

CS 702: Solutions to Final Exam Practice Problems

Problem 0: Review old homeworks, practice problems, and exams.

Problem 1:

- (a) Can a 700MB disk drive be formatted with a FAT16 file system with 4KB clusters?

No – FAT16 allows a maximum of 63356 clusters, and that many clusters of 4KB each gives a total of only 256MB.

- (b) What is the smallest cluster size that would allow full use of the drive from (a)?

16KB

- (c) How much space would be used by each FAT from your answer to (b)? We need 42,725 clusters. At 2 bytes per FAT entry and 1 entry per cluster, there are about 84KB taken up by each FAT.

Problem 2: Consider a file system similar to Second Extended File System (ext2) in which the i-nodes contain 16 double-indirect addresses, each block is 8KB, and each block address takes up 4 bytes. What is the largest file size possible using this system? What would the largest file size be using 16KB blocks?

Each block can hold 2048 block addresses. Each double-indirect address in the i-node then links to 2048^2 (about 4 million) addresses. So the largest possible file has $16 \cdot 2048^2$ blocks, or 512GB.

Problem 3: Which file system, ext2 or FAT-16, allows for a more efficient implementation of the `lseek` system call? Explain your answer.

To retrieve data at an arbitrary position in a file, we need to determine what block (or cluster) that position is in. With FAT16, we would have to traverse the linked list of clusters in the FAT to make that determination. In the worst case, we would have to traverse the entire list. Ext2, on the other hand, requires us to follow at most 3 links: from the i-node we get the triple-indirect block, in which we can determine the correct double-indirect block to examine, in which we can determine the correct single-indirect block to examine, from which we can get the correct block.

Problem 4: A certain system uses a single-level paging system with 64KB pages. Given the following page table, compute the physical addresses for the given virtual addresses. Some virtual addresses may cause page faults; in such cases simply write “Page fault”.

Page Number	Frame	Present
7	3	yes
6	-	no
5	-	no
4	1	yes
3	0	yes
2	2	yes
1	-	no
0	5	yes

- (a) 1000 = page 0, offset 1000 = frame 5, offset 1000 = physical address $5 * 65536 + 1000 = 328680$
- (b) 140000 = page 2, offset 8928 = frame 2, offset 8929 = 140000
- (c) 330000 = Page Fault (page 5)
- (d) 459000 = page 7, offset 248 = frame 3, offset 248 = 196856

Problem 5: Imagine a system that has a 32-bit virtual address space, uses two-level paging with 4KB pages, and allows the system administrator to determine the size of the top-level page directory. Suppose the system administrator notices that the vast majority of processes run on the system are exactly 4MB in size. How should the system administrator configure the system to minimize the total size of the page tables?

Each process uses $4\text{MB} = 1024$ pages. Any system in which the second-level page tables have more than 1024 entries will waste space. Any system with fewer than 1024 entries in the second-level page table will require a larger top-level page directory. Therefore, the optimal configuration have 1024 entries in the second-level tables, which means 10 bits are used for the entry within them. Since 12 bits were used for the offsets within the pages, 10 bits are left to determine the entry in the top-level directory, so there will be 1024 entries in the top-level directory.

Problem 6: Consider a system with four physical memory frames that are initially empty and the page references given below.

0 0 1 1 0 1 2 2 1 2 3 3 1 4 4 0 0 2 1 1 2 1 4 0 4 0 5 1

- (a) If using FIFO page replacement, show which references cause a page fault and show which pages are in which frames at the end of the sequence. Assume that when there are empty frames, pages are loaded into the lowest available frame. **0 0 1 1 0 1 2 2 1 2 3 3 1 4 4 0 0 2 1 1 2 1 4 0 4 0 5 1**
- (b) Repeat (a) for LRU **0 0 1 1 0 1 2 2 1 2 3 3 1 4 4 0 0 2 1 1 2 1 4 0 4 0 5 1**
- (c) Repeat (a) for Second chance **0 0 1 1 0 1 2 2 1 2 3 3 1 4 4 0 0 2 1 1 2 1 4 0 4 0 5 1**

Problem 7: Consider a system with three processes and three resource types. The following chart gives, for each process, how many of each resource type is allocated to it and what its maximum need is for each resource type. Suppose that the number of available (unallocated) resources for each type is $1\ 1\ x$. What is the lowest value of x for which this is a safe state? Explain your answer.

	allocated			max		
Process A	1	1	1	2	1	2
Process B	1	1	0	3	3	3
Process C	1	2	1	4	3	3

$x = 0$ is not safe because no maximum request could be satisfied. $x = 1$ is not safe because, although A's maximum request could be satisfied, after that neither B's nor C's could be satisfied. $x = 2$ is safe: A's maximum can be satisfied and then B's and finally C's.