ECE437/CS481

M06B: DISK SPACE MANAGEMENT

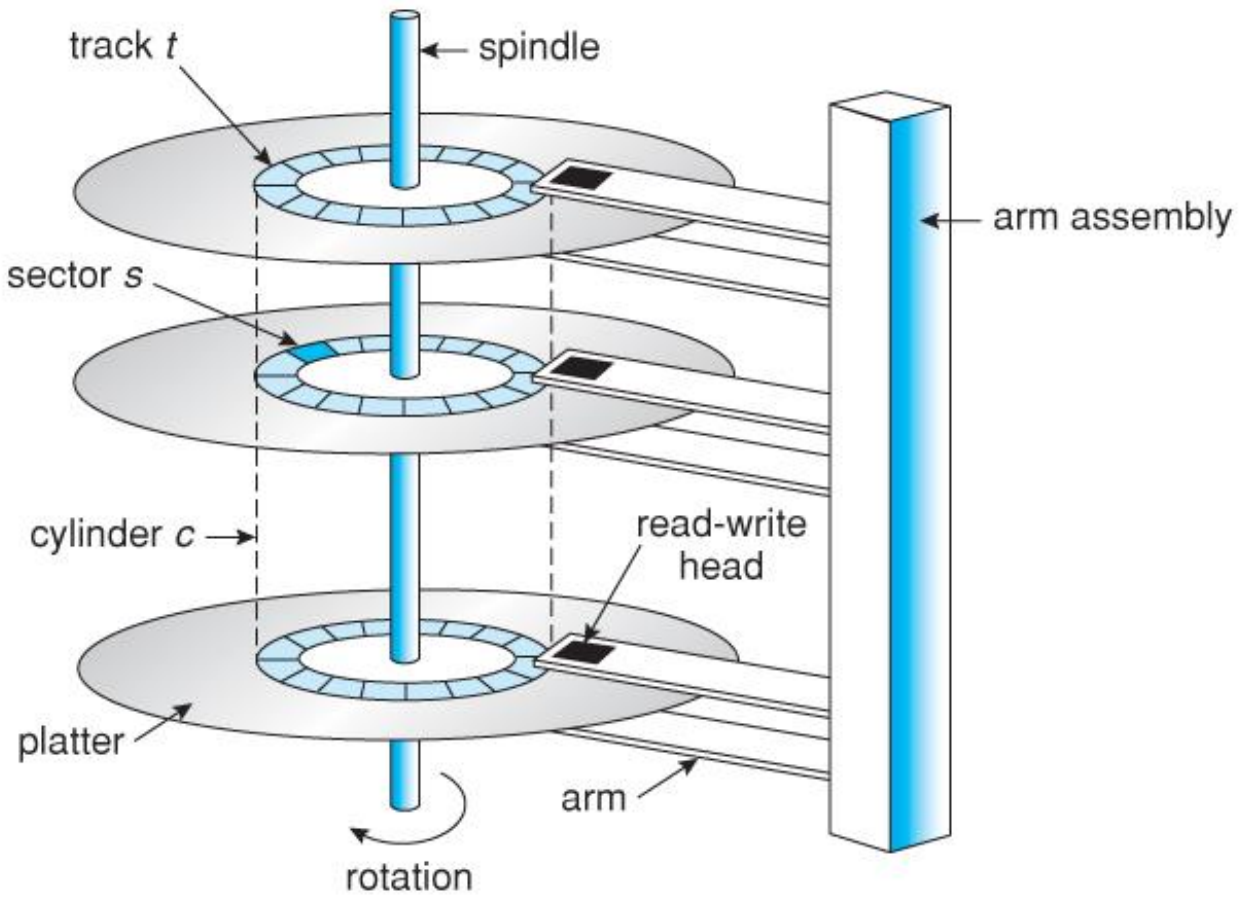
CHAPTER 12.4-12.5

Xiang Sun

The University of New Mexico

File Space Allocation

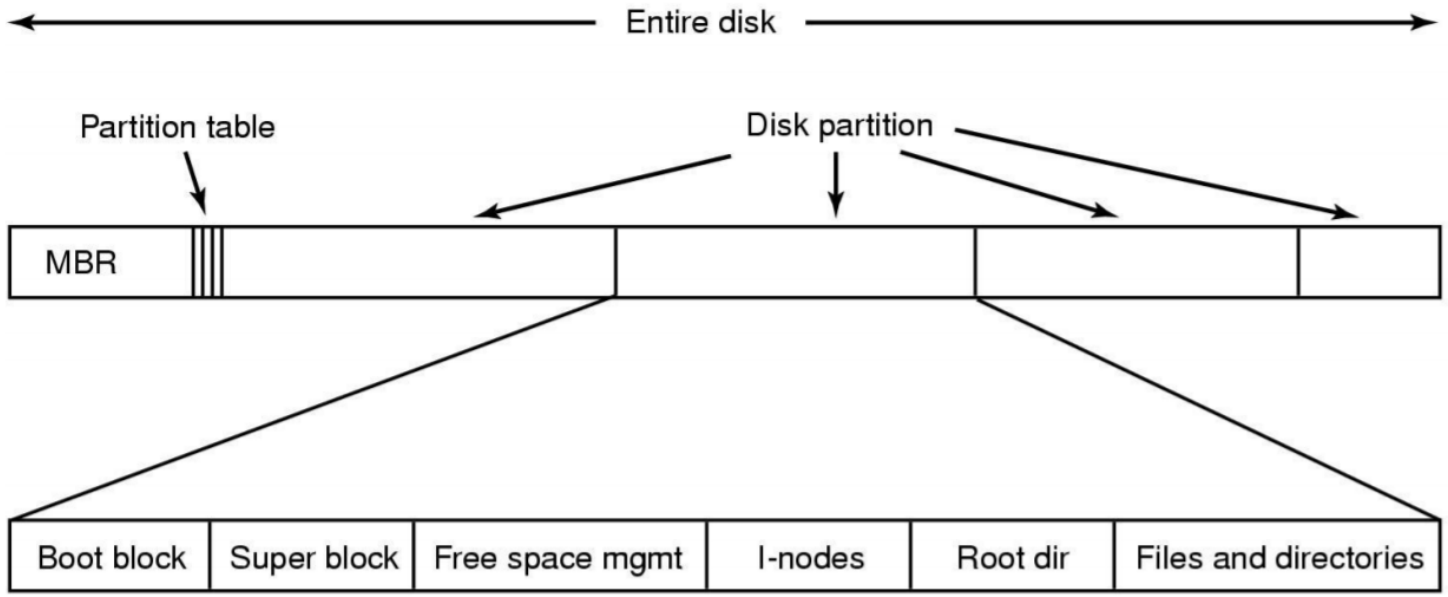
❑ Disk Structure



- Sector: Minimum unit (normally, 512 Byte for old hard disks; 4 KB for new hard disks) for data storage.
  - ✓ Use cat /sys/block/sda/queue/physical_block_size to see the size of sector for your hard disk.
- Cluster: Group of sectors
- Track: One cycle on the platter
- Cylinder: A semantic shape that consists of same level tracks on different platters.
- Each sector is identified by a combination of cylinder number, the head number, and the sector number.

File Space Allocation

❑ File System layout in a Disk



- MBR (sector 0 of the disk) is used to boot the computer.
- Partition table keeps the start and end of each partition.
- One of the partitions is active, MBR program locates the active partition and reads its first block (called the boot block) and execute it.
- The program in the boot block loads the OS in that partition.
- Superblock keeps all the important parameters about the file system (e.g., size, the block size, the empty and the filled blocks, the size and location of the inode tables, etc.) and it is read into memory when the computer is booted or the file system is used.

File Space Allocation

❑ Logic blocks in file systems

- The OS uses logic blocks to read and write files.
  - ❑ Size of a block = 2^n × size of a sector, range from 512 B to 64KB
  - ❑ "stat –f /" to check the size of a block
  - ❑ The way of addressing logic blocks is a simple linear addressing scheme.
  - ❑ Logical block addresses have to map into cylinder/head/sector tuples

LBA	C	H	S
0	0	0	0
1	0	0	1
2	0	0	2
3	0	0	3
4	0	0	4
5	0	0	5
6	0	0	6
7	0	0	7
8	0	0	8
9	0	0	9
10	0	1	0
11	0	1	1
12	0	1	2
13	0	1	3
14	0	1	4
15	0	1	5
16	0	1	6
17	0	1	7
18	0	1	8
19	0	1	9
Cylinder 0			

LBA	C	H	S
20	1	0	0
21	1	0	1
22	1	0	2
23	1	0	3
24	1	0	4
25	1	0	5
26	1	0	6
27	1	0	7
28	1	0	8
29	1	0	9
30	1	1	0
31	1	1	1
32	1	1	2
33	1	1	3
34	1	1	4
35	1	1	5
36	1	1	6
37	1	1	7
38	1	1	8
39	1	1	9
Cylinder 1			

LBA = ((C x HPC) + H) x SPT + S - 1

- ❖ C, H and S are the cylinder number, the head number, and the sector number
- ❖ LBA is the logical block address
- ❖ HPC is the number of heads per cylinder
- ❖ SPT is the number of sectors per track

File Space Allocation

- How to allocate blocks to these files so that disk space is utilized effectively and files can be accessed quickly?—File Space Allocation

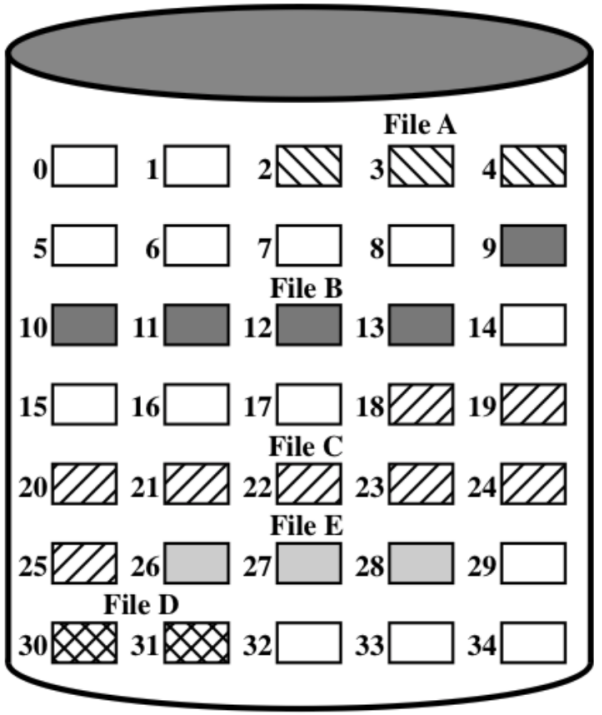
- Three major methods of allocating disk blocks:
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation

File Space Allocation

❑ Continuous allocation

- A file is allocated on contiguous group of blocks.
- Directory only requires a pointer to the 1st block and the # of blocks allocated
- Good performance for sequential access
- Easy to achieve direct access
  - ✓ If the system tries to access block *i* of a file that starts at block *b*, the system can immediately access block b + i

name	1st block	length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



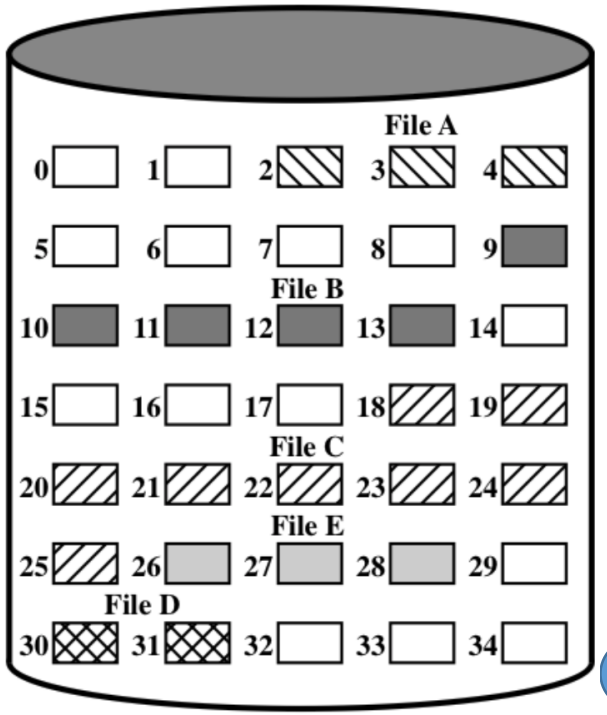
File Space Allocation

❑ Continuous allocation

- Block allocation problem: given the file size, how to pick up contiguous group of blocks to store the file.

First fit:	Best fit:
• choose the first unused contiguous group of blocks of sufficient sizes.	• choose the smallest unused contiguous group of blocks of sufficient sizes.

- If the system try to allocation a file with size equal to one logic block, which logic block will be used for applying FF and BF, respectively?



File Space Allocation

❑ Drawbacks of continuous allocation

- ➢ Suffers from the problem of external fragmentation
  - ✓ External fragmentation: As files are allocated and deleted, the free disk space is broken into little pieces. Thus, none of these pieces is large enough to store files.
  - ✓ Solution: Copy an entire file system onto another disk, and then copy the file system back to the original disk by allocating contiguous space.

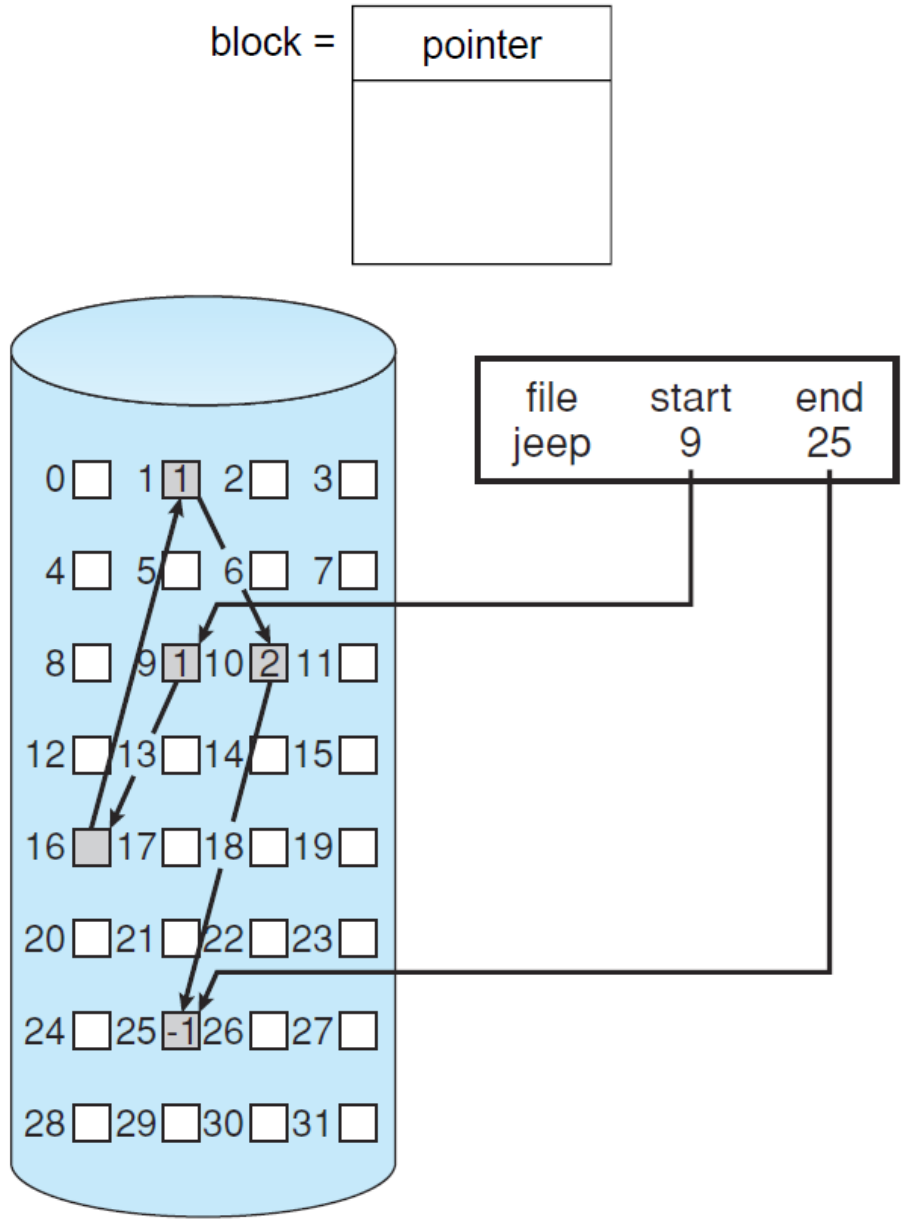
- ➢ May have problems in estimating the size of a file.

  - ✓ Determining how much space is needed for a file in continuous allocation.
  - ✓ In general, let the creator to estimate the size of a file is difficult (e.g., what is the size of a thesis that you will write).
  - ✓ If the estimated size of a file is too small, the file cannot be extended (especially for the best-fit allocation strategy).

File Space Allocation

❑ Linked allocation

- Each file is a linked list of blocks which need NOT be contiguous. The blocks can be scattered anywhere on the disk.
  - ✓ A file of five blocks might start at block 9 and continue at block 16, then block 1, then block 10, and finally block 25.
- Each block contains a pointer to the next
  - ✓ These pointers are not made available to the user,
- Directory only contains pointers to the 1st and/or last blocks of the file



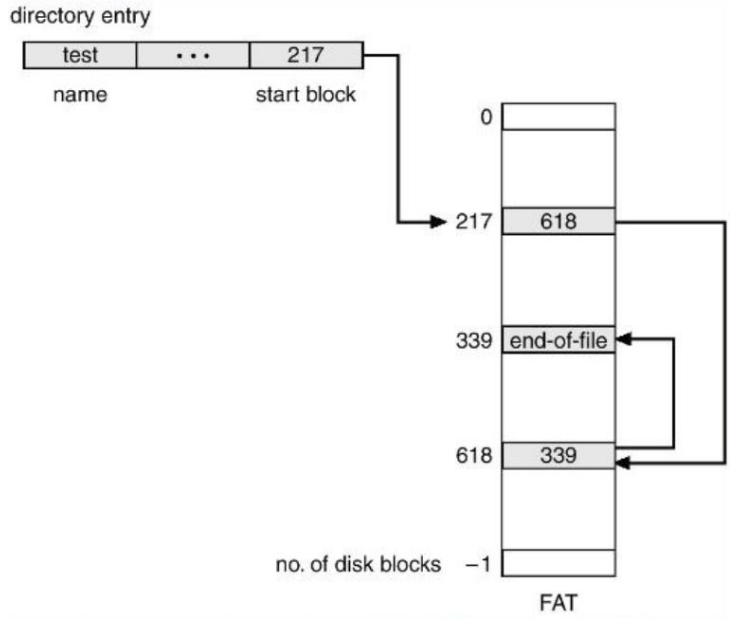
File Space Allocation

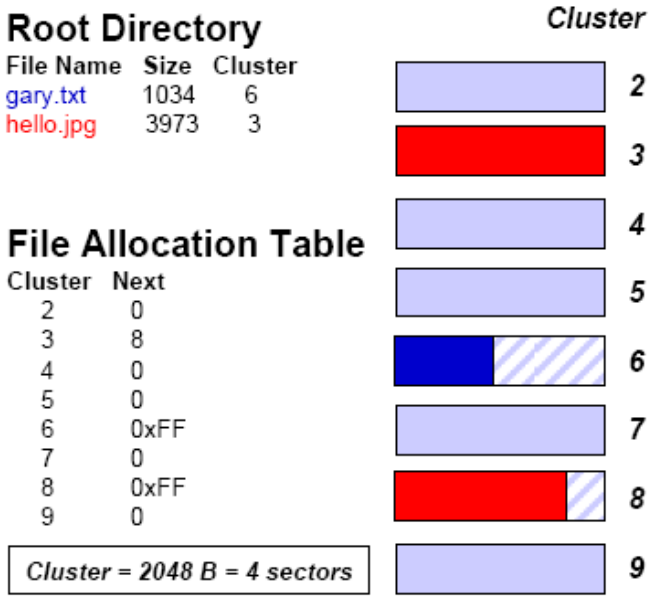
❑ Linked allocation pros and cons

- ➢ Flexible to extend, insert, and delete blocks
  - ✓ A write to the file causes to find a free block, and this new block is written to and is linked to the end of the file.
  - ✓ To read a file, we simply read blocks by following the pointers from block to block.
  - ✓ The size of a file need not be declared when the file is created. A file can continue to grow as long as free blocks are available.
- ➢ No external fragmentation
  - ✓ Any free block can be used to satisfy a request.
- ➢ Ineffectively for direct access
  - ✓ To find the i^{th} block of a file, we must start at the beginning of that file and follow the pointers until we get to the i^{th} block.
- ➢ Extra overhead for storing points in blocks
- ➢ Low reliability
  - ✓ A bug in software/hardware results in picking up the wrong pointer, thus linking into free space or another file.

File Space Allocation

- Linked allocation: MS-DOS FAT as an example
  - FAT = File Allocation Table, a variation of linked allocation
    - ✓ one FAT per file system
    - ✓ one entry per block containing the next block in the file
  - Normally, one block= 512 bytes→ too small
  - User cluster in FAT to replace block, where one cluster=2^x blocks (x=0,1,2,3,4,5,6,7)
  - Size of FAT is decided by total # of clusters on storage
    - ✓ In FAT, each entry represents a cluster
    - ✓ "0" means the cluster is free
    - ✓ "FF" means EOF





File Space Allocation

❑ Linked allocation: MS-DOS FAT as an example

➢ FAT-12: 12-bit addr space to index entries in FAT

- ✓ There are maximum 2^{12} clusters
- ✓ If one cluster=one block= 512 B, then max volume size=512 B/cluster* 2^{12} clusters=2 MB---floppy disk, whose size is 1.44 MB
- ✓ The size of a cluster (i.e., 2^x block) is specified in the boot record.

➢ FAT-16 (1987): 16-bit addr space to index entries in FAT

- ✓ There are maximum 2^{16} clusters
- ✓ If one cluster=one block= 512 B, then max volume size=512 B/cluster* 2^{16} clusters=32 MB
  Too small
- ✓ If one cluster=128 blocks= 64 KB, then max volume size= 64 KB/cluster * 2^{16} clusters=4 GB
  Fairly enough, but incurs internal fragmentation.
- ✓ Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation to the allocation granularity.

File Space Allocation

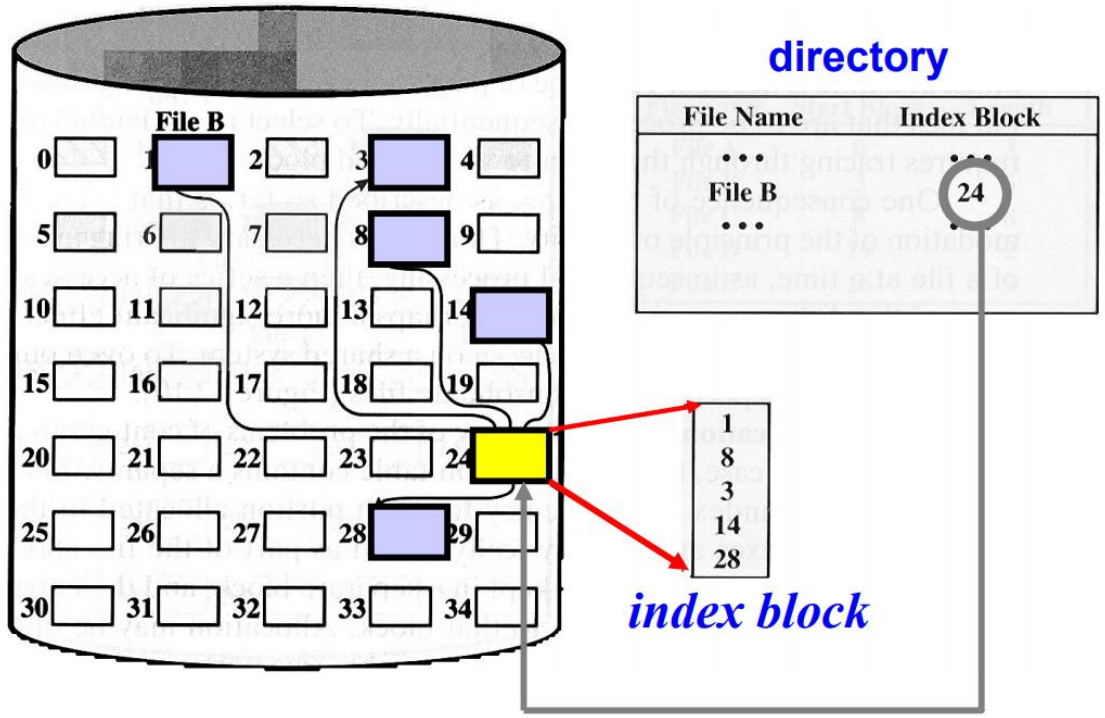
- Linked allocation: MS-DOS FAT as an example
  - FAT-32 (1996, Windows 95): 32-bit addr space to index entries in FAT
    - ✓ There are maximum 2^{32} clusters
    - ✓ If one cluster=one block= 512 B, then max volume size=512 B/cluster* 2^{32} clusters=2 TB
    - ✓ If one cluster=8 blocks= 4 KB, then max volume size=4 KB/cluster* 2^{32} clusters=16 TB

      The setup is used as default by NTFS
  - FAT requires storing the entire table in memory for efficient access --- A huge table!

File Space Allocation

❑ Indexed allocation

- ➢ Bring all pointers together in an index block
- ➢ The i^{th} entry in the index block points to the i^{th} block of the file.
- ➢ A file's directory entry contains a pointer to its index block.



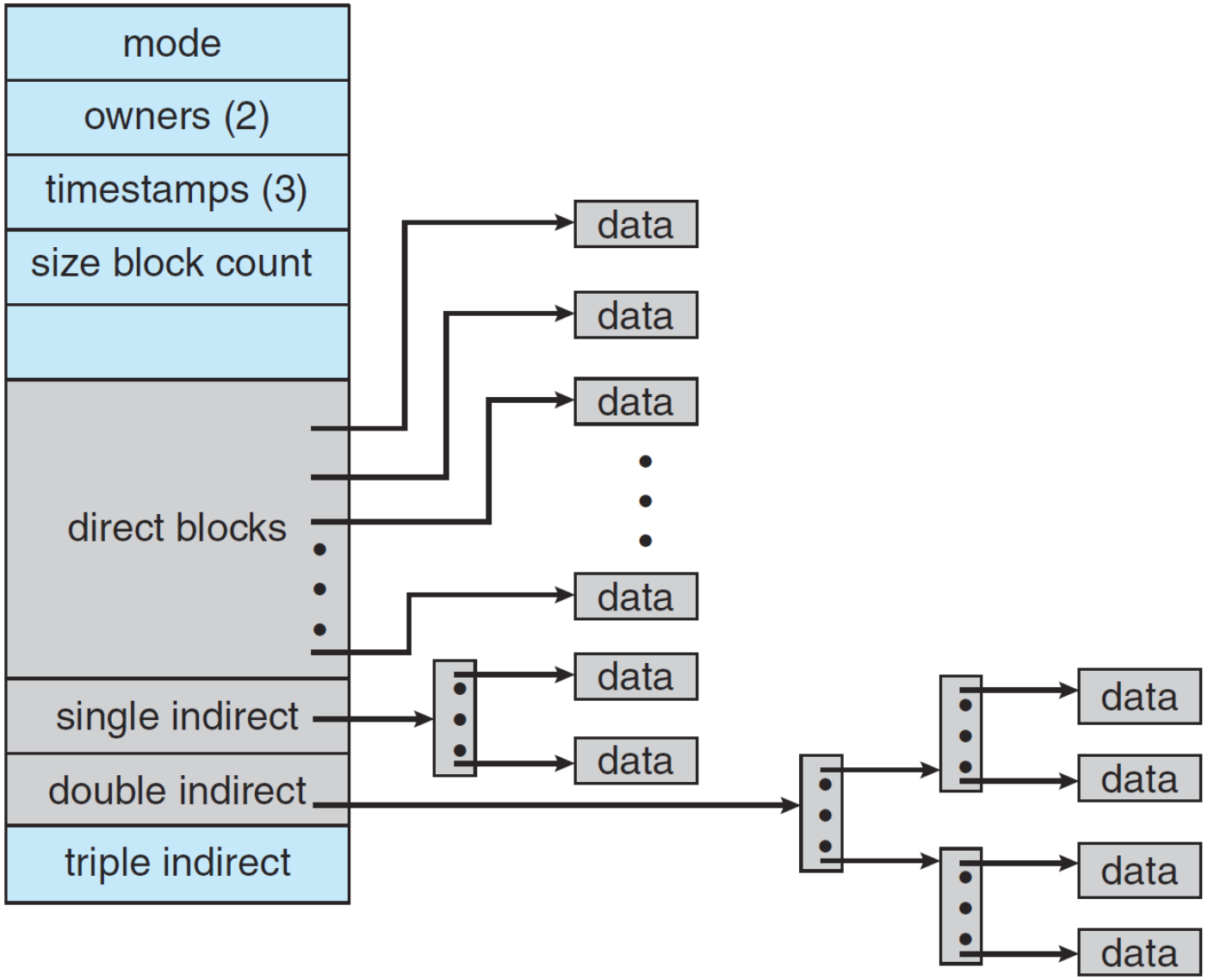
File Space Allocation

❑ Indexed allocation pros and cons

- Flexible to extend, insert, and delete blocks
- Index allocation supports both sequential and direct access without external fragmentation.
- Suffer from wasted space
  - ✓ The overhead of the index block is generally greater than the pointer overhead of linked allocation.
- Difficult to determine the size of a index block
  - ✓ Make the index block as small as possible to reduce the overhead
  - ✓ However, if the index block is too small, however, it will not be able to hold enough pointers for a large file.
  - ✓ Solution: 1) Have multiple index blocks and chain them into a linked-list
    2) Have multiple index blocks, but make them a tree just like the indexed access method
    3) A combination of both

File Space Allocation

- Indexed allocation (Multiple index blocks in UNIX-based file system)



Free Space Management

- How do we keep track free blocks on a disk?

- A free-list is maintained. When a new block is requested, we search this list to find one.

- The following are commonly used techniques:

  - ✓ Bit Vector
  - ✓ Linked List
  - ✓ Grouping
  - ✓ Linked List +Address+Count

Free Space Management

Bit Vector

- Each block is represented by a bit in a vector. Thus, if there are n disk blocks, the vector has n bits.
- If a block is free, its corresponding bit is 0.
- When a block is needed, the vector is searched.
- If the disk capacity is small, the whole bit vector can be stored in memory. For a large disk, this bit vector will consume too much memory.
- We could group a few blocks into a cluster and allocate clusters. This saves space and may cause internal fragmentation.

$$bit[i] = \begin{cases} 0 \Rightarrow block[i] \text{ free} \\ 1 \Rightarrow block[i] \text{ occupied} \end{cases}$$

Free Space Management

Bit Vector

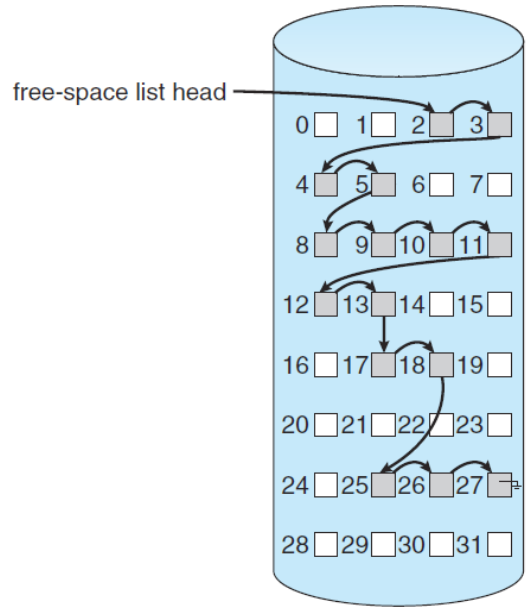
- Block size=4 KB=2^{12} Bytes
- Disk size=1 TB=2^{40} Bytes
- What is the size of the bit vector?

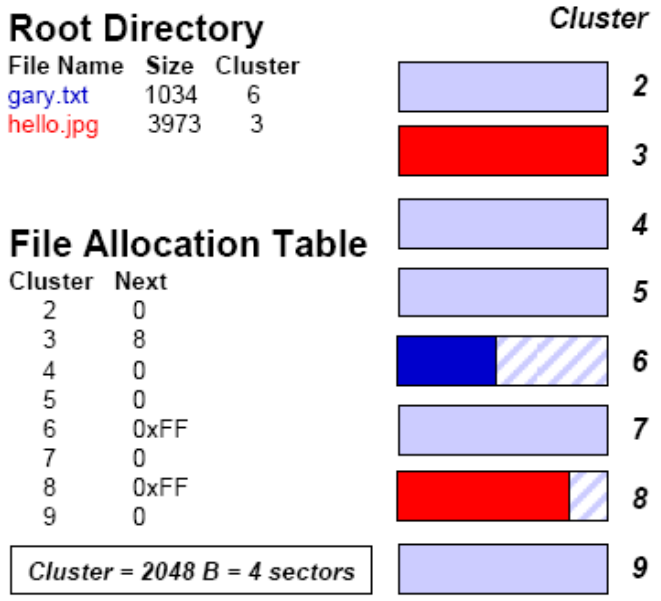
- Solution:
  - ✓ Total number of blocks=2^{40}/ 2^{12} =2^{28}
  - ✓ 2^{28} bits (=256 MB) are used to represent 2^{28} blocks.
  - ✓ That is, the size of the bit vector is 256 MB, which is normally stored in the memory
  - ✓ If we group 4 blocks into a cluster, size of the bit vector= 64 MB.

Free Space Management

Linked List

- Like the linked allocation method, free blocks can be chained into a linked list.
- When a free block is needed, the first in the chain is allocated.
- However, this method has the same disadvantages of the linked allocation method.
- We may use a FAT for the disk and chain the free block pointers together. Note that the FAT may be very large and consume space if it is stored in memory.

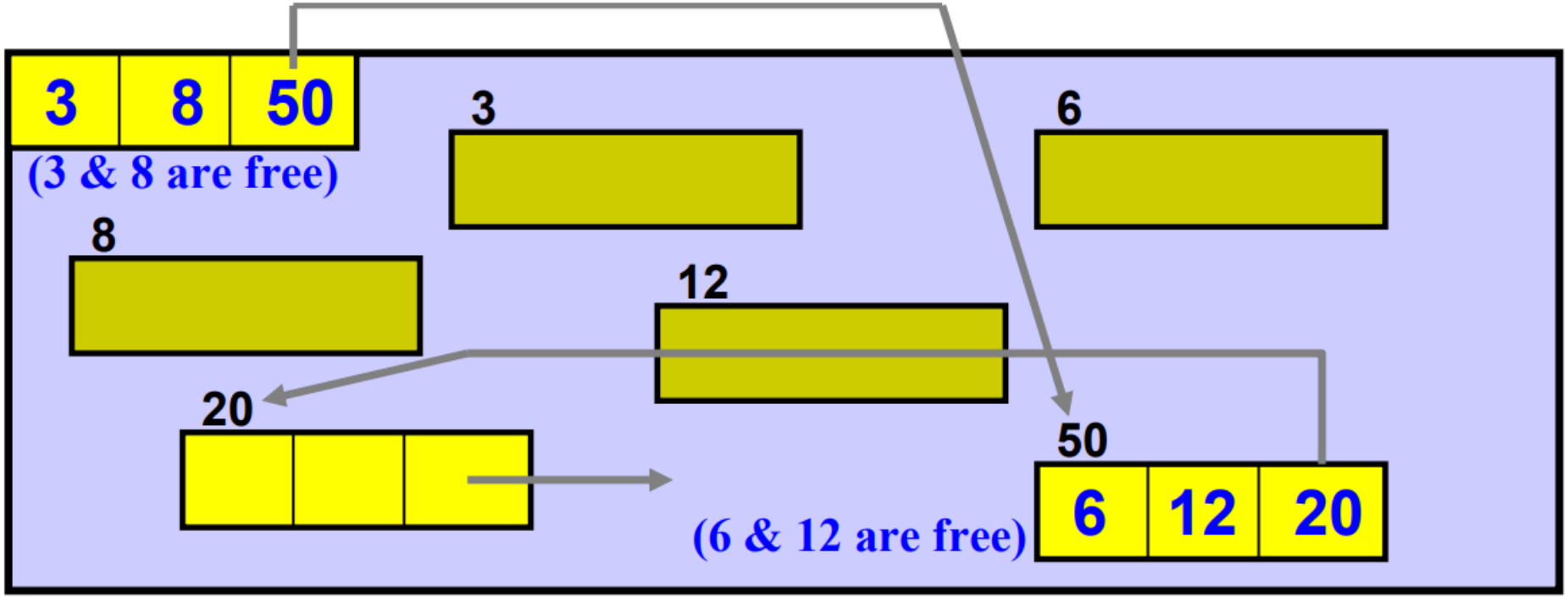




Free Space Management

❑ Grouping

- The first free block contains the addresses of n other free blocks..
- For each group, the first n-1 blocks are actually free and the last (i.e., n^{th}) block contains the addresses of the next group.
- In this way, we can quickly locate free blocks.



Free Space Management

Grouping

- Block size=4 KB=2^{12} Bytes
- Disk size=1 TB=2^{40} Bytes
- 32 bits are used to represent a block number
- How many blocks are needed to represent free blocks in the worst case scenario?

- Solution:

  - ✓ Each block can store $2^{12}/4=2^{10}$ number of block numbers
  - ✓ Each block can store $2^{10}-1$ number of free block numbers
  - ✓ In the worst case scenario, total number of free blocks= $2^{40} / 2^{12} = 2^{28}$
  - ✓ Number of needed blocks= $2^{28} /(2^{10}-1) \approx 2^{18}$, which need 2^{18} *4 KB=1 GB.

Free Space Management

❑ Address + Counting

- Blocks are often allocated and freed in groups
- We can store the address of the first free block and the number of the following n free blocks.