CS471: Operating System Concepts
Fall 2006
(Lecture: TR 11:25-12:40 PM)
Homework #7
Points: 20
Due: October 19, 2006
SOLUTION

Question 1 [Points 2] 10.3

Solution:

Advantages of having mandatory locks to advisory locks:

1) If a process acquires an exclusive lock on a file, then the operating system prevents any other process from accessing the locked file.
2) The operating system ensures locking integrity using mandatory locks.

Disadvantages of having mandatory locks to advisory locks:

1) Processes should acquire exclusive lock only when they access the file otherwise they may block other processes from gaining the lock on those files.
2) Two or more processes may end up in deadlock waiting for each other to release the locks on the files that they acquired.

Question 2 [Points 2] 10.7

Solution: Airline-reservation system is an example of an application that benefits from operating system supporting random access.

Question 3 [Points 2] 10.9

Solution: With a single copy, several concurrent updates to a file may result in a user obtaining incorrect information, and the file being left in an incorrect state. With multiple copies, there is a storage waste and the various copies may not be consistent with respect to each other.

Question 4 [Points 8] The following sequence of operations are to be executed on a file of 10 kbytes, occupying 10 blocks on a disk (1 block = 1kbytes). Determine the number of disk accesses required for each operation for (a) Contiguous allocation (b) Linked allocation (c) Indexed allocation (assume that the index is already in main memory)
Operations:

(i) Read block 5
(ii) Update block 8
(iii) Insert a new block between 4 and 5 (assume that there is enough space preallocated in the contiguous allocation case for this file; the additional space is contiguous to the current blocks).
(iv) Delete block 2

Solution:

Contiguous allocation:
(i) 1 (Read block 5)
(ii) 1 (Write block 8)
(iii) 13 (Read blocks 5-10 and write them in 6-11. Write new block 5) (iv) 1
(iv) 16 **(**Read blocks 3-10, write them in 2-9)

Linked allocation:
(i) 5 (Read blocks 1-5)
(ii) 8 (Read blocks 1-7, write block 8)
(iii) 6 (Read blocks 1-4, write new block, write new 4)
(iv) 3 (read 1-2, write 1)

Indexed allocation;
(i) 1
(ii) 1
(iii) 1
(iv) 1

Question 5 [Points 2] 11.3

Solution:

a) In order to reconstruct the free space list, it would be necessary to perform "Garbage collection". This would entail searching the entire directory structure to determine which pages are already allocated to jobs. Those remaining unallocated pages could be re-linked as the free space list.

   Yes. The method employed is very similar to that used by many LISP systems for garbage collection. First we should establish a data structure representing every block on a disk supporting a file system. A bit map would Be appropriate here. Then, we would start at the root of the filesystem (the "/" directory), and mark every block used by every file we could find through a recursive descent through the file system. When finished, we would create a

a free list from the blocks remaining as unused. This is what the UNIX utility fsck does.

b) To load the directory "a" into memory we need one access. We then get the block address of the b from the index block of a. We then load the directory "b" into the memory. This needs another access. From the index block of "b", we get the block address of "c". We then load the directory "c" into the memory. This requires another access. From the index block of the "c" we read the required file. Together this requires three accesses.

c) The free space list pointer could be stored on the disk at several locations perhaps. Keep a "backup" of the free-space list pointer at one or more places on the disk. Whenever the beginning of the list changes, the "backup" pointers are also updated. This will ensure you can always find a valid pointer value even if there is a memory or disk block failure.

Question 6 [Points 4] 11.6

Let Z be the starting file address(block number).

Contiguous. Divide the logical physical address by 512 with X and Y the resulting quotient and remainder, respectively.

a) Add X to Z to obtain the physical block number. Y is the displacement into that block.

b) 1 (directly go to X+4)

Linked. Divide the logical physical address by 512 with X and Y the resulting quotient and remainder, respectively.

a) Chase down the linked list (getting X + 1 blocks). Y+1 is the displacement into the Last physical block.

b) 4

Indexed. Divide the logical physical address by 512 with X and Y the resulting quotient and remainder respectively.

a) Get the index block into memory. Physical block address is contained in the index Block at location X. Y is the displacement into the desired physical block.

b) 2