Solutions of Lecture Slide Questions of File System Implementation

Dr Tarunpreet Bhatia

Assistant Professor

TIET

Question 1

Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

Number of entries in inode table = 8KB/4 bytes = 2048

Maximum file size = 12 X 8KB + (2048 X 8KB) + (2048 X2048 X 8KB) + (2048 X 2048 X 2048 X 8KB)

Question 2

Consider a system in which a directory entry can store up to 16 disk block address. For files not larger than 16 blocks, the 16 addresses serve as the file's index table. For files larger than 16 blocks, the address point to indirect blocks which in turn point to 256 file blocks each. A block is 1024 bytes. How big can a file be?

Max file size = 16 X 256 X 1024 bytes

Question 3

Consider a file currently consisting of 101 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 (read and write) are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. In calculating I/O operations, if the location of the first block of a file changes, do not include the output operations needed to rewrite that revised information to the directory entry on the disk. In the contiguous-allocation case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory and 51th block is middle block.

a. The block is added at the beginning.

b. The block is added after the middle.

c. The block is added at the end.

d. The block is removed from the beginning.

e. The block is removed from the middle.

f. The block is removed from the end.

Part	Contiguous	Linked	Indexed
a	203	1	1
b	101	53	1
c	1	103	1
d	0	1	0
e	100	52	0
f	0	101	0

Question 4

A file system uses 256-byte physical blocks and block numbering starts from 1. Now, assume that the last physical block read and the directory entries (corresponding to contiguous, linked and indexed strategies) are residing in the main memory. Further, assume the directory entries in indexed allocation contain pointers for 127 file blocks with an additional pointer to the next index block. In addition to the last block read, assume that the index block that contains pointer to the last block read resides in the main memory.

How many physical blocks must be read for the cases given below in context of contiguous, linked and indexed allocation??

i) Last block read: 100; block to be read: 600

ii) Last block read: 500; block to be read: 200

iii) Last block read: 20; block to be read: 21

iv) Last block read: 21; block to be read: 20

Part	Contiguous	Linked	Indexed
i	1	500	5
ii	1	200	3
iii	1	1	1
iv	1	20	1

Question 5

Consider a file system which uses an inode table to organize the files on disk. Each inode consists of a user id (2 bytes), three time stamps (4 bytes each), protection bits (2 bytes), a reference count (2 bytes), a file type (2 bytes) and the file size (4 bytes). The file system also stores the first 436 data bytes of each file in the inode table. Additionally, the inode contains 64 direct data block addresses as well as a single indirect index block and a double indirect index block. Each index entry takes 4 bytes, index table takes up an entire disk block, and a disk block is of 1024 bytes. Assume directory entry is already in main memory. Answer the following questions and show intermediate calculations.

i) What is the maximum data size (in MB up-to 2 decimal places) for a file in this system?

ii) How many disk accesses does it take to read data at relative location 2999885 within a file, assuming no caching?

Solution:
i) An indirect block has 1024 bytes/4 bytes = 256 entries
Maximum size of the file = 436 + (64 + 256 + 256^2) * 1024 bytes = 67699124 bytes = 67.70 MB
ii) 4 reads.