CSE 2431

Systems 2

Mike Green

Key Points – Midterm 1

Slide set A-1 - Intro

1. Definition of an OS

2. What are the three “pieces” of every modern OS, and what is the basic description of each of them? What abstractions are associated with each of the 3 pieces?

3. What are some reasons cited in the class slides for studying operating systems?

Slide set A-2 - Processes

4. Definition of a process

5. What is an execution stream?

6. What is process execution state? [Note that this is DIFFERENT from process state.]

7. What is the difference between a process and a program?

8. What are typical interfaces/actions provided by the process API?

9. What does the OS (Unix/Linux) do to create a process?

10. After the process is created, what must be done for the process to start execution?

11. What are the three process states, and what is a process doing in each state?

12. How is the CPU shared by processes in the system (time-sharing)? What does time-sharing of the CPU mean?

13. What are the differences between direct execution and limited direct execution?

14. What are some problems with direct execution?

15. How does the hardware and the OS provide limited direct execution by processes on the CPU? [status bit/system calls/interrupts and exceptions]

16. What is the difference between an interrupt and an exception?

17. What are the two types of exceptions, and how are they different?

18. Why must the CPU (rather than the OS) change the status bit from user mode to privileged mode? When does the CPU do this?

19. Does the CPU or the OS change the status bit from privileged mode to user mode? When does it do this?

20. What kinds of things are user processes not allowed to do (in user mode, which is how user processes execute instructions)?

21. What happens if a user process tries to do something which is restricted (not permitted for user processes to do)?

22. How is the CPU taken away from a user process which is executing instructions (how does the dispatcher get control of the CPU again)?

23. What is a context switch? What data is saved for one process and restored for another when a context switch occurs? [register values only]

24. What are the problems with cooperative multi-tasking?

25. What does “true multi-tasking” mean?

26. What does Unix/Linux fork do? What are all the things which must be done to create a new process?

27. What is the relationship between the parent’s address space and the child’s address space (assuming the child makes no exec system call)?

Slide set A-3 – CPU Scheduling

28. What transitions in process state can occur, and when do they occur?

29. Which parameters may a scheduler attempt to minimize? Which may it attempt to maximize? (Be sure you have a basic understanding of what each parameter involves).

30. What are the formulas for: turnaround time; response time; waiting/wait time?

31. Be sure you understand the operation of the following schedulers: FIFO/FCFS; SJF; STCF; RR (Know what the acronyms for each scheduler stand for). Which of these schedulers are non-preemptive? Which are preemptive? What does it mean for a scheduler to be preemptive?

32. Which of the schedulers above are subject to the convoy effect? [FIFO/SJF]

33. Why can an RR scheduler provide better response time (assuming an appropriate time quantum?

34. What is a time quantum/time slice (RR scheduling)?

35. In what way is an RR scheduler usually worse than FIFO/FCFS (with equal length jobs)?

36. How do I/O system calls by a process interact with CPU scheduling?

37. Understand the basic operation of the MLFQ scheduler discussed in the class slides and in class.

38. How is “history” used by the MLFQ scheduler?

39. What are the goals of the MLFQ scheduler?

40. What are the potential problems with the MLFQ scheduler? How can each type of problem be addressed?

Slide set A-4 - Memory

41. What are the motivations for the virtualization of memory?

42. What are the four goals of multiprogramming?

43. What is in the address space of a process (code, heap, stack)?

44. Which parts of the address space of a process are dynamic? Which part(s) is/are static?

45. Which mechanisms are mentioned in the class slides for the virtualization of memory (time sharing, static relocation, dynamic relocation with base, dynamic relocation with base/bounds, segmentation, paging)?

46. What does time sharing of memory involve? Even though time sharing greatly simplifies the virtualization of memory, why does it not work out as a good solution to the problem?

47. What does static relocation involve (how does it work)?

48. What are the disadvantages of static relocation?

49. What does dynamic relocation with a base involve (how does it work)?

50. What are the disadvantages of dynamic relocation with base?

51. What does dynamic relocation with base/bounds involve (how does it work)?

52. What are the advantages/disadvantages of dynamic relocation with base/bounds?

53. What does segmentation involve (how does it work)?

54. How are the bits in a virtual address used by the hardware to translate logical/virtual addresses to physical addresses when segmentation is employed?

55. What are the advantages/disadvantages of segmentation?

Slide set A-5 - Paging

56. What can we say about the size of a page? (It has a number of bytes which can be written as a power of 2.]

57. What is the relationship between the size of a virtual page and the size of a physical page (or page frame)? [They must be the same size.]

58. What is the fundamental idea behind paging (see the class slides)? [Divide the address space of each process and also physical memory into pages of the same size.]

59. How does paging work? [Processes use addresses to specify the virtual page and byte offset for instructions/data they access; the hardware looks at a page table in memory to translate the virtual page number to a physical page/frame number, and appends the offset to get the physical address.]

60. How are the bits in a virtual address used when paging is employed (In other words, which part of the process’s address space does each part of the address specify)?

61. Do the number of bits used for the virtual page, and the number of bits used for the physical page (page frame) have to be the same? [No, they do not; there may be fewer physical pages than the number of virtual pages, and this is (almost) always true in real systems.]

62. Which type of fragmentation does paging eliminate? [External fragmentation]

63. Which type of fragmentation can still occur if paging is used? [Internal fragmentation]

64. How is the translation (mapping) of virtual page number to physical page number performed? [By looking at the correct page table entry (PTE) for the virtual page, to find the physical page number where the virtual page is stored, assuming it’s in physical memory currently.]

65. What is a page table? [A large data structure used to store information about the physical location and access information for a process’s virtual pages]

66. Where are page tables stored? [In main memory (See below for hierarchical page tables, which allow the possibility of storing many pages of a process’s page table on disk unless/until they are needed.]

67. What is the PTBR? [A register in the MMU with the address of the page table for a process]

68. What happens to the PTBR when there is a context switch? [The dispatcher must change the PTBR to the address of the page table for the process which is going to run next; it gets the address from the PCB.]

69. What other kinds of data besides the physical page number (page frame) are stored in the page table entries (PTEs)? [Access bits (rwx), valid bit, present bit, reference bit]

70. Why does paging double the number of memory accesses when processes execute instructions? [For each access the process makes to memory, the hardware must get the PTE from the page table in order to translate the virtual address before accessing tha instruction/data the process is attempting to access.]

71. What are the advantages of paging?

72. What are the disadvantages of paging? [As we saw, they can be addressed, but it is important to understand what disadvantages paging involves, so that we know what we need to improve to make it work.]

Slide set A-6 - TLB

73. Which steps in the translation of virtual to physical addresses can be made faster by a TLB (at least, in some cases)? [Accessing the PTE]

74. What does “TLB” stand for? What is the idea behind the name?

75. What is a TLB? [A cache for PTEs for one or more processes running in the system]

76. What is the basic difference between a direct mapped cache and an associative cache?

77. What are the advantages and disadvantages of direct mapped and associative caches?

78. Why are TLBs virtually always fully associative?

79. What is TLB hit rate? What is TLB miss rate?

80. How can TLB hit rate be increased?

81. What is TLB reach?

82. What is a cache replacement policy? Why is a replacement policy needed for every cache?

83. What two TLB replacement policies were mentioned? How does each work? [LRU, Random]

84. Which replacement policy did we say is used most commonly for TLBs? [LRU]

85. Is an LRU replacement policy always the best policy for a TLB?

Slide set A-7 – Memory Policy

86. How can multiple processes that have a combined virtual address space which is larger than the amount of physical RAM installed in a system be run when paging is used?

87. What is “paging in” when a process is running using paging?

88. What is the importance of locality of reference in running processes when not enough physical RAM is available for the combined virtual address spaces of all of the processes?

89. Why is it necessary for paging to be used in order to run a number of processes at the same time which have a combined virtual address space which is larger than the amount of physical RAM installed in a system?

90. What is used as the cache for disk storage, which holds the entire virtual address space of a process which is running in a system?

91. What is the mechanism the OS uses to identify the location of a particular page a process’s virtual address space (in physical memory, or on disk, or perhaps neither)?

92. What are the policies the OS can use to determine whether a given page of a process’s virtual address space is in physical memory or on disk?

93. What is the purpose of the present bit in a PTE of a process’s page table?

94. What is a page fault exception? When a page fault occurs, what must the valid and present bits in the page table for the page be (which of 0 or 1 for each bit)?

95. Why must a page fault exception be a fault, and not a trap?

96. Be able to describe how the hardware (CPU) and the OS cooperate to translate addresses when paging is used.

97. What does the scheduler do when a page fault exception occurs for a process in a running state?

98. What are the policies the OS can use to determine when to bring a page in a process’s virtual address space from disk into memory (RAM)?

99. What two policies discussed can the OS use to determine when to select a page in memory (RAM) as a victim page to be replaced, if there are no free physical pages/frames available?

100. Which of the three page-replacement policies we discussed has the stack property? Which do(es) not have the stack property?

101. Which of the page replacement policies can exhibit Belady’s anomaly?

102. How can a perfect LRU page replacement policy be implemented (hardware perfect, software perfect)?

103. How does the clock algorithm approximate LRU page replacement (understand how the clock algorithm works)?

104. What are three extensions to the clock algorithm which we described, and what is involved in each extension (basic explanation is sufficient)?

105. If a page is referenced by a process, and the page has been referenced previously by the process, will the page always be in memory?

Slide set A-8 - Smaller Tables

106. What is a reason cited in the class slides why pages tables are so large even for processes that use only a small fraction of their virtual address space?

107. What is the basic strategy to make page tables smaller? [Avoid simple linear array page tables]

108. What is the difference between a hardware-controlled TLB and a software-controlled (OS-controlled) TLB?

109. What are the three alternate approaches to page tables discussed in class and the class slides?

110. What is an inverted page table? Why is it called “inverted”?

111. How many page tables are required in the system if an inverted page table is used?

112. What is the big advantage of an inverted page table?

113. What is the primary disadvantage of an inverted page table? [It is hard to get good search performance.] How can it be addressed? [Use a well-chosen hash function to find the page table entry to check, and use a next pointer to resolve collisions.]

114. Which kind of TLB (hardware or software controlled) is required for inverted page tables?

115. How does a segmented page table work?

116. How must virtual memory be managed in order to use a segmented page table? [Segmentation with paging must be used.]

117. How many page tables does a process have if segmented page tables are used? [One page table for each segment in the process.]

118. What are advantages of segmented page tables?

119. What are disadvantages of segmented page tables?

120. How does a hierarchical page table work?

121. What does the page table consist of for a two-level hierarchical page table?

122. What is the significant advantage of a hierarchical page table? [The entire page table of the process does not have to be in memory; only the page directory, and pages of the page table for pages the process is currently using need to be in the page table. This will typically be a small number of pages of the page table.]

123. What is a potential disadvantage of a hierarchical page table (See the next question below)?

124. How many memory accesses are required to get the PTE (Page Table Entry) if a two-level hierarchical page table is used, whenever there is a TLB miss?