Quizlet

CS 450 - OS Quiz Questions

Study online at quizlet.com/_fgwqz

1.	**Q1 - What distinguishes a teal-time operating system?**	A real time operating system functions correctly only if it returns the correct result within a time constrain (embedded systems, pacemakers).
2.	**Q1 - What does an Operating System do?**	It serves as an intermediary between the hardware and software applications, it manages systems resources, and it provides an environment for users programs to run.
3.	**Q1 - What is a cache?**	A really fast subset copy of data (usually memory)
4.	**Q1 - What is the difference between symmetric and asymmetric multiprocessor systems?**	- Symmetric: all processors are considered peers and independently of one another. - Asymmetric:
5.	**Q1 - Why are precess switches so expensive relative to thread switches?**	Because with process switches, memory mapping needs to be replaced
6.	**Q1 - Why do computer systems have operating systems?**	Allows for an easier interface between users, application programmers, and the hardware.
7.	**Q1 - Why does the operating system need a kernel mode and a user mode?**	To allow for different levels of privileges, which in turn keeps privileged executions from being called by non-authorized users.
8.	**Q1 - Why do operating systems use timers on all processes?**	To prevent infinite looping
9.	**Q1 - Why do we need caches?**	To speed things up.
10.	**Q1 - Why do we need to protect the users access to the I/O system?**	...
11.	**Q2 - What 4 segments of memory does a LINUX process have, and what goes in each?**	Text: code and literals Stack: local variables, temporary variables and return points for calling functions. Data: global variables Heap: program managed variables, not automatically managed like stack variables.
12.	**Q2 - What are the two main IPC methods?**	Shared memory Message passing
13.	**Q2 - What does exec() do?**	Replaces the memory of a running process with a new map for a program named in the exec() call.
14.	**Q2 - What does fork() do?**	Makes a copy of a running process.
15.	**Q2 - What is a kernel loadable module, and why is it useful?**	It is a module that runs in kernel space, that is loadable at any time. It is useful because the kernel does not have to be re-compiled to include modules.
16.	**Q2- What is a process control block and why do we need one?**	It is a kernel data structure that keeps the current state of a process, so that it may be properly resumed, paused, created and killed. It also maintains statistics about resource usage for processes.
17.	**Q2 - What is a Virtual Machine?**	A program running on a host machine that provides the illusion of a dedicated machine to a guest OS by faking the hardware interface presented to the guest OS.
18.	**Q2 - What is dtrace and why is it helpful?**	It is a kernel tracing tool that allows tracing the execution of kernel code for debugging a kernel while it is running.
19.	**Q2 - What is the file descriptor that is returned from a call to open a file?**	It is an index into the array of file handles maintained for each process.
20.	**Q2 - Why do we need interprocess communication?**	Because the process model prevents any communication by default.
21.	**Q3 - What are the three required conditions for critical section solutions?**	Mutual Exclusion Progress Bounded waiting
22.	**Q3 - What are the two hardware solutions discussed in the book?**	test and set compare and swap
23.	**Q3 - What is a deadlock?**	When there is a circular dependency among resource requests by processes.

24. **Q3 - What is a monitor?**	An ADT that encapsulates code and variables such that all processes use the monitor when accessing shared resources.
25. **Q3 - What is the difference between kernel threads and user threads?**	Kernel threads are scheduled by the OS, while user threads are managed by the user language runtime libraries.
26. **Q3 - What is the difference between synchronous I/O and asynchronous I/O?**	The process waits immediately for synchronous I/O to complete, and does not wait for asynchronous I/O to complete.
27. **Q3 - What is the idea behind Implicit Threading?**	The user does not specify the threads, but identifies regions of code for the compiler and the runtime system to map to threads.
28. **Q3 - What is the issue with thread-local storage?**	It is useful to have variables that are "global" to the thread, without making them global to the entire process and all threads.
29. **Q3 - What is the issue with threads and signalling?**	Where do the signals go? A specific thread? All threads? The "causing" thread?
30. **Q3 - What is the problem with threads using fork() and exec()?**	The fork() call usually copies the entire process, and the exec() usually replaces the entire process: this is probably overkill for the majority of thread usage.
31. **Q3 - What is wrong with Peterson's algorithm on modern hardware?**	It only works for two processes/threads, and it requires atomic load and store.
32. **Q3 - What is wrong with using lock variables directly?**	All programmers must code lock code perfectly.
33. **Q4 - Describe multilevel queue scheduling:**	Multiple queues, each with its own algorithm. Scheduling among the queues is usually done with priority scheduling, or time-slice scheduling.
34. **Q4 - How do we recover from deadlock?**	By aborting processes until the cycle is eliminated, or by aborting all deadlocked processes.
35. **Q4 - How do you detect deadlock?**	By forming the resource allocation graph, and finding the cycles in it.
36. **Q4 - How do you prevent circular wait?**	By always reserving resources in the same order.
37. **Q4 - How do you prevent deadlock?**	By making sure one of the 4 conditions does not hold.
38. **Q4 - If there are cycles in the resource allocation graph, is there always deadlock?**	No, if there is more than one instance of a resource, deadlock is not certain.
39. **Q4 - What are the main criteria for scheduling processes on a CPU?**	Utilization, Throughput, Turnaround Time, Waiting Time, and Response Time.
40. **Q4 - What are the necessary conditions for deadlock?**	Mutual exclusion, hold and wait, no preemption, circular wait.
41. **Q4 - What are the trade-offs with SJF scheduling?**	It is optimal for wait time, but it is difficult to know the length of the next CPU burst.
42. **Q4 - What is a resource allocation graph?**	A directed graph showing processes and resources as vertices, with edges representing requests and allocations.
43. **Q4 - What is multilevel feedback scheduling?**	Multilevel queues with a method to move processes among the priority queues.
44. **Q4 - What is the difference between SJF and shortest-remaining-time-first?**	Shortest-remaining-time-first is pre-emptive.
45. **Q4 - What is the main issue with FCFS scheduling?**	It may lead to long and unpredictable wait times.
46. **Q4 - What is the main problem with priority scheduling?**	It may lead to indefinite blocking (starvation)
47. **Q4 - Where do OSes use round-robin scheduling?**	For time-sharing systems.

48.	**Q5 - How does paging hardware translate logical addresses to physical addresses?**	Use the page bits of the logical address as an index into the page table to find the page frame, and then append the offset part of the logical address to the page frame number to form the physical address.
49.	**Q5 - How does the segmentation hardware translate logical addresses to physical addresses?**	It checks the length of the segment, and if OK, adds the logical address to the segment base register to generate the physical address.
50.	**Q5 - How do processes share pages?**	The page tables of each process point to the same page frames for shared code.
51.	**Q5 - What does swapping do?**	To make a process active, it copies the entire process from backing store when it becomes active (swap in). To make room for other processes, it copies a process out to backing store (swap out).
52.	**Q5 - What is an inverted page table, and what problem does it solve?**	Inverted page tables are indexed by page frame number instead of page number. This makes the translation tables much smaller, but at a cost of more costly translations.
53.	**Q5 - What is a page fault?**	When a process accesses a page that is not assigned a frame in the page table, and needs OS assistance in finding a free page and updating the page table entry.
54.	**Q5 - What is a translation look-aside buffer?**	A hardware cache of the logical to physical address translation to speed up address translation for paging.
55.	**Q5 - What is COW?**	Copy on write: only copies pages in cloned processes for processes that change the data in those pages. Otherwise, the pages are shared.
56.	**Q5 - What is the difference between absolute and relocatable address bindings?**	If the location of the code is known at compile time, the addresses can be absolute. If not, the compiler must generate relocatable code for the loader.
57.	**Q5 - What is the difference between static linking and dynamic linking?**	Static linking occurs at load time, dynamic linking occurs at run time if and when needed.
58.	**Q5 - What is the difference in fragmentation between segmentation and paging?**	Segmentation leads to external fragmentation (which may be very wasteful), while paging leads to internal fragmentation (usually much smaller).
59.	**Q5 - What is the idea behind demand paging?**	Only load pages as they are referenced, not before.
60.	**Q5 - What is wrong with contiguous allocation?**	It leads to fragmentation of memory.
61.	**Q5 - What problem does virtual memory solve?**	Separate memory spaces for each process, with logical spaces that may be greater than available physical memory.
62.	**Q6 - Contrast inodes with directories:**	There is one inode per file, but there may be many directory blocks per inode. Directories provide names and access rights for inodes, while inodes describe the file itself.
63.	**Q6 - How does UFS keep track of free blocks on a partition?**	Using a bit map.
64.	**Q6 - What are some issues with contiguous allocation?**	External fragmentation, compaction is difficult
65.	**Q6 - What are some of the issues with linked allocation?**	It is best used for sequential access, as direct access may require many seeks for each access.
66.	**Q6 - What is a major issue with ACLs?**	Access Control Lists are cumbersome to implement as they may get quite large, and they meke it difficult to share with groups of users.
67.	**Q6 - What is a mount point?**	A place in the directory file structure where a second file system can be mounted and then accessed as if it were located at that point in the hierarchy.
68.	**Q6- What is a page replacement policy?**	The policy that determines picks which page(s) to evict, and which page(s) to bring in.
69.	**Q6- What is a reference bit and how does it help paging algorithms?**	Reference bits keep track of when page frames have been referenced, so that paging algorithms can determine which pages have been referenced since the last reset of the bits

70. **Q6 - What is a typical access time for a modern disk, vs memory access times?**	Tens of milliseconds vs 100 nanoseconds: about a 10,000 to 1 ratio.
71. **Q6 - What is a UNIX inode?**	The structure that represents an individual file: type, owner, location of blocks, access times, etc.
72. **Q6 - What is direct access?**	When access patterns are not in any particular order. Database queries, for example.
73. **Q6- What is LRU, and what is wrong with it?**	Least Recently Used: difficult to manage the stack data structure usually used to implement it
74. **Q6 - What is sequential access?**	When most accesses are in order. Tape, for example.
75. **Q6 - What is the concept of log based file systems?**	All transactions are written sequentially to a log file, so that the log may be replayed to recover from errors, and so that updates to a file do not overwrite previous data.
76. **Q6- Wha...t is the concept of memory mapped files?**	Map a file into virtual memory, and let the paging system manage I/O to that file.
77. **Q6 - What is the key question governing a choice of backup strategy?**	What errors are expected, and how can the backup recover from those errors?
78. **Q6- What is the optimal page replacement algorithm?**	Replace the page that will not be needed until the farthest in the future.
79. **Q6- What is the page frame allocation issue?**	How many page frames to allocate to a process, and how to adjust the allocation if necessary
80. **Q6- What is the Working Set Model?**	A model that says most processes tend to have a fairly stable reference pattern among pages over time, and those pages form a "working set".
81. **Q6- What is thrashing?**	Very high paging activity.
82. **Q6 - What is wrong with FCFS scheduling?**	First Come First Served: average service times may be quite poor.
83. **Q6 - What is wrong with SSTF scheduling?**	Shortest Seek Time First: may cause indefinite waits for some requests.
84. **Q6- What is wrong with the optimal page replacement policy?**	It is impractical for most processes: It can not be implemented without knowledge of the sequence of execution of a process.
85. **Q6 - What makes up disk access time calculations?**	Seek time, Rotational latency, transfer time.
86. **Q7 - How does a bit vector keep track of which blocks of a disk partition are used?**	Each block is represented by a bit in the vector, 0 for unused, 1 for used.
87. **Q7 - Under what conditions is indexed allocation better than linked allocation?**	For random access: finding the right data block is one array access.
88. **Q7 - What does classic UNIX use for protection?**	Domains are associated with users, and program files have a setuid bit that allows the process running the program to assume the domain of the owner of the program file.
89. **Q7 - What does the first level interrupt handler need to do?**	Record the cause and source of the interrupt, and then re-enable interrupts.
90. **Q7 - What four registers does a typical I/O port have? What**	data-in register data-out register control register status register
91. **Q7 - What is an access matrix representation of protection?**	A matrix with one axis for domains, and one for objects, with the entries for rights of domains for use of objects.
92. **Q7 - What is DMA?**	Direct Memory Access, where the I/O transfers do not directly involve the CPU.
93. **Q7 - What is the advantage of keeping error checking information in the directory blocks, like ZFS does?**	The blocks can be checked using metadata that is separate from the data being checked, and may be on separate devices.
94. **Q7 - What is the concept of domains of protection?**	Domains specify what resources a process may access, and how.
95. **Q7 - What is the difference between DMA and programmed I/O?**	Programmed I/O uses CPU instructions for every transfer, while DMA only uses programmed I/O to set up the addresses, counts, control and status for a transfer.

96. **Q7 - What is the difference between memory-mapped I/O and Isolated I/O addresses?**	Memory mapped I/O can use ordinary memory access instructions, where isolated I/O needs special instructions for access to I/O device registers.
97. **Q7 - What is the difference between synchronous and asynchronous I/O?**	Synchronous I/O pauses the calling process until completion, where asynchronous allows the process to continue while the I/O completes.
98. **Q7 - What is the flaw in heirarchical domain protection?**	There is no way to enforce the need-to-know principle.
99. **Q7 - What is the main problem with Capability based systems?**	It is too difficult to correctly manage the capability keys.
100. **Q7 - What is the princple of least privilege?**	Processes, users and systems should be given just enough privileges to perform their tasks.
101. **Q7 - What problem does Role Based Access Control solve?**	It allows users to assume Roles, which have fine grained access rights, to solve to problem of heirarchical protection enforcing need-to-know.
102. **Q7 - What system does UNIX use for keeping track of data block?**	It uses a multi-level index system, where the first index block keeps pointers to the first few blocks of the file, and there are three pointers to single indirect, double indirect and triple indirect block.
103. **Q7 - When is it faster to use polled I/O than interrupt driven I/O?**	When doing I/O to devices that are periodic with predictable service times.
104. **Q7 - Why do some device interfaces have streams?**	To allow filters to be inserted between devices and calling programs.