Lecture

- In the lectures in week 15 we will finish the discussion of Chapter 9 (Virtual Memory) and start with Chapter 10/11 (File Systems). Memory mapping will be shown by example.

Exercises

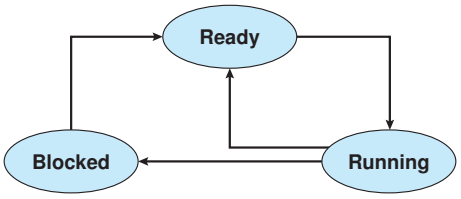
Note, as usual, that you find even more exercises including solutions here :
`http://codex.cs.yale.edu/avi/os-book/OS9/practice-exer-dir/index.html`

Prepare for the Tutorial Sessions in week 16, 2018 all exercises that have not been discussed so far as wall as the following:

9.1 Assume a program has just referenced an address in virtual memory. Describe a scenario how each of the following can occur: (If a scenario cannot occur, explain why.)

- TLB miss with no page fault
- TLB miss and page fault
- TLB hit and no page fault
- TLB hit and page fault

9.2 A simplified view of thread states is Ready, Running, and Blocked, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O). This is illustrated in the figure below. Assuming a thread is in the Running state, answer the following questions, and explain your answer:



a. Will the thread change state if it incurs a page fault? If so, to what new state?

b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what new state?

c. Will the thread change state if an address reference is resolved in the page table? If so, to what new state?

9.3 Consider a system that uses pure demand paging:

a. When a process first starts execution, how would you characterize the page fault rate?

b. Once the working set for a process is loaded into memory, how would you characterize the page fault rate?

c. Assume a process changes its locality and the size of the new working set is too large to be stored into available free memory. Identify some options system designers could choose from to handle this situation?

9.4 What is the copy-on-write feature, and under what circumstances is it beneficial? What hardware support is required to implement this feature?

9.5 A certain computer provides its users with a virtual-memory space of 232 bytes. The computer has 218 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

9.6 Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

9.7 When a page fault occurs, the process requesting the page must block while waiting for the page to be brought from disk into physical memory. Assume that there exists a process with five user-level threads and that the mapping of user threads to kernel threads is many to one. If one user thread incurs a page fault while accessing its stack, will the other user threads belonging to the same process also be affected by the page fault—that is, will they also have to wait for the faulting page to be brought into memory? Explain.

9.8 Consider the following page reference string: 7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1. Assuming demand paging with three frames, how many page faults would occur using a.) LRU replacement, b.) FIFO replacement, and c.) Optimal replacement.

9.9 The following page table is for a system with 16-bit virtual and physical addresses and with 4,096-byte pages. The reference bit is set to 1 when the page has been referenced. Periodically, a thread zeroes out all values of the reference bit. A dash for a page frame indicates the page is not in memory. The pagereplacement algorithm is localized LRU, and all numbers are provided in decimal.

a. Convert the following virtual addresses (in hexadecimal) to the equivalent physical addresses. You may provide answers in either hexadecimal or decimal. Also set the reference bit for the appropriate entry in the page table.

* 0xE12C

* 0x3A9D
* 0xA9D9
* 0x7001
* 0xACA1

b. Using the above addresses as a guide, provide an example of a logical address (in hexadecimal) that results in a page fault.

c. From what set of page frames will the LRU page-replacement algorithm choose in resolving a page fault?

9.10 Assume that you are monitoring the rate at which the pointer in the clock algorithm (which indicates the candidate page for replacement) moves. What can you say about the system if you notice the following behavior:

a. pointer is moving fast

b. pointer is moving slow

9.11 Discuss situations in which the LFU page-replacement algorithm generates fewer page faults than the LRU page-replacement algorithm. Also discuss under what circumstances the opposite holds.

9.12 Discuss situations in which the MFU page-replacement algorithm generates fewer page faults than the LRU page-replacement algorithm. Also discuss under what circumstances the opposite holds.

9.13 The VAX/VMS system uses a FIFO replacement algorithm for resident pages and a free-frame pool of recently used pages. Assume that the free-frame pool is managed using the least recently used replacement policy. Answer the following questions:

a. If a page fault occurs and if the page does not exist in the free-frame pool, how is free space generated for the newly requested page?

b. If a page fault occurs and if the page exists in the free-frame pool, how is the resident page set and the free-frame pool managed to make space for the requested page?

c. What does the system degenerate to if the number of resident pages is set to one?

d. What does the system degenerate to if the number of pages in the free-frame pool is zero?

9.14 Consider a demand-paging system with the following time-measured utilizations:

CPU utilization: 20%
Paging disk: 97.7%
Other I/O devices: 5%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Explain your answers.

a. Install a faster CPU.

b. Install a bigger paging disk.

c. Increase the degree of multiprogramming.

d. Decrease the degree of multiprogramming.

e. Install more main memory.

f. Install a faster hard disk or multiple controllers with multiple hard disks.

g. Add prepaging to the page fetch algorithms.

h. Increase the page size.

9.15 Suppose that a machine provides instructions that can access memory locations using the one-level indirect addressing scheme. What is the sequence of page faults incurred when all of the pages of a program are currently nonresident and the first instruction of the program is an indirect memory load operation? What happens when the operating system is using a per-process frame allocation technique and only two pages are allocated to this process?

9.16 Suppose that your replacement policy (in a paged system) is to examine each page regularly and to discard that page if it has not been used since the last examination. What would you gain and what would you lose by using this policy rather than LRU or second-chance replacement?

9.17 A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:

   * What the initial value of the counters is
   * When counters are increased
   * When counters are decreased
   * How the page to be replaced is selected

b. How many page faults occur for your algorithm for the following reference string 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2 in the case of four page frames?

c. What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

9.18 Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of 1 microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page-table entry is in the associative memory. Assume that 80 percent of the accesses are in the associative memory and that, of those remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

9.19 What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

9.20 Is it possible for a process to have two working sets, one representing data and another representing code? Explain.

9.21 Consider the parameter Δ used to define the working-set window in the working-set model. What is the effect of setting Δ to a small value on the page fault frequency and the number of active (non-suspended) processes currently executing in the system? What is the effect when Δ is set to a very high value?

9.22 Assume there is an initial 1024 KB segment where memory is allocated using the Buddy system. Using the figure explaining the Buddy system (course book) as a guide, draw the tree illustrating how the following memory requests are allocated:

- request 240 bytes
- request 120 bytes
- request 60 bytes
- request 130 bytes

Next, modify the tree for the following releases of memory. Perform coalescing whenever possible:

- release 240 bytes
- release 60 bytes
- release 120 bytes

9.23 A system provides support for user-level and kernel-level threads. The mapping in this system is one to one (there is a corresponding kernel thread for each user thread). Does a multithreaded process consist of (a) a working set for the entire process or (b) a working set for each thread? Explain.

9.24 The slab allocation algorithm uses a separate cache for each different object type. Assuming there is one cache per object type, explain why this doesn't scale well with multiple CPUs. What could be done to address this scalability issue?

9.25 Consider a system that allocates pages of different sizes to its processes. What are the advantages of such a paging scheme? What modifications to the virtual memory system provide this functionality?