Quizlet

OS quiz 1 (Chapters 1-3)

34 terms | katracho7

Terms in this set (34)

1.1 What are the three main purposes of an operating system?	- 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"	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. For example, when we use PCs or handheld computers, it's appropriate for the operating system to forsake this principle and to "waste resources. Because those operating systems are designed mostly for individual usability, for ease of

resources? Why is such a system not really wasteful?	use, with some attention paid to performance, and none paid to resource utilization.	
1.3 What are the main differences between operating systems for mainframe computers and personal computers?	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.	☆
1.4 List the four steps that are necessary to run a program on a completely dedicated machine.	- Reserve machine time. - Manually load program into memory. - Load starting address and begin execution. - Monitor and control execution of program from console.	☆
1.5 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?	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.	☆
1.6 What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?	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.	
1.7 Consider 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 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-than- satisfactory 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.	☆
1.8 How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?	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.	☆
1.9 Which of the following instructions should be privileged? a. Set value of timer.	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.	☆

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.		
1.10 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.	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.	☆
1.11 Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?	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.	
1.12 Timers could be used to compute the current time. Provide a short description of how this could be accomplished.	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.	☆
1.13 Is the Internet a LAN or a WAN?	The Internet is a WAN as the various computers are located at geographically different places and are connected by long-distance network links.	☆
1.14 How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?	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.	☆
1.15 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	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	☆

that large and eliminate the device?	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.	
1.16 Distinguish between the client - server and peer-to-peer models of distributed systems.	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).	☆
2.1 What is the purpose of system calls?	System calls allow user-level processes to request services of the operating system.	☆
2.2 What are the five major activities of an operating system with regard to process management?	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	☆
2.3 What are the three major	a. Keep track of which parts of memory are currently being used and by whom.	☆

activities of an operating system with regard to memory management?	b. Decide which processes are to be loaded into memory when memory space becomes available. c. Allocate and deallocate memory space as needed.	
2.4 What are the three major activities of an operating system with regard to secondary-storage management?	• Free-space management. • Storage allocation. • Disk scheduling.	☆
2.5 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.	☆
2.6 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.	☆
2.7 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.	☆
2.8 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.	☆
2.9 List five services provided by an operating system, and explain	a. Program execution. The operating system loads the contents (or se file into memory and begins its execution. A user-level program coul	

how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

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 th and the operation to perform on it, while the system converts that re device-or controller-specific commands. User-level programs canno to access only devices they should have access to and to access ther they are otherwise unused.
c. File-system manipulation. There are many details in file creation, de allocation, and naming that users should not have to perform. Blocks space are used by files and must be tracked. Deleting a file requires r the name file information and freeing the allocated blocks. Protectio be checked to assure proper file access. User programs could neithe adherence to protection methods nor be trusted to allocate only fre deallocate blocks on file deletion.
d. Communications. Message passing between systems requires mes turned into packets of information, sent to the network controller, tra across a communications medium, and reassembled by the destinatic Packet ordering and data correction must take place. Again, user pro not coordinate access to the network device, or they might receive p 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 process- independent (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.	
2.10 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 not be available for the device. In this situation, the operating system must be stored in firmware.	☆
2.11 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.	☆
3.1 Using the program shown in Figure 3.30, explain what the output will be at Line A.	The result is still 5 as the child updates its copy of value. When control returns to the parent, its value remains at 5.	☆
3.2 Including the initial parent process, how many processes are created by the program shown in Figure 3.31?	There are 8 processes created.	☆
3.3 Original versions of Apple's mobile iOS operating system	1. A time sharing mechanism (for example: preemption) must be used to prevent process from monopolizing the CPU.	☆

provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.	2. Protection of resources: to avoid that one process can damage the execution of another process (for example: memory protection). 3. Prevent deadlocks: so that process don't wait forever for another process to release resources.	
3.4 The Sun UltraSPARC processor has multiple register sets. Describe what happens when a context switch occurs if the new context is already loaded into one of the register sets. What happens if the new context is in memory rather than in a register set and all the register sets are in use?	The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time. If the context is in memory, one of the contexts in a register set must be chosen and be moved to memory, and the new context must be loaded from memory into the set. This process takes a little more time than on systems with one set of registers, depending on how a replacement victim is selected.	☆
(Has code in book) 3.5 When a process creates a new process using the fork() operation, which of the following state is shared between the parent process and the child process? a. Stack b. Heap c. Shared memory segments	Only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.	☆
(Code in book) 3.6 With respect to the RPC	The "exactly once" semantics ensure that a remore procedure will be executed exactly once and only once. The general algorithm for ensuring this combines an acknowledgment (ACK) scheme combined with timestamps (or some other	☆

mechanism, consider the "exactly once" semantic. Does the algorithm for implementing this semantic execute correctly even if the ACK message back to the client is lost due to a network problem? Describe the sequence of messages and discuss whether "exactly once" is still preserved.	incremental counter that allows the server to distinguish between duplicate messages). The general strategy is for the client to send the RPC to the server along with a timestamp. The client will also start a timeout clock. The client will then wait for one of two occurrences: (1) it will receive an ACK from the server indicating that the remote procedure was performed, or (2) it will time out. If the client times out, it assumes the server was unable to perform the remote procedure so the client invokes the RPC a second time, sending a later timestamp. The client may not receive the ACK for one of two reasons: (1) the original RPC was never received by the server, or (2) the RPC was correctly received—and performed—by the server but the ACK was lost. In situation (1), the use of ACKs allows the server ultimately to receive and perform the RPC. In situation (2), the server will receive a duplicate RPC and it will use the timestamp to identify it as a duplicate so as not to perform the RPC a second time. It is important to note that the server must send a second ACK back to the client to inform the client the RPC has been performed.	
3.7 Assume that a distributed system is susceptible to server failure. What mechanisms would be required to guarantee the "exactly once" semantics for execution of RPCs?	The server should keep track in stable storage (such as a disk log) information regarding what RPC operations were received, whether they were successfully performed, and the results associated with the operations. When a server crash takes place and a RPC message is received, the server can check whether the RPC had been previously performed and therefore guarantee "exactly once" semanctics for the execution of RPCs.	☆

Operating Systems Chapter 2

Operating Systems Ch 2

67 terms

CMPSC472TEST1