FINAL EXAM with SOLUTION

Dec 17, 2007

Operating systems, V22.0202, Yap, Fall'07

> *1. PLEASE READ INSTRUCTIONS CAREFULLY.*
> *2. This is a closed book exam, but you may refer to two $8'' \times 11''$ sheets of prepared notes.*
> *3. Please write ONLY on the right-hand side of a double page. USE THE left-hand side of the double page for scrap.*
> *4. Attempt all questions.*

PART 1. VERY SHORT QUESTIONS. (5 Points each)
One or two-sentence answers only. For True/False questions, it is essential to provide the justifications asked for.

---

1. Give three reasons why a CPU is sometimes interrupted. One sentence per reason.

   SOLUTION *(1) There is an error in execution.*

   (2) An I/O device has completed an operation.

   (3) A user needs to access a protected (system) function. ♠

2. TRUE or FALSE: Our Toy HardwarE (THE machine) has relocatable addressing.

   SOLUTION *TRUE. It uses a base register to relocate logical addresses.* ♠

3. TRUE or FALSE: Paging causes external fragmentation. If true, justify; if false, modify into a correct statement by changing one word.

   SOLUTION *FALSE. The correct formulation can be one of the following: "Paging causes internal fragmentation" or "Paging solves external fragmentation".* ♠

4. Describe Belady's anomaly.

   SOLUTION *This is the phenomenon in page-replacement algorithms in which an increase in the number of page frames can cause the number of page faults to increase.*

   NOTE: FIFO algorithm exhibits Belady's anomaly. ♠

5. What is the advantage of using page sizes that is a power of two?

   SOLUTION *This allows you to take a logical address (in binary) and take its lower order bits as the page offset, and the higher order bits as the page number.* ♠

6. Suppose our paging system uses a Translation Lookaside Buffer (TLB). Each memory reference takes 300 ns, and each look up of the TLB takes 20 ns. What is the effective memory reference time if 80 percent of page-table references are found in the TLB?

   SOLUTION *Effective time in nanoseconds is* $(0.8 \times 320) + (0.2 \times 620) = 380$. ♠

7. An I-Node does NOT contain the name of the file it represents. Explain why this ought to be so.

   SOLUTION *An I-Node represents a physical file, but there might be several logical files with different names that point (links) to the same physical file.*

   REMARK: the file name is stored in the directory containing the file. ♠

PART 2. SHORT QUESTIONS. (10 points each)
Answers must be at most 4 sentences. So please sketch a rough solution first!

1. The code for wait() and signal() on a semaphore S is as follows.

$$wait(S) \equiv \{\text{while } S \leq 0; S\text{--}; \}$$

$$signal(S) \equiv \{S\text{++}; \}$$

Assume S is initially 1 and two processes P and Q protects a critical region with the semaphore S. *Show that if one of these operations is not executed atomically, we may violate mutual exclusion.*

SOLUTION *Suppose wait(S) is not atomic. Initially P reads S, finds the value positive, and proceeds to decrement S. But before it writes the value 0 to S, P is interrupted and Q begins to to execute $Wait(S)$. Q also reads $S = 1$, decrement the value to 0 and write out $S = 0$, and enters the critical region. Then Q is interrupted and P begins to run. P resumes by setting $S = 0$ (it is already set to 0 by Q) enters the critical region – violating mutual exclusion!!*

Note: signal(S) does not have to be atomic. ♠

2. When we use paging, each memory reference can turn into two *or more* memory references. Explain this remark.

SOLUTION *Each memory reference must first involve looking up the page table to find the frame containing the page containing our logical address. Looking up the page table is one memory reference. The second memory reference is to read the particular offset in a frame, corresponding to the logical address.*

Now, if the page table is large, we need to organize this as a hierarchical table. If the depth of this hierarchy is more $h+1$, then we will need $h+1$ memory references. ♠

3. In virtual memory (demand paging), we associate an r-bit and m-bit with each frame. (a) How are these bits updated? (b) How are these bits used in implementing the NRU (Not Recently Used) policy for page replacement?

SOLUTION *(a) The m-bit is set each time we write into the frame while the r-bit is set each time we access the frame. However, the r-bit is periodically reset, depending on the system clock.*

(b) each page belongs to a category 0,1,2 or 3, where the category of a page is given by the pair (r-bit,m-bit), viewed (r-bit,m-bit) as a binary number. We choose a frame with lowest category for eviction. ♠

4. Segmentation serves a different purpose than paging. Describe three uses of segmentation of processes.

SOLUTION *(1) Sharing of code segments across processes. (2) Refined protection control for data and code. (3) Management of multiple address spaces for data structures that can grow and shrink during run-time. (4) Paging performance can be improved also.* ♠

5. Consider the organization of memory for processes, and the organization of disk for files. Describe one similarity, and one difference.

SOLUTION *(1) Similarity in discrete space allocation: We divide memory into fixed-size units called frames, and divide disk into fixed-size units called blocks. (Both are methods of reducing fragmentation.)*

(2) Difference in Caching: To speed up memory access in the presence of paging, we use a caching mechanism based on the Translation Lookaside Buffer (TLB). There is no corresponding hardware support for caching between disk and memory. The closest to this idea is the use of swap space on the disk, allowing raw disk blocks to transferred somewhat faster. $\spadesuit$

6. Compute the inverse of 47 modulo 60. Explain your workings for full credit.

   SOLUTION *We use the extended Euclidean algorithm. Here are the details of the computation*

i	m_i	q_i	s_i	t_i
0	60		1	0
1	43		0	1
2	17	1	1	-1
3	9	2	-2	3
4	8	1	3	-4
5	1	1	-5	7

   Check: $1 = 60 \times (-5) + 43 \times 7$. $\spadesuit$

PART 3. LONG QUESTIONS. (20 Points each)
You may use more than 3 sentences to explain, and ought to consider the issues in greater depth.

---

1. (Scheduling)

   Joe Smart says "If you want to schedule processes to minimize the TOTAL wait time of all processes, it is quite easy. Just use the Shortest Jobs First rule." *First explain what Joe means. Then prove or argue why Joe is right. Finally, discuss the issues raised by this Smart idea and how we can resolve them.*

   SOLUTION *Smart is right: here is the proof. Suppose P takes time T and P' takes time T' where $T > T'$. Let us compare the total wait time S_1 when P is scheduled ahead of P', with the total wait time S_2 when P' is scheduled ahead of P. We see that $S_1 - S_2 = (2T + T') - (2T' + T) = T - T' > 0$. Thus S_1 is more than S_2.*

   There are two issues.

   (1) This solution assumes that WE KNOW THE RUNNING TIME OF EACH JOB. But we don't usually. To get this idea to work, we must have some ways to estimate the "expected" run time of each job. One possible solution is to use past executions of each program (perhaps combined with "aging" – see midterm) to estimate the expected run time.

   (2) If the jobs arrive at different times, we need to use preemption to minimize the total wait time. I.e., we now look at the remaining time to completion of job, and schedule the one with the least remaining time.

   This raises an issue of fairness: what if a long job is REPEATELY preempted? Say a 2-unit job is repeated preempted by the arrivals of many 1-unit jobs. Now it is not so clear that total wait time is the best policy. Of course, policy issue is not easily decided. $\spadesuit$

2. (Paging) Consider a paging system with a 32-bit logical address space. Each address refers to a byte in memory. Let the page size be 16 KB, and main memory size be 256 MB. What is the minimal size (in bytes) of the page table? Please show working.

   SOLUTION *ANSWER: 0.5MB.*

   Each page has 2^{14} Bytes, and main memory has 2^{28} Bytes or $2^{28}/2^{14} = 2^{14}$ frames. Hence the frames have 14-bit addresses. Thus, each entry of the page table need at least 14 bits, *or at least 2 bytes. Since logical address space has* $2^{32}/2^{14} = 2^{18}$ *pages, the page table has* $2^{18} \times 2 = 2^{19}$ *bytes, or* $0.5MB$.

REMARK: In practice, we probably want more than 2 bytes per entry: we may need space for the m-bit, r-bit, protection bits, present/absent bit, etc. ♠

3. (RSA Cryptosystem) Alice sent a secret message M to Bob using protocol A (which authenticates the receiver). Bob's public key is the $(e, n) = (43, 77)$. You are Jim Bond, the master spy in Her Majesty's secret service. You intercepted Alice's encrypted message $\widetilde{M}$. Normally, this code is unbreakable, but as Bond, you have access to unlimited computing power and can even factor numbers (gasp!). So you proceeded to break Alice's secret.

   What is the message M *given that* $\widetilde{M} = 2$*? Describe your method and show your computations.* You can get partial credits even if you could not compute M.

   SOLUTION $M = 51$. *Using the super-computer in Her Majesty's service, you managed to factor* $n = 77$ *after a long computation. You found that* $n = pq = 7 \times 11$. *So you compute* $\phi(n) = 6 \times 10 = 60$. *You know that* $e = 43$. *Using the Extended Euclidean algorithm, you compute* $d = e^{-1} \,\mathbf{mod}\, 60 = 7$ *(see a problem in the Short Questions Part). Now the original message must be* $(\widetilde{M})^7 \equiv 2^7 \equiv 128 \equiv 51 (\mod 77)$. ♠