Quizlet Operating Systems Concepts Ch. 1-8

Study online at quizlet.com/_dadky

Advantages of Multiprocessor	Increased Throughput, Economy of Scale, Increased Reliability
Application Programming Interface (API)	specifies a set of functions that are available to an application programmer including parameters and return values
Asymmetric Multiprocessing	Each processor is assigned a specific task and a master controls the system. Master/Slave
Asynchronous cancellation	One thread immediately terminates the target thread
Atomically	one uninterrupted unit
backing store	process swapped out of memory to this temporarily then brought back in
Belady's anomaly	page-fault rate might increase as allocated frames increases
Benefits of Multithreading	responsiveness, resource sharing, economy, utilization of multiprocessor (scalability)
Blocking receive	receiver blocks until a message is available
Blocking Send	sending process blocked until the message is received by the receiving process/mailbox
Bounded-buffer problem	problem with multithreading
Bounded capacity	Queue has finite length, at most n messages can reside.
Bounded waiting	there is a number of times other processes can enter cs's before a process who requested goes in it's
Buffering	a temporary queue for messages
busy waiting	continual looping while waiting to enter CS
Cache Coherency	All copies of "A" must be updated across multiple processors
Caching	As information is used, it is copied to a faster storage system on a temporary basis
circular wait deadlock avoidance	impose a total ordering on all resource types and require each process request resources in increasing order
Command Interpreter	Shells, get and execute the next user-specified command.
compile time address binding	absolute code can be generated if you know where process will reside
Context Switch	State save of the current process and state restore of a different process
contiguous memory allocation	each process is contained in a single contiguous portion of memory
copy-on-write	allows parent and child process initially to share the same pages
CPU Bound Process	generates I/O requests infrequently, does computations
CPU-I/O Burst Cycle	Alternation of CPU Burst and I/O Burst (timeline)
CPU Scheduling	If several jobs are ready to run at the same time, the system must choose among them
CPU Utilization	Keep the CPU as busy as possible
Critical Section Problem	When one process is in it's Critical Section, no others can be. Each must ask permission to enter it's section. _entry section, _exit section, _remainder section
Data Section	contains global variables
deadlock	a set of resources is in deadlock when every process in the set is waiting for an event that can be used only by another process in the set
deadlock avoidance	give the os additional information about which resources a process will request and use during its lifetime
Deadlock conditions	Mutual exclusion, Hold and Wait, No Preemption, Circular wait
deadlock detection	if single instance of each resource type, build wait-for graph search for cycle
deadlock prevention	provides methods for ensuring at least one of necessary conditions cannot hold
deadlock recovery	process termination: abort all deadlocked processes, or one at a time
deadlock recovery	algorithm to detect whether deadlock has occurred and algorithm to recover
Deferred cancellation	Target thread periodically checks whether it should terminate, terminates itself
Degree of Multiprogramming	the number of processes in memory
demand paging	load pages only as they are needed
Device Controller	processor that controls the physical actions of one or more storage devices
Device Driver	a program that determines how a computer will communicate with a peripheral device
Device Queue	Processes waiting for I/O Action

43. **Dining philosophers problem**	Five philosophers spend their lives thinking, eating, only five chopsticks
44. **Direct communication**	send(p, message), receive(q, message)
45. **Direct Memory Access (DMA)**	Uses one interrupt per block without intervention from the CPU
46. **Dual Mode Operation**	Privileges for kernel can not be performed by users (prevents infinite loops)
47. **effective access time**	how long to compute if there is a page fault
48. **equal allocation**	split m frames among n processes equally leaving remainder for buffer
49. **exec()**	loads a binary file into memory and starts execution
50. **Execution**	Parent/Child run concurrently
51. **execution time address binding**	process can be moved during execution, binding delayed with special hardware
52. **First-Come First-Served Scheduling**	Process that requests the CPU first is allocated first. FIFO queue. Average waiting time high, Gantt chart.
53. **fork()**	Creates a new process
54. **fragmentation**	first-fit and best-fit cause fragmentation, free memory space broken into small pieces
55. **frame-allocation algorithm**	how many frames to allocate to each process, when page replacement required
56. **global/local replacement**	global looks at all frames, local just at one process's
57. **Graphical User Interface**	Users employ a mouse-based window-and-menu system (desktop) and uses a mouse to click on images/icons
58. **hashed page table**	hash value is virtual page number, linked list of elements
59. **Heap**	memory that is dynamically allocated during process run time
60. **hierarchical paging**	page table is a tree, unused branches don't exist
61. **Hybrid Structure**	layered system in which one layer is a Mach microkernel. Application Environments > Kernel Environment.
62. **Indirect communication**	messages sent/received from mailboxes (create, send/receive, delete mailboxes)
63. **Inter-process Communication**	Information sharing between processes to speed up computation, allow modularity, convenience
64. **inverted page tables**	page table indexes physical memory, search to find entry
65. **I/O Bound Process**	spends more time doing I/O than computations
66. **Job Queue**	All processes in system
67. **job scheduling**	OS selects a job from the job pool and loads into memory for execution
68. **Kernel**	The one program always running (the operating system)
69. **Kernel Mode**	Mode Bit: 0. Denotes tasks executed on behalf of the kernel
70. **Layered OS Approach**	Dijskra. Scheduler > Memory Management > Communication > Manage I/O and all devices > Compiler Executing user programs/printing > User Programs
71. **lazy swapper**	never swaps a page into memory unless that page will be needed
72. **least-recently-used replacement**	uses recent past to predict future
73. **Load Balancing**	attempts to keep the workload evenly distributed across all processors in a system
74. **load time address binding**	compiler generates relocatable code for bindings
75. **locality of reference**	allows prediction of needed pages
76. **local replacement algorithm**	if one process thrashing, cannot steal frames from another
77. **logical address**	address located in the CPU
78. **Logical Memory**	Abstraction of main memory that the user sees (includes VM)
79. **Long Term Scheduler**	selects processes from the pool and loads them into memory for execution
80. **main memory**	process as one rectangle of memory addresses, range of legal addresses
81. **Mechanism**	how to do something
82. **Medium Term Scheduler**	Removes processes from memory to reduce degree of multiprogramming and later re-introduce.
83. **Memory Hierarchy**	Registers, Cache, Main Memory, Electronic/Magnetic Disks, Optical Disks, Tapes
84. **memory resident**	in memory, as long as needed pages are in memory no problems

85. **Message Passing Model** — A connection is opened between processes and messages are exchanged either directly or through a common mailbox (send/receive)

86. **Microkernel** — moves as much as possible out of kernel and creates modules (Mac OSX)

87. **MMU** — adds value in relocation register to every address

88. **modify bit** — every frame has a modify bit set by hardware whenever written

89. **Modules** — Scheduling classes, file systems, loadable system calls, executable formats, STREAMS modules, miscellaneous, device and bus drivers

90. **Monitors** — an abstract data type with public methods, the set of programmer-defined operations are provided mutual exclusion within the monitor. Each process signals wait(mutex) before CS and signal(mutex) after

91. **MS-DOS** — Simple structure. Application Program > Resident System Program > MS-DOS Drivers > ROM/BIOS & Device Drivers

92. **Multilevel Queue Scheduling** — partitions ready queue into several separate queues with absolute priority

93. **Multiple Processor Scheduling** — processors must be homogeneous, asymmetric (one processor accesses system data & structures), symmetric (each processor self-scheduling)

94. **Multiprocessor Systems** — Systems with 2 or more processors in close communication sharing the computer bus, clock, memory, and peripherals

95. **Multiprogramming** — organizes jobs (code and data) so the CPU always has one to execute

96. **Mutual Exclusion** — If a process is in its critical section, no other process can be in theirs

97. **Non-blocking receive** — Receiver retrieves either a valid message or null

98. **Non blocking Send** — the sending process sends the message and resumes operation

99. **Non-preemptive Scheduling** — Scheduling takes place only for Ready > Waiting or Process Termination

100. **non-uniform memory access** — memory access times vary significantly

101. **optimal-page-replacement algorithm** — lowest page-fault rate, does not exist

102. **OS Services** — User Interface, Program Execution, I/O Operations, File-system manipulation, Communications, Error detection, resource allocation, accounting, Protection/Security

103. **over-allocating** — more frames are required than available

104. **page fault** — trying to access a page not brought into memory, access to a page marked invalid

105. **page-fault frequency** — establish upper and lower bounds on desired page-fault rate

106. **page-fault rate** — measures the slowdown of the computer due to paging

107. **pager** — concerned with the individual pages of a process.

108. **page replacement** — os swaps out a process freeing all frames and reducing multiprogramming

109. **page table structure** — logical address spaces are too big, structure techniques

110. **paging** — permits physical address space of a process to be non-contiguous by avoiding external fragmentation. break memory into blocks

111. **physical address** — address seen by the memory unit

112. **physical address** — base address of page in physical memory + offset within page

113. **Physical Memory** — The Amount of RAM that is installed in a computer

114. **Pipe** — acts as a conduit allowing two processes to communicate

115. **Policies** — determine what will be done

116. **pool** — group of free pages for requests

117. **Preemptive** — Incurs a cost to access shared data

118. **Priority Scheduling Algorithm** — CPU allocated to process with highest priority. Internal (ratio average I/O burst to CPU burst) or External (importance). Indefinite blocking/starvation solved by aging.

119. **Privileged Instructions** — Designated machine instructions that may cause harm only allowed in kernel mode

120. **Process** — A Program in execution (program counter, stack, data section)

121. **Process Control Block** — Process State, Program Counter, CPU Registers, CPU Scheduling Information, Memory Management, Accounting Information, I/O Status Information

122. **Process Creation** — Parent creates child, resource sharing, children a subset of parents resources but not shared

123. **Process Scheduler** Manages the Job queue, Ready Queue, and Device queue

124. **Process States** New > Running > Waiting > Ready > Terminated

125. **Process Termination** child exit(), parent wait()

126. **Process termination** completion of execution

127. **Progress** If no process is in CS and some processes wish to enter theirs, only processes not in remainder can decide who goes next, cannot be postponed indefinitely

128. **proportional allocation** allocate available memory to each process according to size

129. **pure demand paging** never bring a page into memory until it is required

130. **Ready > Waiting** invokes wait()

131. **Ready Queue** All processes in memory ready to run

132. **reference bit** set by hardware whenever a page is referenced for LRU

133. **reference string** evaluate an algorithm by running on this string in memory

134. **release** process releases the resource (system call)

135. **Remote Procedure Call** a way to abstract the procedure-call mechanism for use between systems with network connections

136. **Rendezvous** both send() and receive() are blocking

137. **request** process requests resource. if it can't be granted requesting process must wait (system call)

138. **Resource Allocation** Multiple users or jobs running at the same time must share CPU cycles, main memory, file storage, and I/O devices

139. **Resource allocation graph** edge from process Pi to resource Rj is a _request edge other is _assignment edge

140. **Response Time** how fast do you get anything back (submission > response)

141. **response time** The time it takes to respond to user interactions such as a mouse click

142. **Round Robin Scheduling** designed for time-sharing systems, similar to FCFS but allows preemption

143. **Running > Ready** when interrupt occurs

144. **Scheduling Criteria** What are we trying to maximize/minimize? What runs next?

145. **secondary memory** holds pages no present in main memory

146. **Semaphores** contains an integer variable that can be accessed only through two standard operations - acquire/release

147. **shared memory** allowed by virtual memory

148. **Shared-Memory Model** Processes use shared-memory-create and shared-memory-attach system calls to create and gain access to regions of memory owned by other processes

149. **Shortest-Job-First Scheduling** Associates length of process's next CPU burst, assigns next CPU cycle to shortest. Preemptive (processes booted if shorter comes), Nonpreemptive. Sometimes no way to know length of next CPU burst (short-term CPU)

150. **Short Term CPU Scheduler** selects from among the processes ready to execute and allocates the CPU to one

151. **Short Term Scheduler** CPU Scheduler. Selects a process from memory that's ready to execute, allocates CPU

152. **Signal** used in UNIX to notify a process that a particular event has occurred

153. **Signal and continue** Q either waits until P leaves the monitor or for another condition

154. **Signal and wait** P either waits until Q leaves the monitor or for another condition

155. **simplest deadlock avoidance** require each process declare the _maximum number resources of each type it may need, linear algebra

156. **Socket** endpoint for communication. Identified by an IP Address concatenated with a port number.

157. **sparse** virtual address spaces with holes

158. **Stack** contains temporary data (function parameters, return addresses, local variables)

159. **stack algorithms** can't exhibit Belady's anomaly

160. **Storage Hierarchy** Magnetic Disk > Main Memory > Cache > Hardware Register

161. **Swapping** Processes are swapped in and out of main memory to the disk

162. **Swapping** Processes are swapped out and later swapped in by the Medium Term Scheduler

163.	**Symmetric Multiprocessing**	Most common type, each processor performs all tasks, all are peers
164.	**Synchronization hardware**	a lock is required for the CS. allow test/modify contents of a word or swap contents of two words atomically
165.	**System Calls**	Provide an interface to the services made available by the OS
166.	**system libraries**	shared by several processes through mapping
167.	**Text Section**	program code
168.	**thrashing**	high paging activity
169.	**Thread cancellation**	terminating a thread before it has completed
170.	**Threading**	One>One, Many>One, Many>Many
171.	**Thread Pool**	create a number of threads at process startup and place them in a pool where they wait for work. Limits number that can run at one time, faster than waiting to create new
172.	**Threads**	Allows one task at a time
173.	**Throughput**	Getting more work done in less time
174.	**time sharing**	Multitasking: CPU executes multiple jobs by switching among them so frequently the user doesn't notice
175.	**Trap**	A Software generated interrupt caused by an error or by a specific request from a user program that an operating-system service be performed
176.	**Turnaround Time**	time from submission (airbag deployment)
177.	**Types of System Calls**	process control, file manipulation, device manipulation, information maintenance, communications, protection
178.	**Unbounded capacity**	Queue's length is potentially infinite, any number of messages can wait, sender never waits
179.	**UNIX**	User (Shell & Commands, Compilers, System Libraries) > Kernel (Signals, File System, Swapping, CPU Scheduling, Virtual Memory) > Hardware (Disks, terminals)
180.	**use**	the process can operate on the resource
181.	**User Mode**	Mode Bit: 1. Denotes tasks executed on behalf of the user.
182.	**User OS Interface**	Command Interpreter or Graphical User Interface
183.	**victim frame**	frame killed by page-replacement algorithm
184.	**virtual address space**	logical/virtual view of how a process is stored in memory
185.	**Virtual Machines**	abstract the hardware into several different execution environments. Create illusion that each environment is running its own private computer.
186.	**Virtual Memory**	allows the execution of a process that is not completely in memory
187.	**virtual memory**	separation of logical memory as perceived by users from physical memory. no need to worry.
188.	**virtual memory fork**	parent process suspended, child uses address space of parent
189.	**VM Benefits**	complete protection of the system resources, perfect for development
190.	**Waiting > Ready**	completion of I/O event
191.	**Waiting Time**	short and long jobs arrive at the same time
192.	**Zero capacity**	queue has max length zero, link can't have any messages waiting. Sender must block until recipient receives.
193.	**zero-fill-on-demand**	pages zeroed-out before being allocated (erasing previous contents)