Module 7063 - Operating Systems
Exercises 12: File System Implementation

Problem 1

Explain the purpose of the **open** and **close** operations.

Answer

The **open** operation informs the system that the file specified in the path variable is about to become actively used by the calling process. The file can now be referenced by a file descriptor.

The **close** operation informs the system that the file given by the specified file descriptor is no longer in active use by the calling process.

Problem 2 (OSC-problem 11.2)

What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file?

Answer

The advantage is that while processing a block that is stored at the middle of a file, its location can be determined by chasing the pointers stored in the FAT as opposed to accessing all of the individual blocks of the file in a quasi sequential manner to find the pointer to the target block. Typically, most of the FAT can be cached in memory and therefore the pointers can be determined with just memory accesses instead of having to access the disk blocks.

Problem 3 (OSC-problem 11.4)

Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks.
How could the FS take advantage of this flexibility to improve performance?
What modifications would have to be made for the free space management scheme in order to support this feature?

Answer

Such a scheme would decrease internal fragmentation in files. If the size of a file is 5 KB, then it could be allocated a 4KB block and two contiguous 512-byte blocks. In addition to maintaining a bitmap of free blocks, one would also have to maintain extra state regarding which of the sub-blocks are currently used inside a block. The space manager would then have to inspect this extra state to coallesce the sub-blocks to obtain a larger block when all of the subblocks become free.

Problem 4 (OSC-problem 11.6)

Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the trhee allocations strategies (contiguous, linked, and indexed) answer these questions:

1. How is the logical to physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less that 512 blocks long)
2. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Answer

Let Z be the block number of the first block in the file.

- Contiguous Allocation: Divide the logical address A by 512 with X and Y the resulting quotient and remainder respectively.
    1. Add X to Z to obtain the physical block number. Y is the displacement into that block.
    2. 1
- Linked List Allocation: Divide the logical address A by 511 with X and Y the resulting quotient and remainder respectively.
    1. Chase down the linked list getting X+1 blocks. Y+1 is the displacement into the last physical block.
    2. 4
- Indexed Allocation: Divide the logical address A by 512 with X and Y the resulting quotient and remainder respectively.
    1. Get the index block into memory. The physical block address is contained in the index block at location X. Y is the displacement into the desired physical block.
    2. 2

Problem 5 (Problem 11.8)

In what situations would it be more useful to using a RAM disk than to using a disk cache?

Answer

In the cases where the user (or the system) knows exactly what data is going to be needed. Caches are algorithm-based, while RAM disks are user-directed.

Problem 6

Consider a file that consists of 100 blocks. Assume that the file control block (and the index block in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and single level indexed allocation strategies, if, for one block, the following conditions hold: in the contiguous allocation case, assume that there is no room to grow in the beginning, but there is room to grow in the end. Assume that the block information to be added is stored in memory.

1. The block is added at the beginning.
2. The block is added in the middle (after block #50)
3. The block is added at the end.
4. The block is removed from the beginning.
5. The block is removed from the middle (remove block #50)

6. The block is removed from the end.

Answer

```
(a) Contiguous Allocation
Case  Number of disk Operations
  1     201   move 100 blocks (one read/write) and write the new block
  2     101   move 50 blocks and write the new block
  3       1   write the block and change the length in dir
  4       0   only change start location in directory
  5     100   move blk 51..100 (50 blks)
  6       0   change length in directory

(b) Linked Allocation

Case  Number of disk Operations
  1   1    write new block and change start in directory
  2   52   read 50 links, write new block, rewrite block #50 (modified link)
  3   3    read block #100, modify link, rewrite block #100 and new block
  4   1    read block #1 copy its link to start in directory
  5   51   read 50 links, copy link of #50 to link in #49, rewrite #49
  6   100  read 99 links, set link of #99 to nil and rewrite #99

(b) Indexed Allocation: Note that it is wise to save the index block
                        when it has been modified

Case  Number of disk Operations
  1,2,3   2    write block, modify index block and rewrite latter
  2,4,5   1    modify index block and rewrite it
```

Franz Meyer
Last modified: Tue Jan/9/2011