

COS214 Tutorial 8

Roberto Togneri, 2000

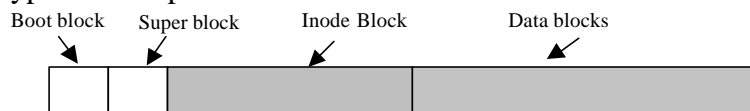
1. (a) Consider a system that supports the strategies of contiguous, linked, and indexed allocation. What criteria should be used in deciding which strategy is best utilized for a particular file?
(b) Fragmentation on a storage device could be eliminated by re-compaction of the information. Typical disk devices do not have relocation or base registers (such as are used when memory is to be compacted), so how can we relocate files? Give reasons why re-compacting and relocation of files often are avoided.

2. 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 three allocation strategies (contiguous, linked, and indexed), how is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

3. 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. Describe the type of operations 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 in the beginning, but there is room to grow in the end. 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 at the end.

4. (a) Consider a system where free space is kept in a free-space list.
 - (i) Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.
 - (ii) Suggest a scheme to ensure that the pointer is never lost as a result of memory failure.

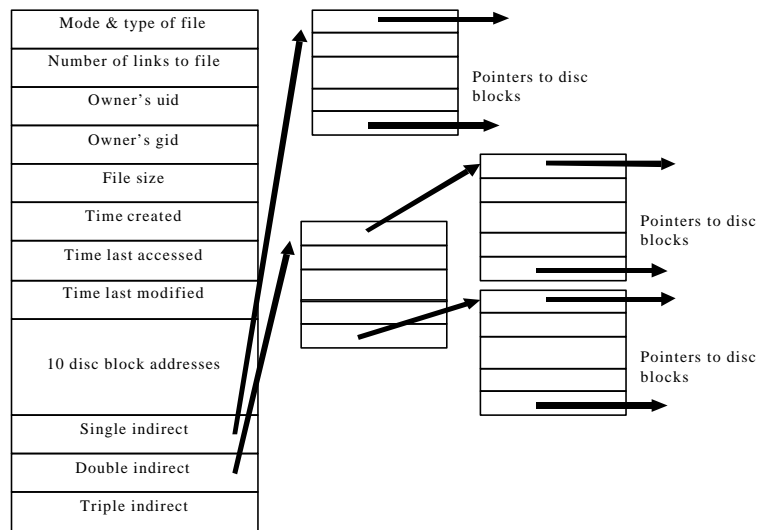
5. Consider a typical Unix partition:



The Unix directory entry looks like:

Inode Number 2 bytes	File name 14 bytes
-------------------------	-----------------------

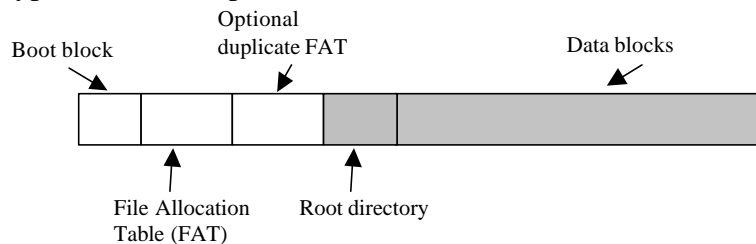
The other information is kept in the inode entry:



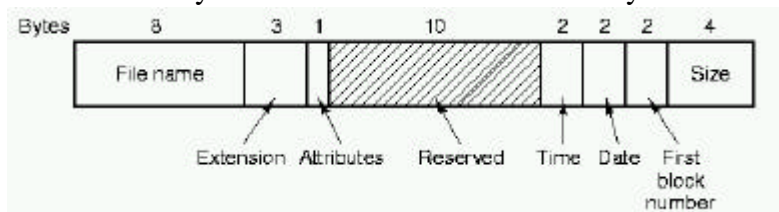
Assume disk blocks are 8K bytes and that disc addresses are 32 bits

- How is a file opened? Use as an example the file `/usr/home/test.c`.
- What size of file can be directly addressed from the information in the inode?
- What size of file requires a double indirect block?
- What is the largest possible file?

6. Consider a typical MS-DOS partition:



Both regular files and directory files can be defined and located by the FAT. Each directory contains an entry for each file within that directory:



Clearly show how the FAT is used to retrieve the data from the file `c:\windows\system.ini`.

7. A file system checker has built up its two lists as shown below:

Block #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
In Use:	1	0	1	0	0	1	1	1	1	0	2	0	0	1	0
Free:	0	0	0	1	1	1	0	0	0	1	0	1	2	0	1

Are there any errors? If so, are they serious? Why?