel modes, some CPUs have supported multiple modes. Multiple modes could be used to provide a finer-grained security policy. For example, rather than distinguishing between just user and kernel mode, you could distinguish between different types of user mode. Perhaps users belonging to the same group could execute each other's code. The machine would go into a specified mode when one of these users was running code. When the machine was in this mode, a member of the group could run code belonging to anyone else in the group.

Another possibility would be to provide different distinctions within kernel code. For example, a specific mode could allow USB device drivers

to run. This would mean that USB devices could be serviced without having to switch to kernel mode, thereby essentially allowing USB device drivers to run in a quasi-user/kernel mode.

73. Timers could be used to compute the current time. Provide a short description of how this could be accomplished.

Answer: A program could use the following approach to compute the current time using timer interrupts. The program could set a timer for some time in the future and go to sleep. When it is awakened by the interrupt, it could update its local state, which it is using to keep track of the number of interrupts it has received thus far. It could then repeat this process of continually setting timer interrupts and updating its local state when the interrupts are actually raised.

74. Is the Internet a LAN or a WAN?

Answer: The Internet is a WAN as the various computers are located at geographically different places and are connected by long-distance network links.

75. 1.1 What Operating Systems Do pg32

76. 1.2 Computer-System Organization pg 35

77. 1.3 Computer-System Architecture pg 44

78. 1.4 Operating-Systems Operations pg 49

79. 1.5 Resource Management pg 55

80. 2.1 Operating-Systems Services pg86

81. 2.2 User and Operating System Interface pg88

82. 2.3 System Calls pg93

83. 2.7 Operating -System Design and Implementation pg104

84. 2.8 Operating-System Structure pg109

85. 2.9.2 System Boot - how does the hardware know where the kernel is or how to load the kernel - page 124

On most systems, the boot process proceeds as follows

1.A small piece of code known as the bootstrap program or boot loader

locates the kernel.

2. The kernel is loaded into memory and started.

3. The kernel initializes hardware.

4. The root file system is mounted.

86. What is the BIOS-base bootstrap?

Some computer systems use a multistage boot process: When the computer

is first powered on, a small boot loader located in nonvolatile firmware known

as BIOS is run.

87. What is UEFI - compare this to BIOS ?

Many recent computer systems have replaced the BIOS-based boot process with UEFI (Unified Extensible Firmware Interface). UEFI has several advantages over BIOS, including better support for 64-bit systems and larger disks. Perhaps the greatest advantage is that UEFI is a single, complete boot manager and therefore is faster than the multistage BIOS boot process.

88. How does the boot process work for mobile systems?

The boot process for mobile systems is slightly different from that for traditional PCs. For example, although its kernel is Linux-based, Android does not use GRUB and instead leaves it up to vendors to provide boot loaders. The most common Android boot loader is LK (for “little kernel”). Android systems use the same compressed kernel image as Linux, as well as an initial RAM file system. However, whereas Linux discards the initramfs once all necessary drivers have been loaded, Android maintains initramfs as the root file system for the device. Once the kernel has been loaded and the root file system mounted, Android starts the init process and creates a number of services before displaying the home screen.

89. The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories, and discuss how they differ

Answer: One class of services provided by an operating system is to enforce protection between different processes running concurrently in the system. Processes
are allowed to access only those memory locations that are associated with their
address spaces. Also, processes are not allowed to corrupt files associated with other

users. A process is also not allowed to access devices directly without operating
system intervention. The second class of services provided by an operating system is
to provide new functionality that is not supported directly by the underlying hardware.
Virtual memory and file systems are two such examples of new services provided by
an operating system.

90. Describe three general methods for passing parameters to the operating system.
Answer:
a. Pass parameters in registers
b. Registers pass starting addresses of blocks of parameters
c. Parameters can be placed, or pushed, onto the stack by the program, and popped off
the stack by the operating system.

91. Describe how you could obtain a statistical profile of the amount of time spent by
a program executing different sections of its code. Discuss the importance of obtaining
such a statistical profile.
Answer: One could issue periodic timer interrupts and monitor what instructions or
what sections of code are currently executing when the interrupts are delivered. A
statistical profile of which pieces of code were active should be consistent with the
time spent by the program in different sections of its code. Once such a statistical
profile has been obtained, the programmer could optimize those sections of code that
are consuming more of the CPU resources.

92. What are the five major activities of an operating system in regard to file management?
Answer:
• The creation and deletion of files
• The creation and deletion of directories
• The support of primitives for manipulating files and directories
• The mapping of files onto secondary storage
• The backup of files on stable (nonvolatile) storage media

93. What are the advantages and disadvantages of using the same system call
interface for manipulating both files and devices?
Answer: Each device can be accessed as though it was a file in the file system. Since
most of the kernel deals with devices through this file interface, it is relatively easy to
add a new device driver by implementing the hardware-specific code to support this
abstract file interface. Therefore, this benefits the development of both user program
code, which can be written to access devices and files in the same manner, and device
driver code, which can be written to support a well-defined API. The disadvantage
with using the same interface is that it might be difficult to capture the functionality of
certain devices within the context of the file access API, thereby either resulting in a
loss of functionality or a loss of performance. Some of this could be overcome by the
use of ioctl operation that provides a general purpose interface for processes to invoke
operations on devices.

94. What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?
Answer: The two models of interprocess communication are message passing model and the shared-memory model. The message-passing model controls the

95. Why is the separation of mechanism and policy desirable?
Answer: Mechanism and policy must be separate to ensure that systems are easy to
modify. No two system installations are the same, so each installation may want to
tune the operating system to suit its needs. With mechanism and policy separate, the
policy may be changed at will while the mechanism stays unchanged. This arrangement provides a more flexible system.

96. It is sometimes difficult to achieve a layered approach if two components of the
operating system are dependent on each other. Identify a scenario in which it is

unclear how to layer two system components that require tight coupling of their
functionalities.
Answer: The virtual memory subsystem and the storage subsystem are typically
tightly-coupled and requires careful design in a layered system due to the following
interactions. Many systems allow files to be mapped into the virtual memory space of
an executing process. On the other hand, the virtual memory subsystem typically uses
the storage system to provide the backing store for pages that do not currently reside in
memory. Also, updates to the file system are sometimes buffered in physical memory
before it is flushed to disk, thereby requiring careful coordination of the usage of
memory between the virtual memory subsystem and the file system.

97. What is the main advantage of the microkernel approach to system design? How
do user programs and system services interact in a microkernel architecture? What are
the disadvantages of using the microkernel approach?
Answer: Benefits typically include the following (a) adding a new service does not
require modifying the kernel, (b) it is more secure as more operations are done in user
mode than in kernel mode, and (c) a simpler kernel design and functionality typically
results in a more reliable operating system. User programs and system services interact
in a microkernel architecture by using interprocess communication mechanisms such
asmessaging. These messages are conveyed by the operating system. The primary
disadvantage of the microkernel architecture are the overheads associated with
interprocess communication and the frequent use of the operating system’s messaging
functions in order to enable the user process and the system service to interact with
each other.

98. In what ways is the modular kernel approach similar to the layered approach? In

what ways does it differ from the layered approach?
Answer: The modular kernel approach requires subsystems to interact with each other
through carefully constructed interfaces that are typically narrow (in terms of the
functionality that is exposed to external modules). The layered kernel approach is
similar in that respect. However, the layered kernel imposes a strict ordering of
subsystems such that subsystems at the lower layers are not allowed to invoke
operations corresponding to the upper-layer subsystems. There are no such restrictions
in the modular-kernel approach, wherein modules are free to invoke each other
without any constraints.

99. What is the main advantage for an operating-system designer of using virtualmachine architecture? What is the main advantage for a user?
Answer: The system is easy to debug, and security problems are easy to solve. Virtual
machines also provide a good platform for operating system research since many
different operating systems may run on one physical system.

100. Why is a just-in-time compiler useful for executing Java programs?
Answer: Java is an interpreted language. This means that the JVM interprets the
bytecode instructions one at a time. Typically, most interpreted environments are
slower than running native binaries, for the interpretation process requires converting
each instruction into native machine code. A just-in-time (JIT) compiler compiles the
bytecode for a method into native machine code the first time the method is encountered. This means that the Java program is essentially running as a native
application (of course, the conversion process of the JIT takes time as well but not as
much as bytecode interpretation.) Furthermore, the JIT caches compiled code so that it
may be reused the next time the method is encountered. A Java program that is run by
a JIT rather than a traditional interpreter typically runs much faster.

101. What is the relationship between a guest operating system and a host operating system in a system like VMware? What factors need to be considered in choosing the host operating system?
Answer: A guest operating system provides its services by mapping them onto the functionality provided by the host operating system. A key issue that needs to be considered in choosing the host operating system is whether it is sufficiently general in terms of its system-call interface in order to be able to support the functionality associated with the guest operating system.

102. The experimental Synthesis operating system has an assembler incorporated within the kernel. To optimize system-call performance, the kernel assembles routines within kernel space to minimize the path that the system call must take through the kernel. This approach is the antithesis of the layered approach, in which the path through the kernel is extended to make building the operating system easier. Discuss the pros and cons of the Synthesis approach to kernel design and to systemperformance optimization.
Answer: Synthesis is impressive due to the performance it achieves through on-the-fly compilation. Unfortunately, it is difficult to debug problems within the kernel due to the fluidity of the code. Also, such compilation is system specific; making Synthesis difficult to port (a new compiler must be written for each architecture).
1.1 What are the three main purposes of an operating system?
Answer:
The three main puropses are:
• To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.
• To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.
• As a control program it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.

1.2 We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?
Answer:
Single-user systems should maximize use of the system for the user. A GUI might “waste” CPU cycles, but it optimizes the user’s interaction with the system.

1.3 What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?
Answer:
The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time frame, it may cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don’t allow response time to exceed the time constraint.

1.4 Keeping in mind the various definitions of operating system, consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not, and support your answers.
Answer:
An argument in favor of including popular applications with the operating system is that if the application is embedded within the operating system, it is likely to be better able to take advantage of features in the kernel and therefore have performance advantages over an application that runs outside of the kernel.
Arguments against embedding applications within the operating system typically dominate
however:
(1) the applications are applications - and not part of an operating system,
(2) any performance benefits of running within the kernel are offset by security vulnerabilities,
(3) it leads to a bloated operating system.

1.5 How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?
Answer:
The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices could be accessed only when the program is executing in

kernel mode. Control over when interrupts could be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.

1.6 Which of the following instructions should be privileged?
a. Set value of timer.
b. Read the clock.
c. Clear memory.
d. Issue a trap instruction.
e. Turn off interrupts.
f. Modify entries in device-status table.
g. Switch from user to kernel mode.
h. Access I/O device.
Answer:
The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.

1.7 Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.
Answer:
The data required by the operating system (passwords, access controls, accounting information, and so on) would have to be stored in or passed through unprotected memory and thus be accessible to unauthorized users.

1.8 Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?
Answer:
Although most systems only distinguish between user and kernel modes, some CPUs have supported multiple modes. Multiple modes could be used to provide a finer-grained security policy. For example, rather than distinguishing between just user and kernel mode, you could distinguish between different types of user mode. Perhaps users belonging to the same group could execute each other’s code. The machine would go into a specified mode when one of these users was running code. When the machine was in this mode, a member of the group could run code belonging to anyone else in the group.
Another possibility would be to provide different distinctions within kernel code. For example, a specific mode could allow USB device drivers to run. This would mean that USB devices could be serviced without

having to switch to kernel mode, thereby essentially allowing USB device drivers to run in a quasi-user/kernel mode.

1.9 Timers could be used to compute the current time. Provide a short description of how this could be accomplished.
Answer:
A program could use the following approach to compute the current time using timer interrupts. The program could set a timer for some time in the future and go to sleep. When it is awakened by the interrupt, it could update its local state, which it is using to keep track of the number of interrupts it has received thus far. It could then repeat this process of continually setting timer interrupts and updating its local state when the interrupts are actually raised.

1.10 Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?
Answer:
Caches are useful when two or more components need to exchange data, and the components perform transfers at differing speeds. Caches solve the transfer problem by providing a buffer of intermediate speed between the components. If the fast device finds the data it needs in the cache, it need not wait for the slower device. The data in the cache must be kept consistent with the data in the components. If a component has a data value change, and the datum is also in the cache, the cache must also be updated. This is especially a problem on multiprocessor systems where more than one process may be accessing a datum. A component may be eliminated by an equal-sized cache, but only if: (a) the cache and the component have equivalent state-saving capacity (that is, if the component retains its data when electricity is removed, the cache must retain data as well), and (b) the cache is affordable, because faster storage tends to be more expensive.

1.11 Distinguish between the client–server and peer-to-peer models of distributed systems.
Answer:
The client-server model firmly distinguishes the roles of the client and server. Under this model, the client requests services that are provided by the server. The peer-to-peer model doesn’t have such strict roles. In fact, all nodes in the system are considered peers and thus may act as either clients or servers—or both. A node may request a service from another peer, or the node may in fact provide such a service to other peers in the system.
For example, let’s consider a system of nodes that share cooking recipes. Under the client-server model, all recipes are stored with the

server. If a client wishes to access a recipe, it must request the recipe from the specified server. Using the peer-to-peer model, a peer node could ask other peer nodes for the specified recipe. The node (or perhaps nodes) with the requested recipe could provide it to the requesting node. Notice how each peer may act as both a client (it may request recipes) and as a server (it may provide recipes).

1.12 In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.
a. What are two such problems?
b. Can we ensure the same degree of security in a time-shared machine as in a dedicated machine? Explain your answer.
Answer:
a. Stealing or copying one's programs or data; using system resources (CPU, memory, disk space, peripherals) without proper accounting.
b. Probably not, since any protection scheme devised by humans can inevitably be broken by a human, and the more complex the scheme, the more difficult it is to feel confident of its correct implementation.

1.13 The issue of resource utilization shows up in different forms in different types of operating systems. List what resources must be managed carefully in the following settings:
a. Mainframe or minicomputer systems
b. Workstations connected to servers
c. Mobile computers
Answer:
a. Mainframes: memory and CPU resources, storage, network bandwidth
b. Workstations: memory and CPU resources
c. Mobile computers: power consumption, memory resources

1.14 Under what circumstances would a user be better off using a timesharing system rather than a PC or a single-user workstation?
Answer:
When there are few other users, the task is large, and the hardware is fast, time-sharing makes sense. The full power of the system can be brought to bear on the user's problem. The problem can be solved faster than on a personal computer. Another case occurs when lots of other users need resources at the same time.
A personal computer is best when the job is small enough to be executed reasonably on it and when performance is sufficient to execute the program to the user's satisfaction.

1.15 Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems?
Answer:
Symmetric multiprocessing treats all processors as equals, and I/O can be processed on any CPU. Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves. The master distributes tasks among the slaves, and I/O is usually done by the master only. Multiprocessors can save money by not duplicating power supplies, housings, and peripherals. They can execute programs more quickly and can have increased reliability. They are also more complex in both hardware and software than uniprocessor systems.

1.16 How do clustered systems differ from multiprocessor systems? What is required for two machines belonging to a cluster to cooperate to provide a highly available service?
Answer:
Clustered systems are typically constructed by combining multiple computers into a single system to perform a computational task distributed across the cluster. Multiprocessor systems on the other hand could be a single physical entity comprising of multiple CPUs. A clustered system is less tightly coupled than a multiprocessor system. Clustered systems communicate using messages, while processors in a multiprocessor system could communicate using shared memory.
In order for two machines to provide a highly available service, the state on the two machines should be replicated and should be consistently updated. When one of the machines fails, the other could then takeover the functionality of the failed machine.
1.17 Consider a computing cluster consisting of two nodes running a database. Describe two ways in which the cluster software can manage access to the data on the disk. Discuss the benefits and disadvantages of each.
Answer:
Consider the following two alternatives: asymmetric clustering and parallel clustering. With asymmetric clustering, one host runs the database application with the other host simply monitoring it. If the server fails, the monitoring host becomes the active server. This is appropriate for providing redundancy. However, it does not utilize the potential processing power of both hosts. With parallel clustering, the database application can run in parallel on both hosts. The difficulty in implementing parallel clusters is providing some form of distributed locking mechanism for files on the shared disk.

1.18 How are network computers different from traditional personal computers? Describe some usage scenarios in which it is advantageous to use network computers.

Answer:
A network computer relies on a centralized computer for most of its services. It can therefore have a minimal operating system to manage its resources. A personal computer on the other hand has to be capable of providing all of the required functionality in a stand-alone manner without relying on a centralized manner. Scenarios where administrative costs are high and where sharing leads to more efficient use of resources are precisely those settings where network computers are preferred.

1.19 What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?
Answer:
An interrupt is a hardware-generated change of flow within the system. An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call operating system routines or to catch arithmetic errors.

1.20 Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load.
a. How does the CPU interface with the device to coordinate the transfer?
b. How does the CPU know when the memory operations are complete?
c. The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.
Answer:
The CPU can initiate a DMA operation by writing values into special registers that can be independently accessed by the device. The device initiates the corresponding operation once it receives a command from the CPU. When the device is finished with its operation, it interrupts the CPU to indicate the completion of the operation.
Both the device and the CPU can be accessing memory simultaneously. The memory controller provides access to the memory bus in a fair manner to these two entities. A CPU might therefore be unable to issue memory operations at peak speeds since it has to compete with the device in order to obtain access to the memory bus.

1.21 Some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is and that it is not possible.
Answer:

An operating system for a machine of this type would need to remain in control (or monitor mode) at all times. This could be accomplished by two methods:
a. Software interpretation of all user programs (like some BASIC, Java, and LISP systems, for example). The software interpreter would provide, in software, what the hardware does not provide.
b. Require that all programs be written in high-level languages so that all object code is compiler-produced. The compiler would generate (either in-line or by function calls) the protection checks that the hardware is missing.

1.22 Many SMP systems have different levels of caches; one level is local to each processing core, and another level is shared among all processing cores. Why are caching systems designed this way?
Answer:
The different levels are based on access speed as as well as size. In general, the closer the cache is to the CPU, the faster the access. However, faster caches are typically more costly. Therefore, smaller and faster caches are placed local to each CPU, and shared caches that are larger, yet slower, are shared among several different processors.

1.23 Consider an SMP system similar to the one shown in Figure 1.6. Illustrate
with an example how data residing in memory could in fact have a different value in each of the local caches.
Answer:
Say processor 1 reads data A with value 5 from main memory into its local cache. Similarly, processor 2 reads data Ainto its local cache as well. Processor 1 then updates A to 10. However, since A resides in processor 1’s local cache, the update only occurs there and not in the local cache for processor 2.

1.24 Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following processing environments:
a. Single-processor systems
b. Multiprocessor systems
c. Distributed systems
Answer:
In single-processor systems, the memory needs to be updated when a processor issues updates to cached values. These updates can be performed immediately or in a lazy manner. In a multiprocessor system, different processors might be caching the same memory location in its local caches. When updates are made, the other cached locations need to be invalidated or updated. In distributed systems, consistency of cached memory values is not an issue. However, consistency problems might arise when a client caches file data.

1.25 Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.
Answer:
The processor could keep track of what locations are associated with each process and limit access to locations that are outside of a program's extent. Information regarding the extent of a program's memory could be maintained by using base and limits registers and by performing a check for every memory access.

1.26 Which network configuration—LAN or WAN—would best suit the following environments?
a. A campus student union
b. Several campus locations across a statewide university system
c. A neighborhood
Answer:
a. LAN
b. WAN
c. LAN or WAN

1.27 Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.
Answer:
The greatest challenges in designing mobile operating systems include:
• Less storage capacity means the operating system must manage memory carefully.
• The operating system must also manage power consumption carefully.
• Less processing power plus fewer processors mean the operating system must carefully apportion processors to applications.

1.28 What are some advantages of peer-to-peer systems over client-server systems?
Answer:
Peer-to-peer is useful because services are distributed across a collection of peers, rather than having a single, centralized server. Peer-to-peer provides fault tolerance and redundancy. Also, because peers constantly migrate, they can provide a level of security over a server that always exists at a known location on the Internet. Peer-to-peer systems can also potentially provide higher network bandwidth because you can collectively use all the bandwidth of peers, rather than the single bandwidth that is available to a single server.

1.29 Describe some distributed applications that would be appropriate for a peer-to-peer system.

Answer:
Essentially anything that provides content,in addition to existing services such as file services, distributed directory services such as domain name services, and distributed e-mail services.

1.30 Identify several advantages and several disadvantages of open-source operating systems. Include the types of people who would find each aspect to be an advantage or a disadvantage.
Answer:
Open source operating systems have the advantages of having many people working on them, many people debugging them, ease of access and distribution, and rapid update cycles. Further, for students and programmers there is certainly an advantage to being able to view and modify the source code. Typically open source operating systems are free for some forms of use, usually just requiring payment for support services. Commercial operating system companies usually do not like the competition that open source operating systems bring because these features are difficult to compete against. Some open source operating systems do not offer paid support programs. Some companies avoid open source projects because they need paid support, so that they have some entity to hold accountable if there is a problem or they need help fixing an issue. Finally, some complain that a lack of discipline in the coding of open source operating systems means that backward-compatiblity is lacking making upgrades difficult, and that the frequent release cycle exacerbates these issues by forcing users to upgrade frequently.