L39- File Management Tutorials

File Allocation Strategies

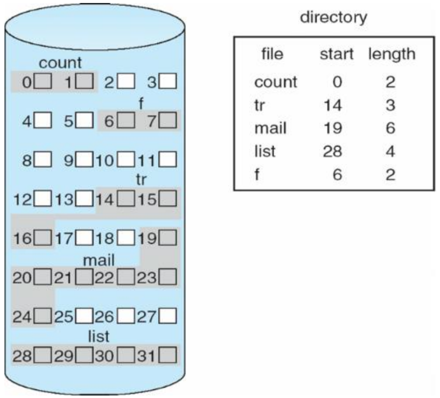
Q1: Consider a file currently consisting of 100 blocks. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. 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.

a. The block is added at the beginning. b. The block is added in the middle.

c. The block is added at the end. d. The block is removed from the beginning.

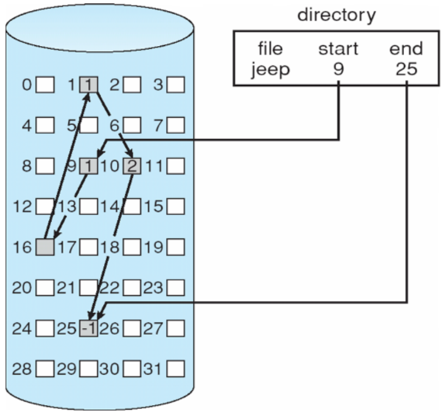
e. The block is removed from the middle. f. The block is removed from the end.

File Allocation Strategies



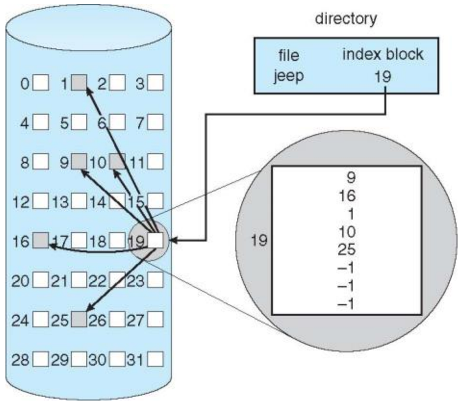
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous



file	start	end
jeep	9	25

Linked



file	index block
jeep	19

Indexed

File Allocation Strategies

File currently consisting of 100 blocks. Assume that the file control block/index block is already in memory. 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. Block information to be added is stored in memory.

a. The block is added at the beginning.

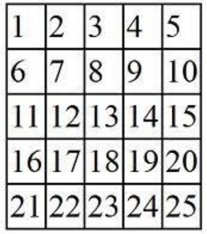
b. The block is added in the middle.

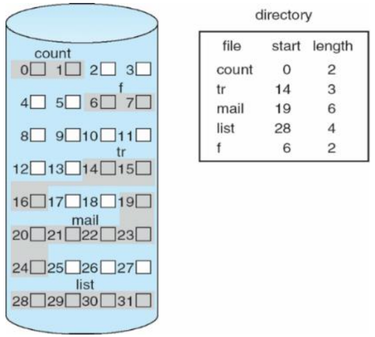
c. The block is added at the end.

d. The block is removed from the beginning.

e. The block is removed from the middle.

f. The block is removed from the end.





file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Case	**Cont**
a	201 (100R, 101W)
b	101 (50R, 51W)
c	1 (1W)
d	198 (99R, 99W)
e	98 (49R, 49W)
f	0

File Allocation Strategies

File currently consisting of 100 blocks. Assume that the file control block/index block is already in memory. 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. Block information to be added is stored in memory.

a. The block is added at the beginning.

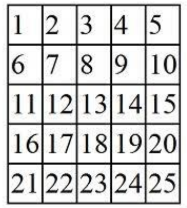
b. The block is added in the middle.

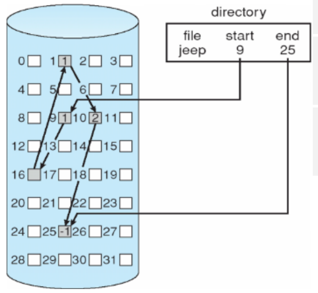
c. The block is added at the end.

d. The block is removed from the beginning.

e. The block is removed from the middle.

f. The block is removed from the end.





Case	Cont	Link
a	201	1 (1W)
b	101	52 (50R, 2W)
c	1	3 (1R, 2W)
d	198	1 (1R)
e	98	52 (51R, 1W)
f	0	100 (99R,1W)

File Allocation Strategies

File currently consisting of 100 blocks. Assume that the file control block/index block is already in memory. 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. Block information to be added is stored in memory.

a. The block is added at the beginning.

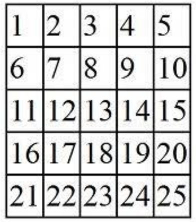
b. The block is added in the middle.

c. The block is added at the end.

d. The block is removed from the beginning.

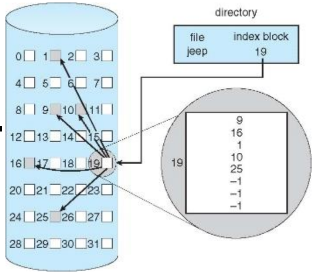
e. The block is removed from the middle.

f. The block is removed from the end.



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Case	Cont	Link	Index
a	201	1	1
b	101	52	1
c	1	3	1
d	198	1	0
e	98	52	0
f	0	100	0

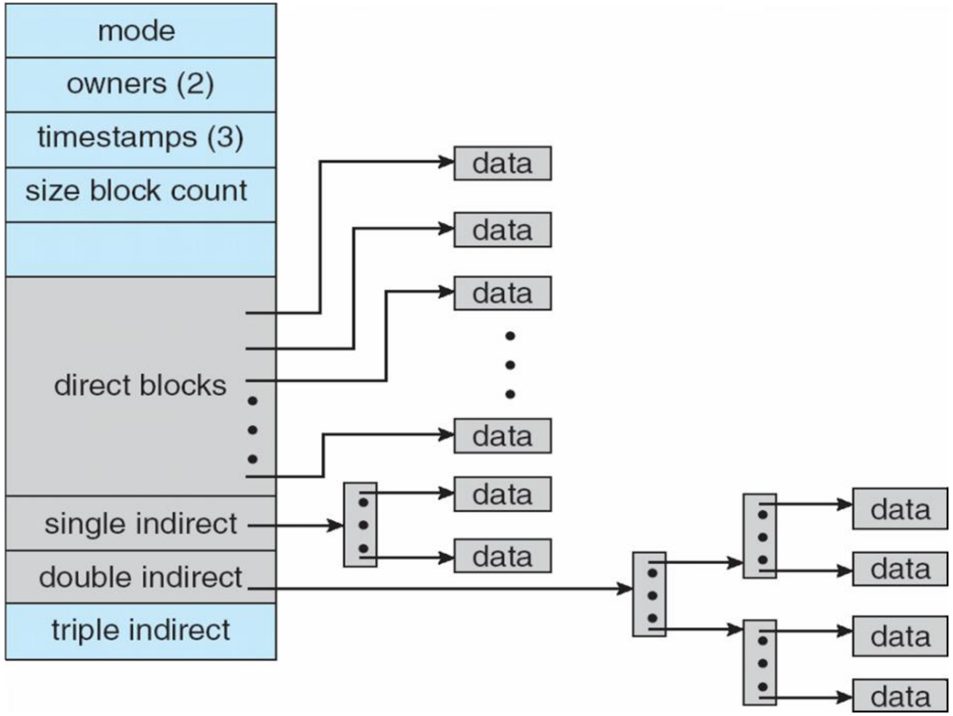


File Allocation using i-node

Q3: Consider a file management system that uses indexed allocation scheme on a hard disk with a 256B per blocks storage capacity. The blocks can be classified either as index blocks or data blocks. Every file is assigned a primary index block similar to the inode structure in Unix file system. The data block does not contain any header/ control information and the entire 256B can be used for data storage. Each primary index block contains a 32B header entry and 28 block index entries each of which is 8B. Each such block index entry carries the address of another block which can be either another index block or a data block. Out of the 28 block index entries, first 25 of them are pointing to data blocks directly and the remaining three are single indirect, double indirect and triple indirect index blocks like the one used in Unix inode. The 28 block index entries of a single indirect block are all pointing to data blocks, where as the 28 block index entries of a double indirect block are all pointing to addresses of another set of single indirect index block. Similarly the triple indirect index block can also be defined with 3 levels of indirection. Block mapping for a file is done by fully completing each lower level (entry 1-25) before going to next level of indirection (entry 26,27 and 28).

File Allocation using i-node

256B per data block



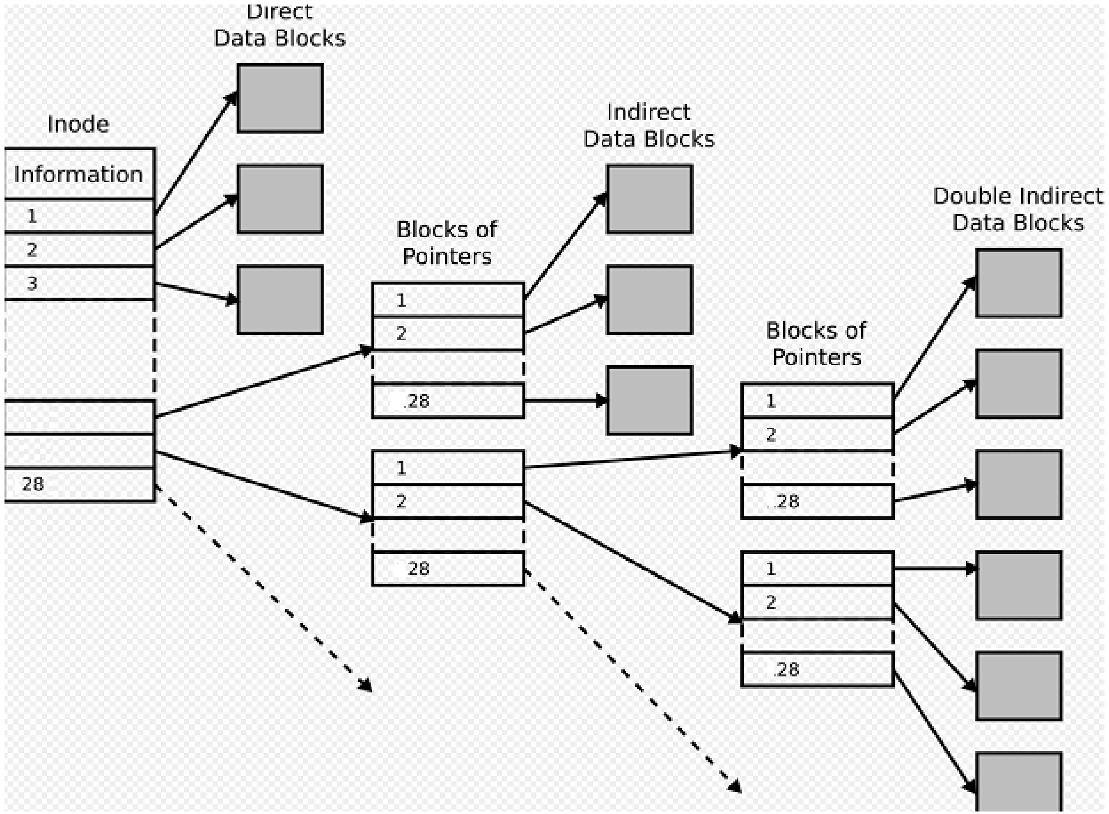
a. What is the maximum size of a file that can be mapped only with direct 25 data blocks of the primary index block (single, double and triple indirect entry is null)?

b. A file was assigned a valid entry in the single indirect entry of its primary index block and 20 out of 28 entries of that single indirect index block were valid. What can be the maximum size of the file?

c. What is max file size possible?

File Allocation using i-node

256B per data block



a. 25 x 256B= 6400B

b. (25x256B) + (20x256B) = 11520B

c. (25x256B)
+ (28x256B)
+ (28x28x256B)
+ (28x28x28x256B)

Thank You