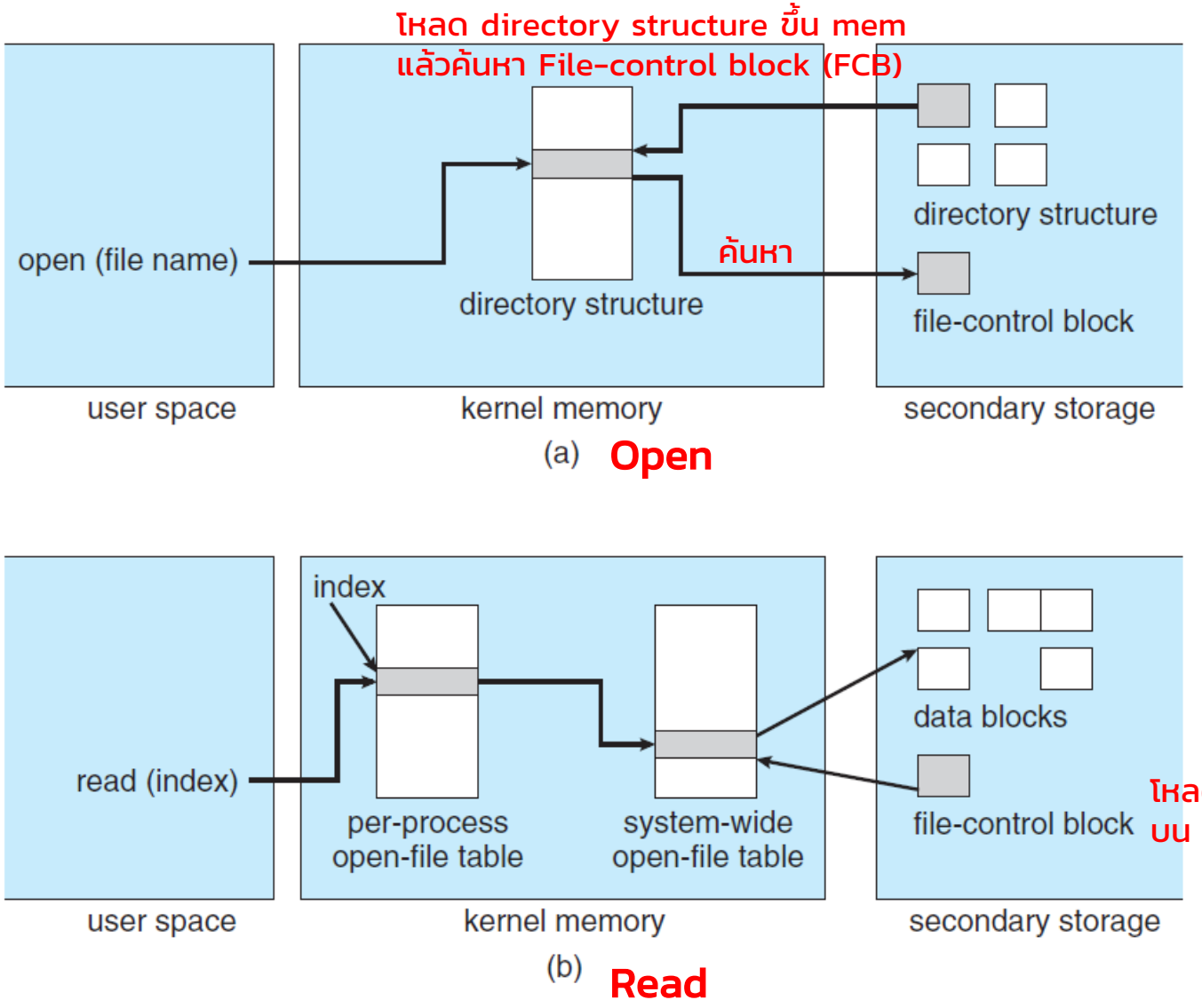
File-System Implementation



File-control block

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

ACL = access-control list

Figure 14.3 In-memory file-system structures. (a) File open. (b) File read.

Metadata

File-control block (FCB) = inode (in Linux)

Metadata vs. Actual Data

- Metadata
  - Ownership
  - Permissions
  - Location of the file contents

  etc.
- Actual data
  - File content

Directory Implementation

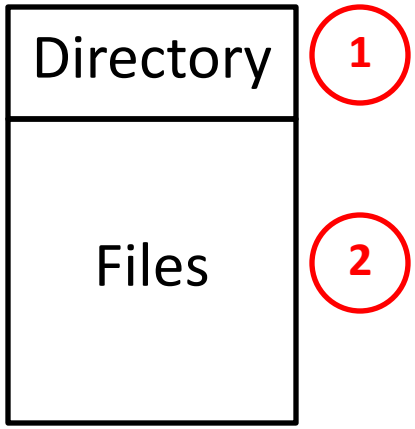
- Linear List
- Hash Table

How to store all entries (all files in the directory) (1)

Allocation Method

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

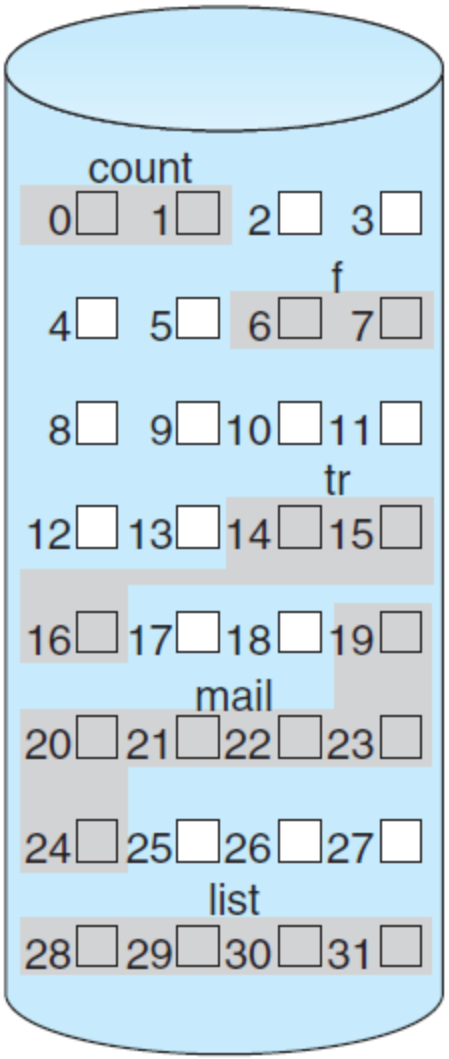
How to store each entry (each file) (2)



A partition (volume)

A directory can contain directories (nested structure).

Contiguous Allocation



directory

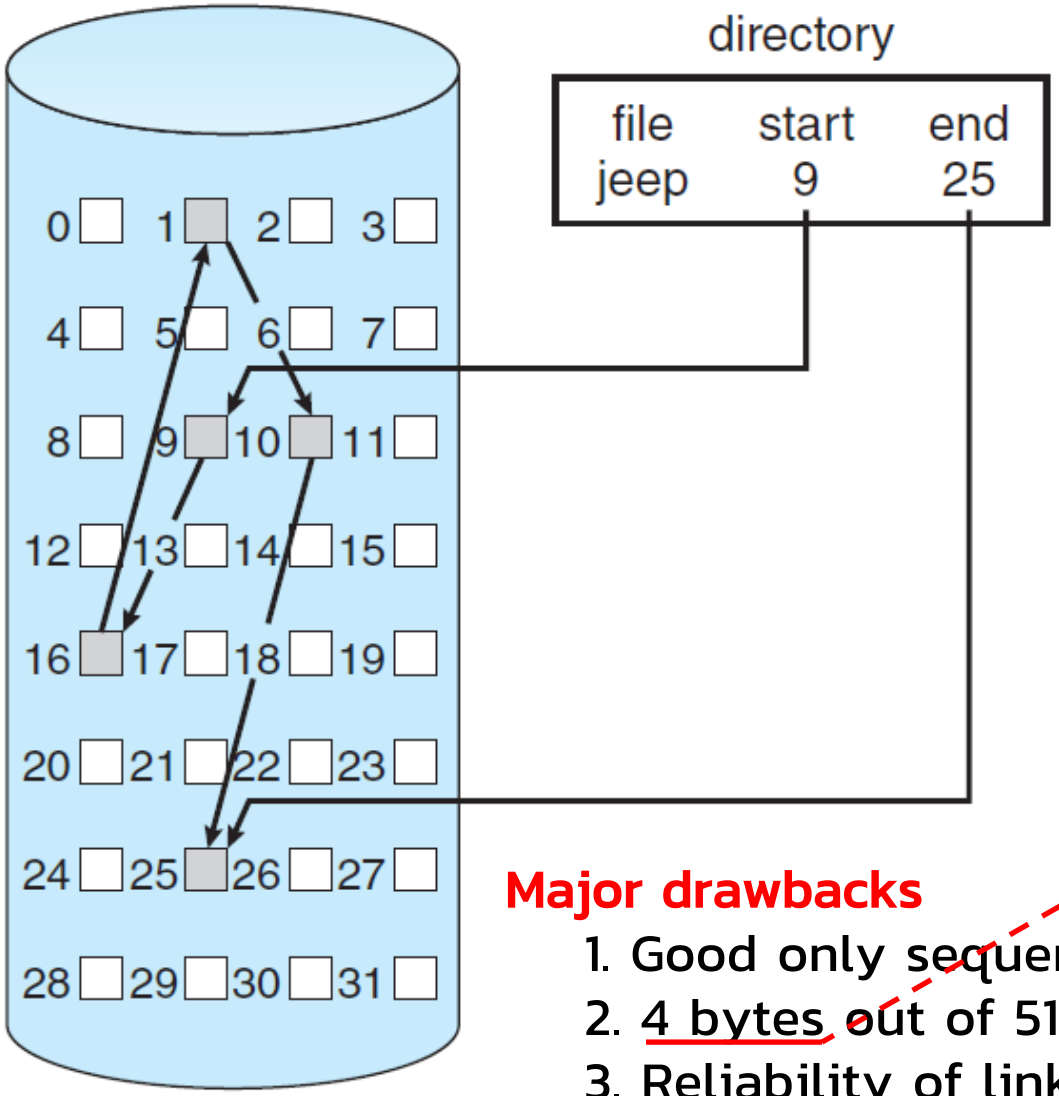
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Major drawbacks

1. External fragmentation.
2. File size must be declared.

Figure 14.4 Contiguous allocation of disk space.

Linked Allocation



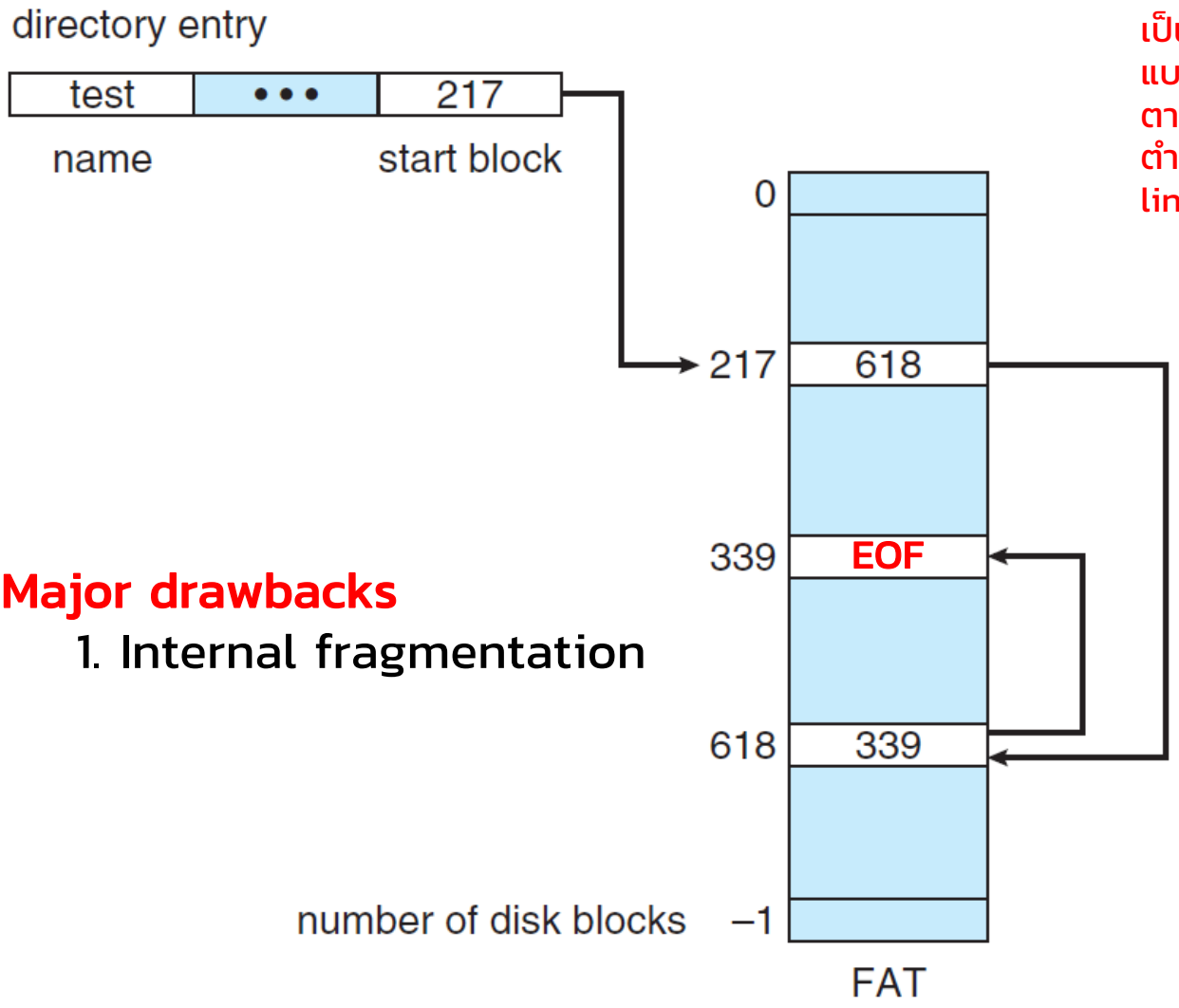
pointer ไปยัง block ถัดไป

Major drawbacks

1. Good only sequential access.
2. 4 bytes out of 512-byte block = 0.78%
3. Reliability of linked list

Figure 14.5 Linked allocation of disk space.

File Allocation Table (FAT)



เป็นแบบ linked allocation แต่ดีกว่าแบบ linked list ในสไลด์ก่อนหน้านี้ ตารางนี้โหลดขึ้น mem ได้ ทำให้ไปยังตำแหน่งที่ต้องการบนไฟล์ได้เร็วกว่าแบบ linked list ที่ต้องอ่านดิสก์ไปทีละ block

A table per volume
(not in the textbook)

Block (cluster)	Next
...	
217	618
...	
339	-1
...	
618	339

Major drawbacks

1. Internal fragmentation

Figure 14.6 File-allocation table.

ข้อเสียของ FAT

Average Cluster Efficiency
Note: Disk Size does not apply to FAT32 where the 4K cluster is usually used.

Cluster Size	Efficiency	Disk Size (applies to standard FAT only)
2K	98.4%	0-127 MB
4K	96.6%	128-255 MB
8K	92.9%	256-511 MB
16K	85.8%	512-1023 MB
32K	73.8%	1024-2047 MB
64K	56.6%	2047 MB >

เพิ่ม cluster size ตาม disk size เพื่อให้ table มีขนาดคงที่ จะได้ไม่เปลือง memory พอ cluster ใหญ่ จะเกิด internal fragmentation ทำให้ efficiency ลด

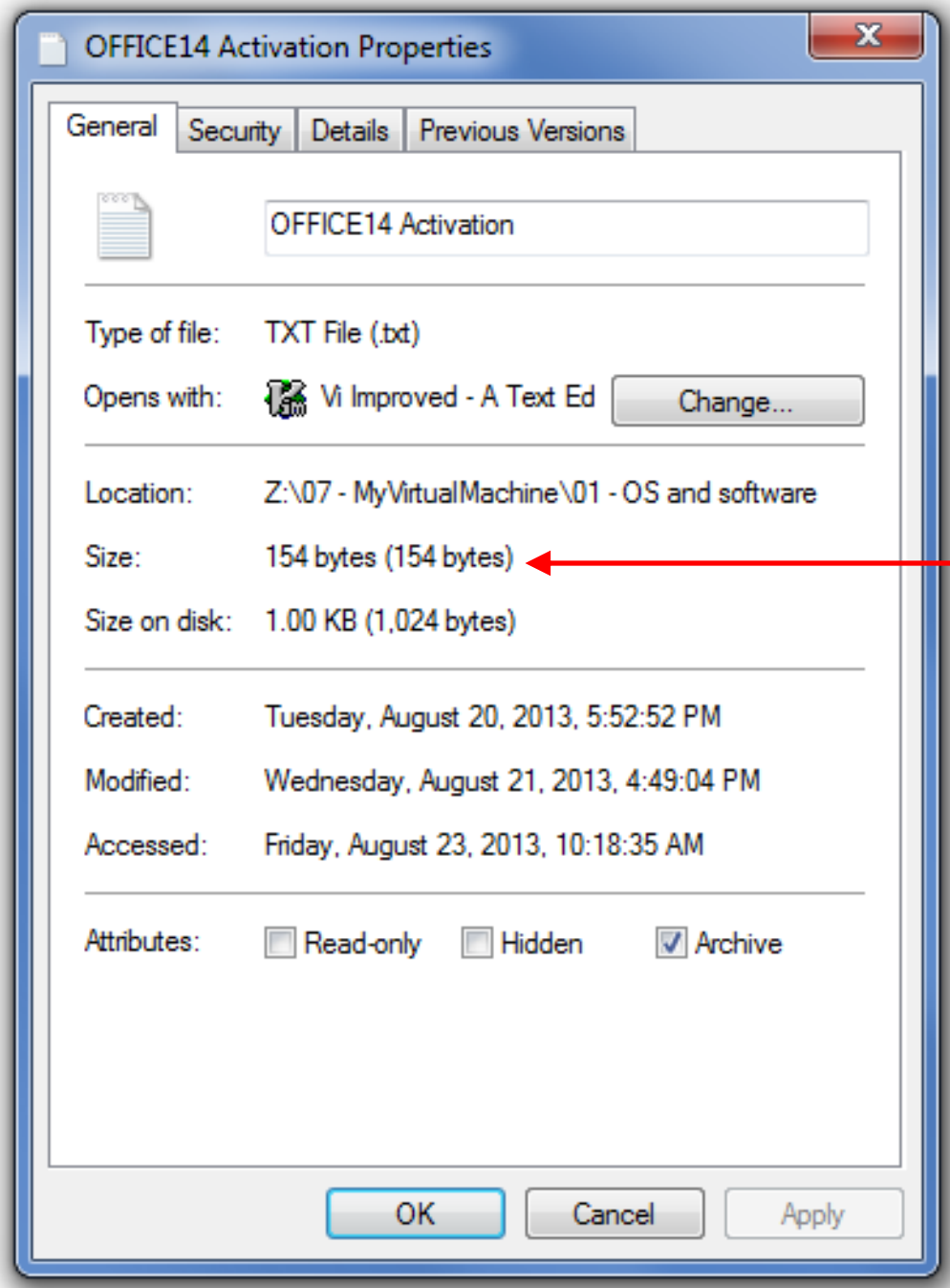
จะไม่ใช้ FAT32 กับ partition ที่ใหญ่มาก ๆ ต้องแบ่งเป็นหลาย ๆ partition ปัจจุบันใช้ NTFS หมดแล้ว FAT32 เหมาะกับ flash drive ที่มีขนาดเล็ก

FAT32 Cluster Sizes and Efficiency
Note: The minimum size for a FAT32 partition is about 260 MB.

FAT32

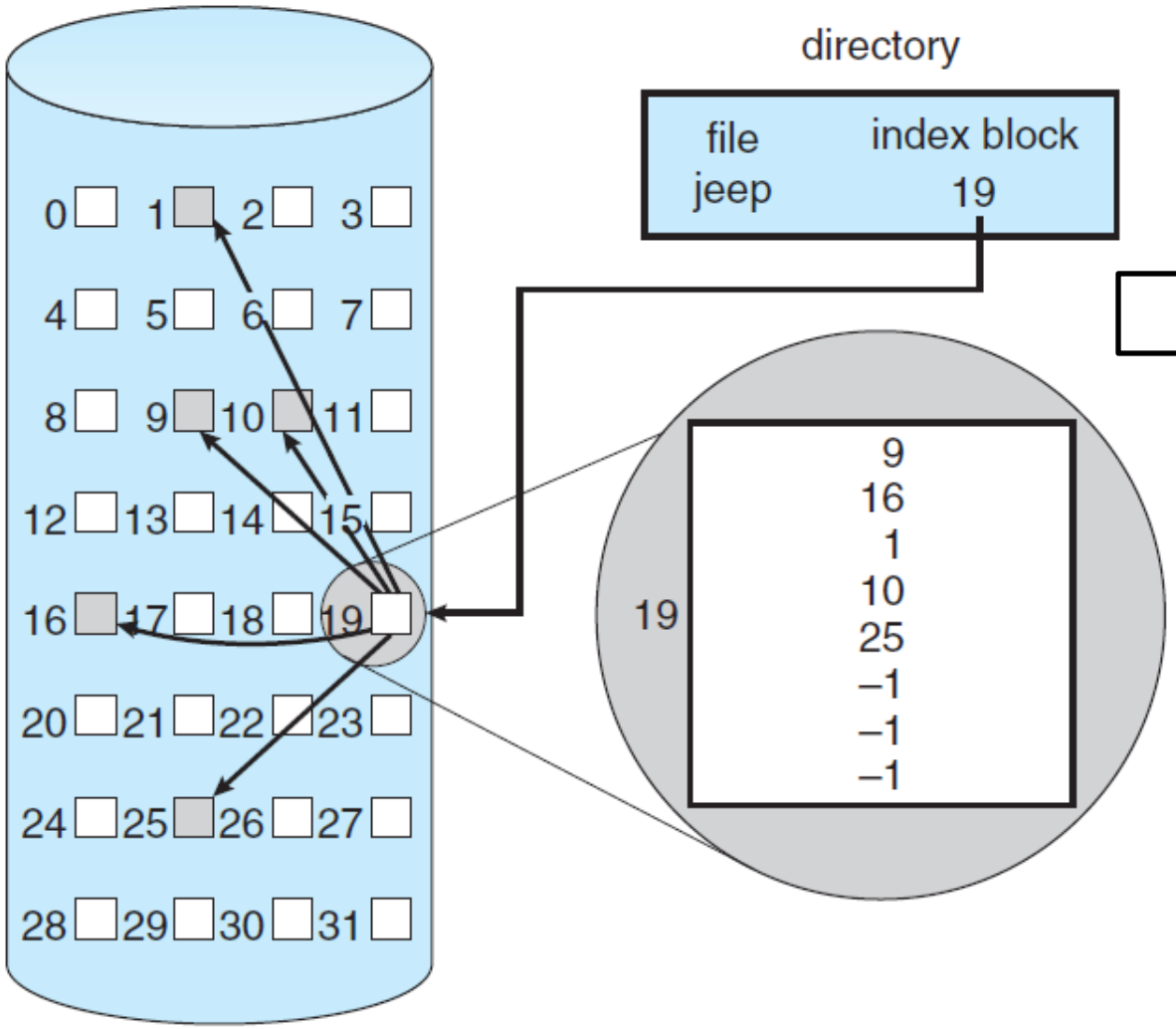
Disk Size	Cluster Size	Efficiency
> 260meg	4K	96.6%
> 8gig	8K	92.9%
> 60gig	16K	85.8%
> 2tril	32K	73.8%

http://www.project9.com/fat32/



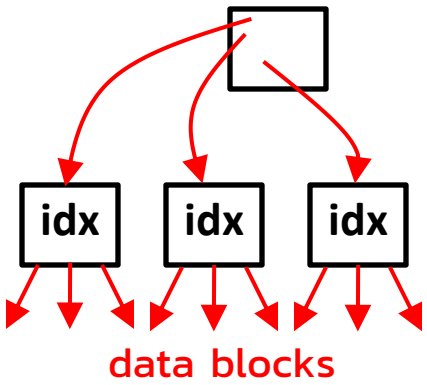
Internal Fragmentation
เพราะต้อง allocate ทีละ 1 block

Indexed Allocation



ถ้า index block ไม่พอก็ขยายออกไป

แบบ multi-indexed



แบบ multi-indexed ดีกว่า
เพราะค้นแค่ใน 2 block
ก็จะหา data block ที่ต้องการได้

Figure 14.7 Indexed allocation of disk space.

How to deal with very large files

- Linked scheme

  Slow access due to linked structure.
- Multilevel index

  Fast access (jump to a desired byte quickly)

  but waste 2 blocks for a small file.
- Combined scheme

  The first 15 pointers are in inode.

  12 pointer to direct blocks.

  1 pointers to single indirect blocks.

  1 pointers to double indirect blocks.

  1 pointers to triple indirect blocks.

Ex. block ละ 4k bytes pointer มีขนาด 4 bytes
1 ไฟล์จุได้มากสุด
12 x 4k = 49,152
(4k/4) * 4k = 4,194,304
(4k/4) * (4k/4) * 4k = 4,294,967,296
(4k/4) * (4k/4) * (4k/4) * 4k = 4,398,046,511,104
รวม 4,402,345,721,856 bytes ≈ 4 TB

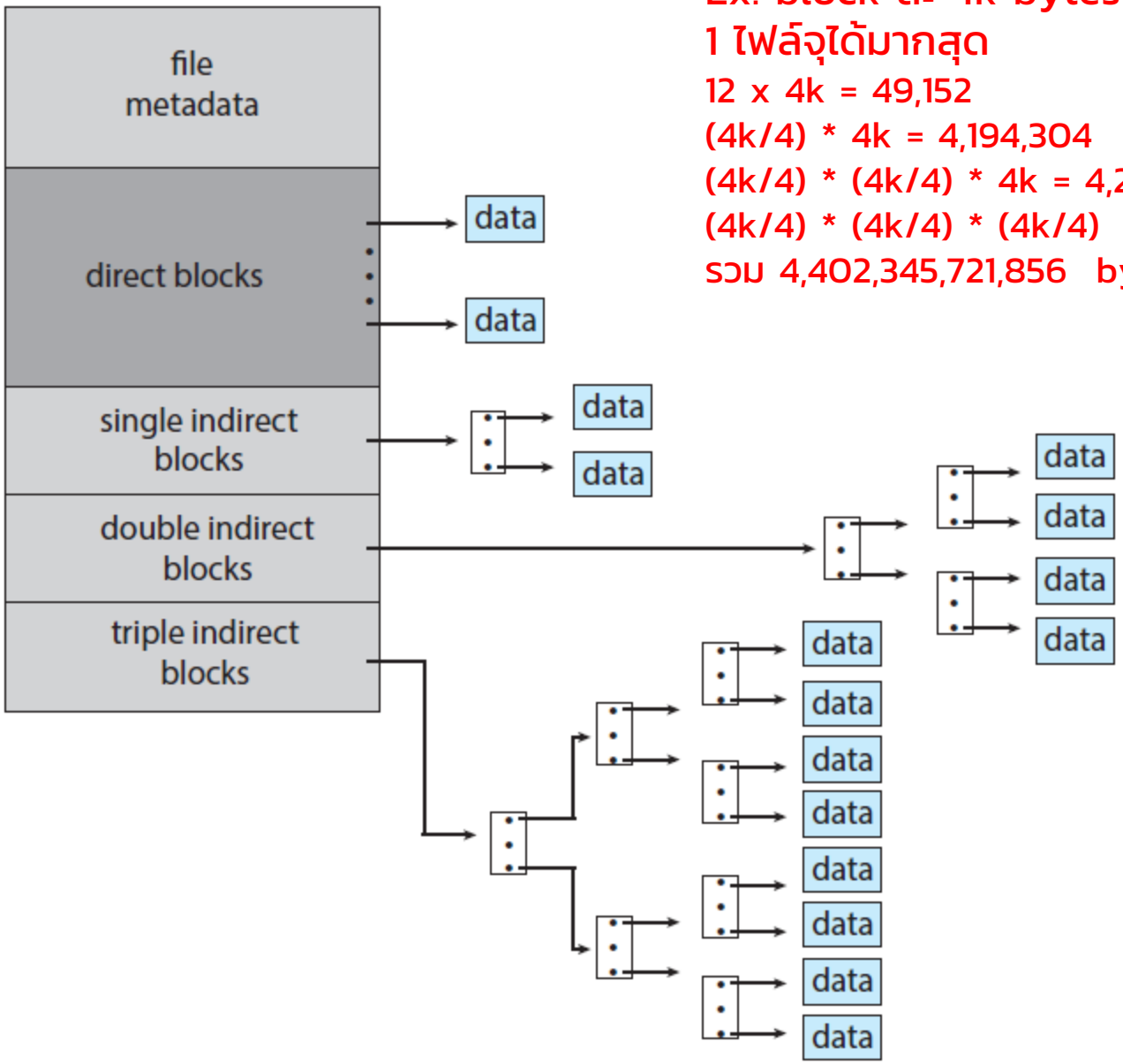


Figure 14.8 The UNIX inode.

Free-Space Management

$$\text{1TB}\ \frac{2^{42}}{\text{4KB}\ 2^{12}} \times \frac{1}{8\ \text{1 byte}} = 2^{15}\ \text{32 MB}$$

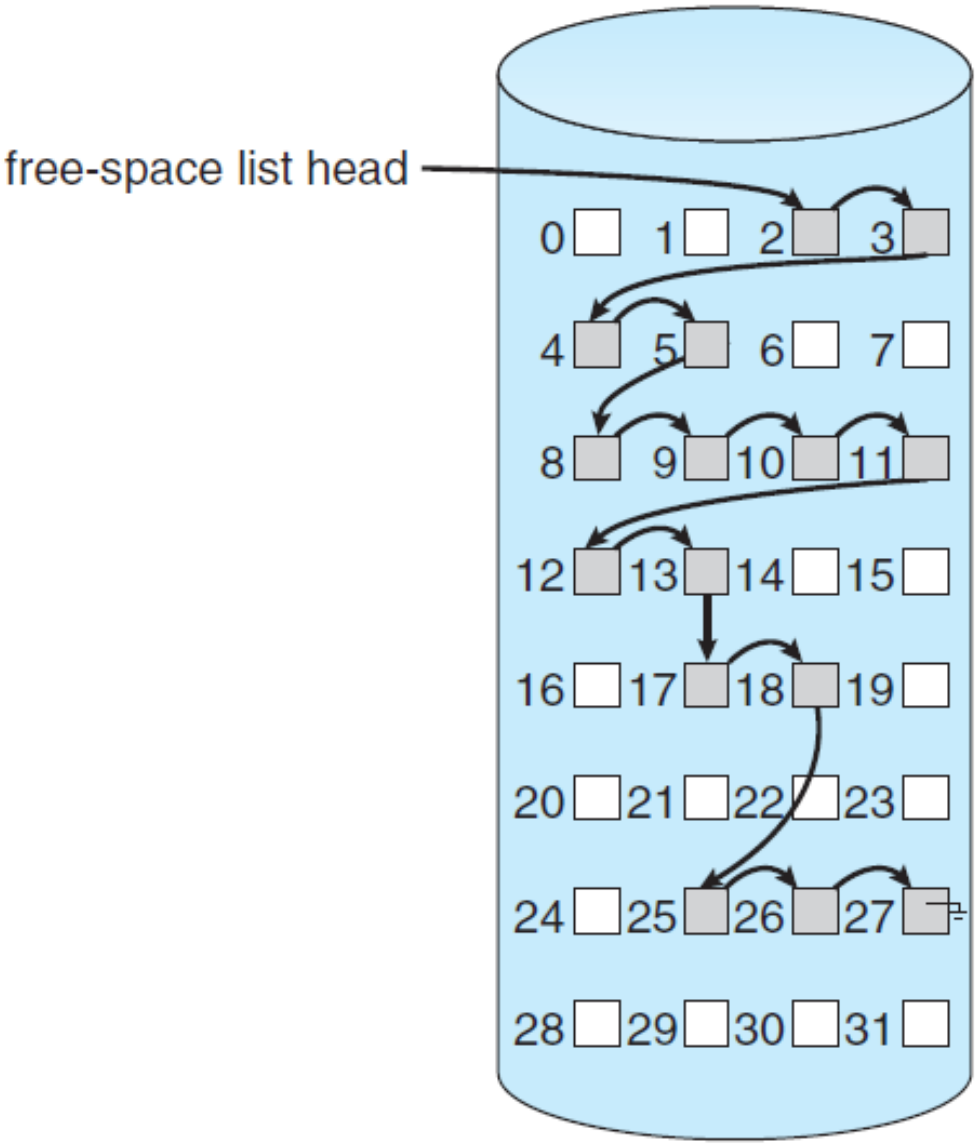
เก็บได้ 8 blocks

- Bit vector
  1-TB disk with 4KB blocks requires 32 MB.
- Linked List
  Not efficient, require substantial I/O.
  However, not often.
- Grouping (ดูสไลด์ถัดไป)
  The first block contains n – 1 free blocks.
  The last block contains another n – 1 free blocks
  and so on.
- Counting (ดูสไลด์ถัดไป)
- Space Maps (ใน ZFS)
  แบ่ง space เป็น metaslab แต่ละ metaslab มี data structure ชื่อ space map มีขนาดเล็กและใช้ I/O น้อยกว่าวิธี counting

ZFS เป็น file system ของระบบปฏิบัติการ Oracle Solaris, 1 zettabytes (ZB) = 1000^7 bytes

Grouping

ข้อดีคืออ่านแค่ block เดียวก็ได้ free block มาใช้จำนวนมาก (1 block เก็บ pointer ได้หลายตัว เช่น 4k / 4 = 1,024



block 2

3,4,5,8,9

block 9

10,11,12,13,17

block 17

18,25,26,27,-1

Figure 14.9 Linked free-space list on disk.

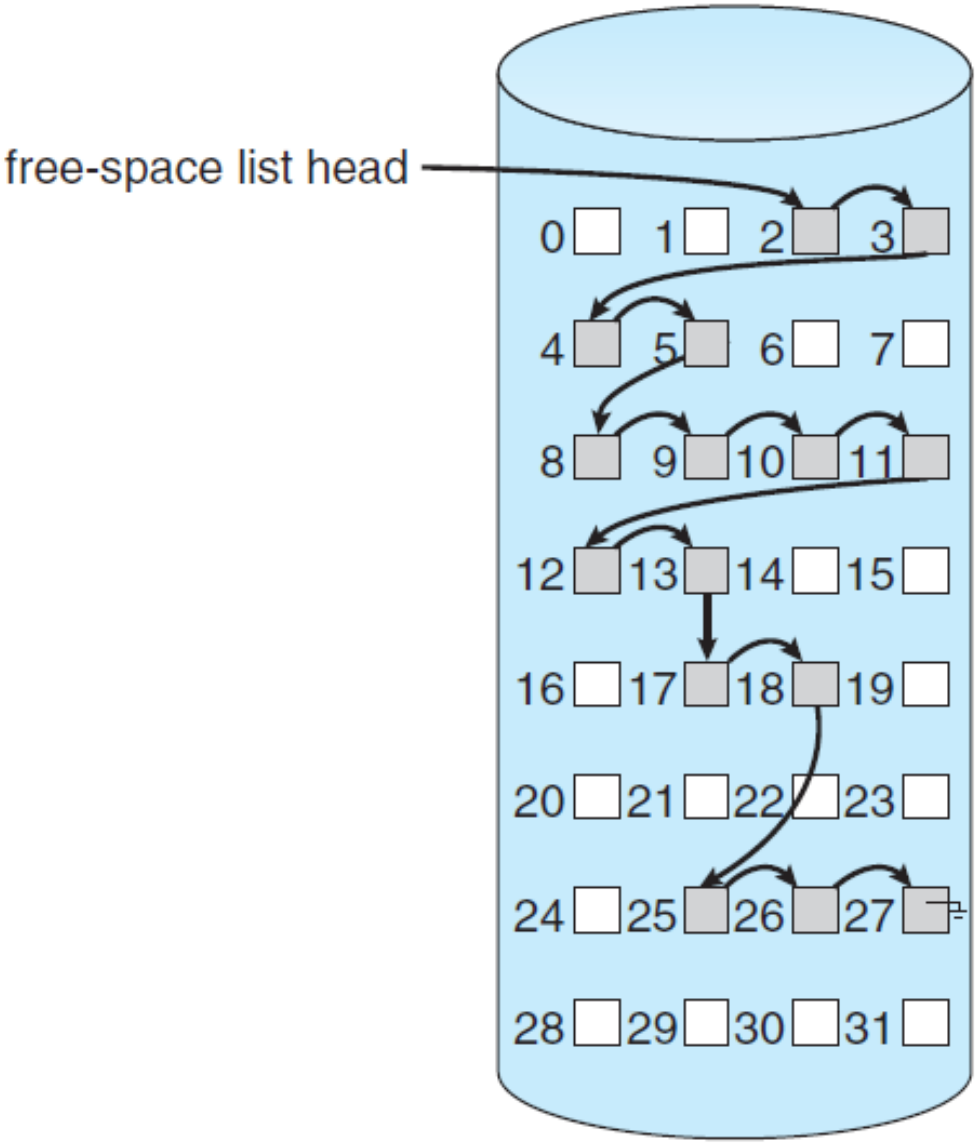


Figure 14.9 Linked free-space list on disk.

Counting

contiguous blocks

block	count
2	4
8	6
17	2
25	3

ถ้าที่ว่างอยู่กระจัดกระจาย
ตารางนี้ก็จะใหญ่มาก

List of File Systems

Contents [hide]

https://en.wikipedia.org/wiki/List_of_file_systems

Linux

ext3, ext4

Windows

FAT32, NTFS

macOS, iOS

HFS Plus or HFS+

Apple File System (APFS) ปี 2017

เป็นสิบปีจะออกเวอร์ชันใหม่สักครั้ง ไม่เปลี่ยนบ่อย ๆ

exFAT คืออะไร?

Journaling File System

A **journaling file system** is a file system that keeps track of changes not yet committed to the file system's main part by recording the intentions of such changes in a data structure known as a "journal", which is usually a circular log. In the event of a system crash or power failure, such file systems can be brought back online more quickly with a lower likelihood of becoming corrupted.[1][2]

For example, deleting a file on a Unix file system involves three steps:

1. Removing its directory entry.
2. Releasing the inode to the pool of free inodes.
3. Returning all disk blocks to the pool of free disk blocks.

If a crash occurs after step 1 and before step 2, there will be an orphaned inode and hence a storage leak; if a crash occurs between steps 2 and 3, then the blocks previously used by the file cannot be used for new files, effectively decreasing the storage capacity of the file system.

https://en.wikipedia.org/wiki/Journaling_file_system

Chapter 14 File-System Implementation

Chapter 15 File-System Internals