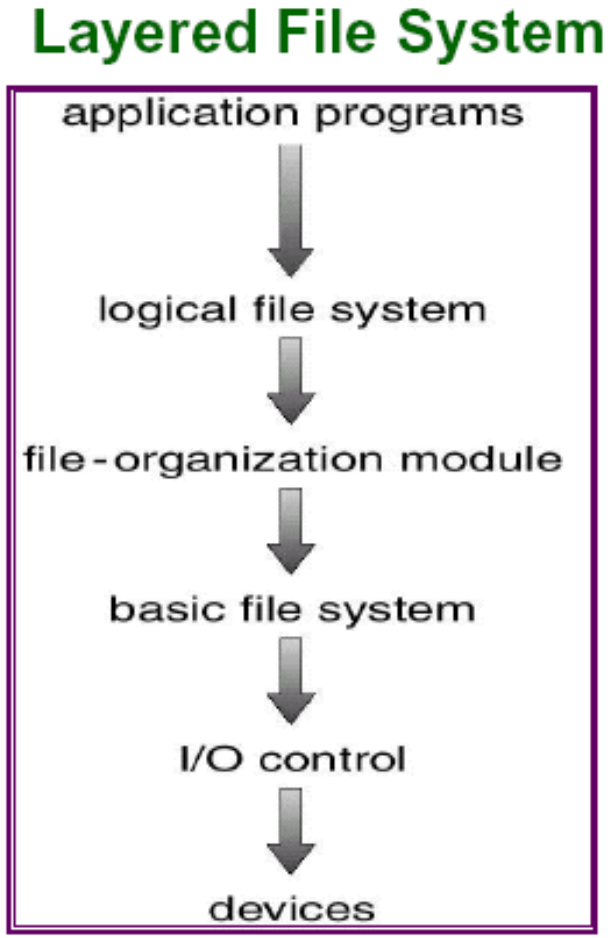
Hard Disk and File Systems

CE334

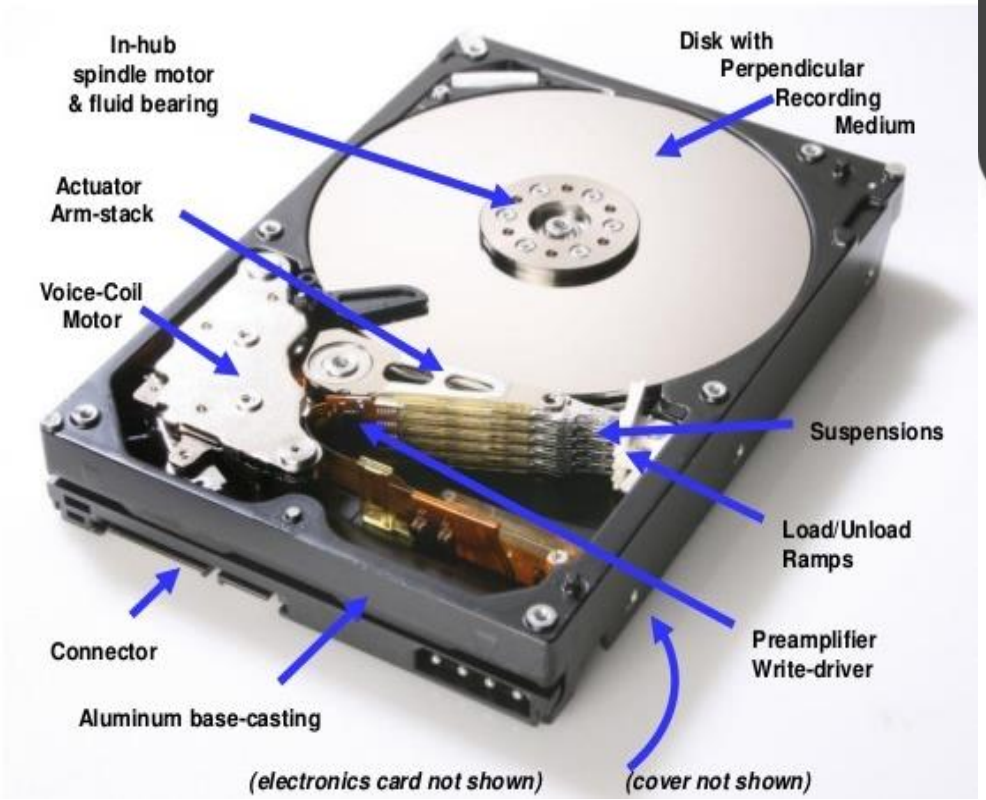
Chaiyaporn Khemapatapan

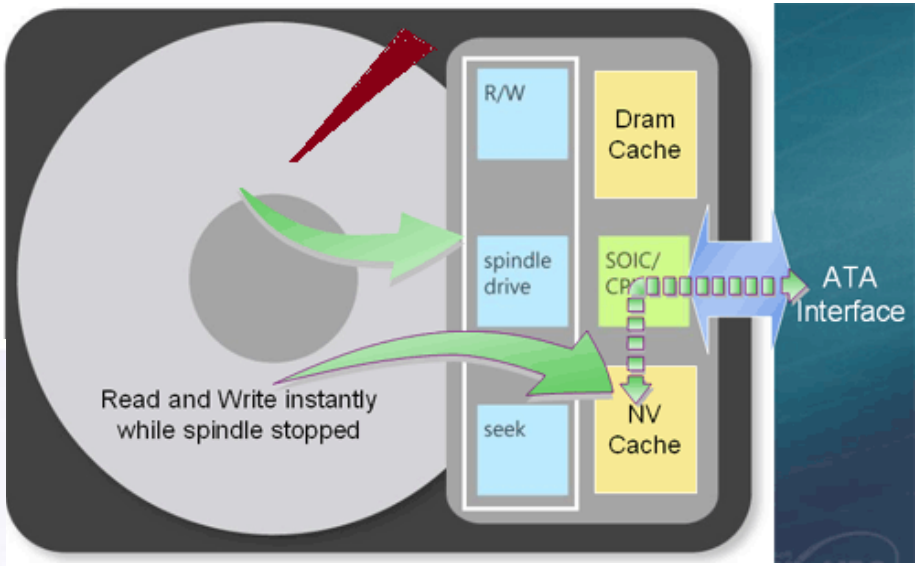
Layered File System

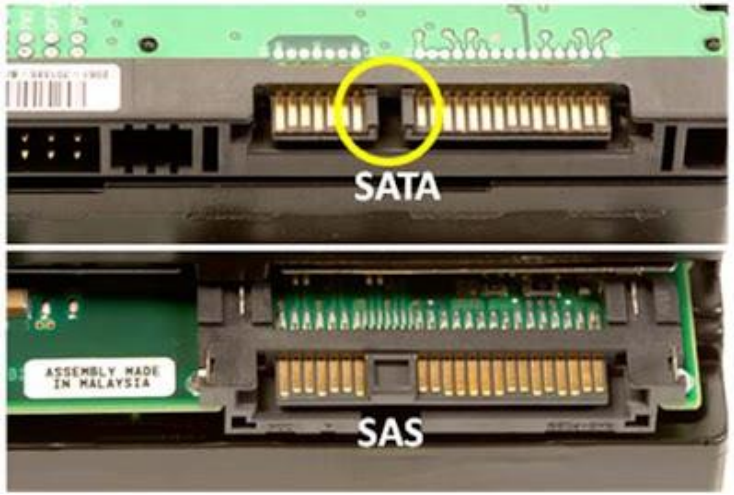


- Logical File System
  - Maintains file structure via FCB (file control block)
- File organization module
  - Translates logical block to physical block
- Basic File system
  - Converts physical block to disk parameters (drive 1, cylinder 73, track 2, sector 10 etc)
- I/O Control
  - Transfers data between memory and disk

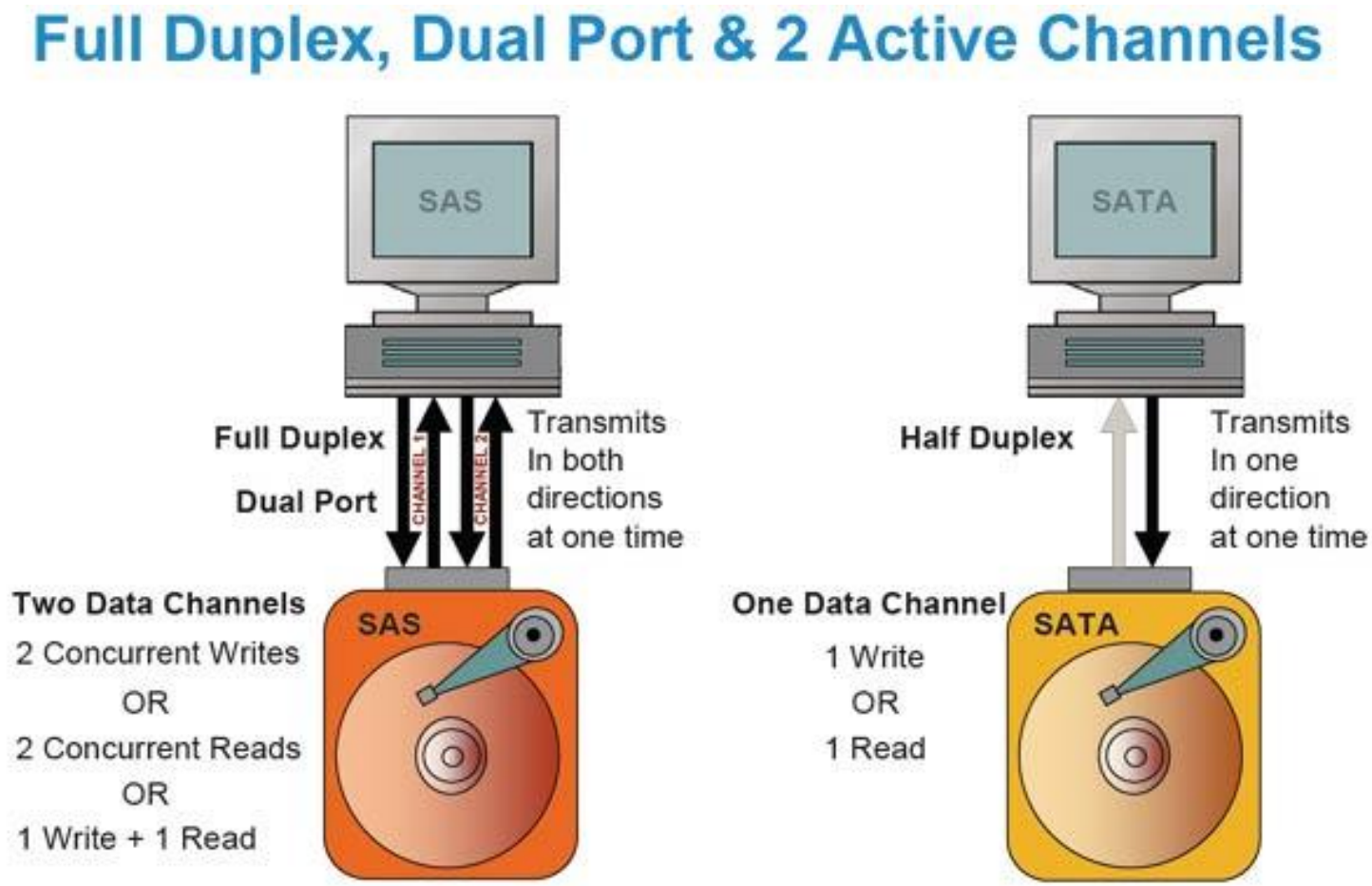
Hard Disk


In-hub
spindle motor
& fluid bearing
Disk with
Perpendicular
Recording
Medium
Actuator
Arm-stack
Voice-Coil
Motor
Suspensions
Load/Unload
Ramps
Connector
Preamplifier
Write-driver
Aluminum base-casting
(electronics card not shown)
(cover not shown)

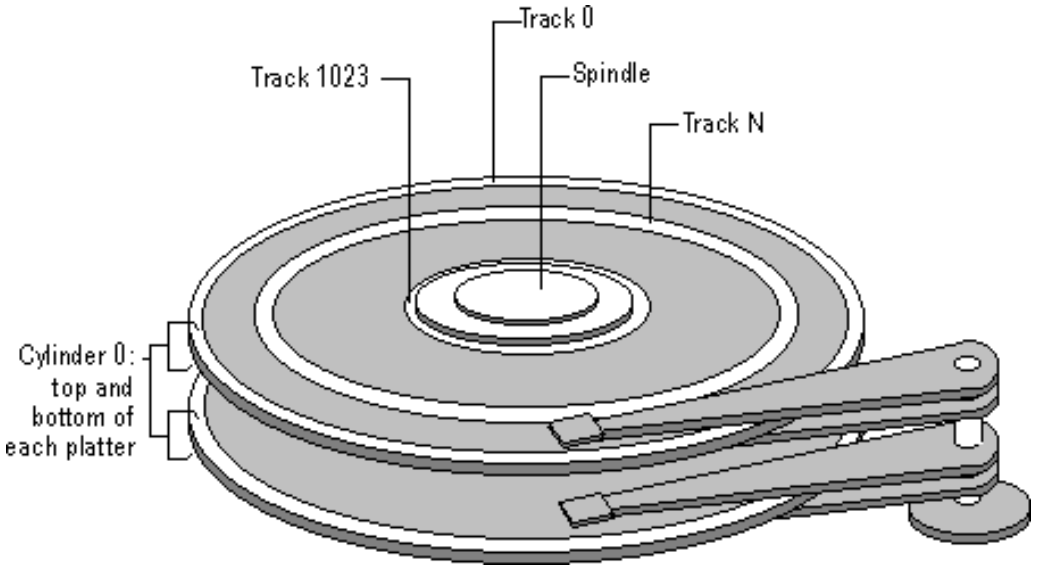

R/W
Dram
Cache
spindle
drive
SOIC/
CP
ATA
Interface
Read and Write instantly
while spindle stopped
seek
NV
Cache

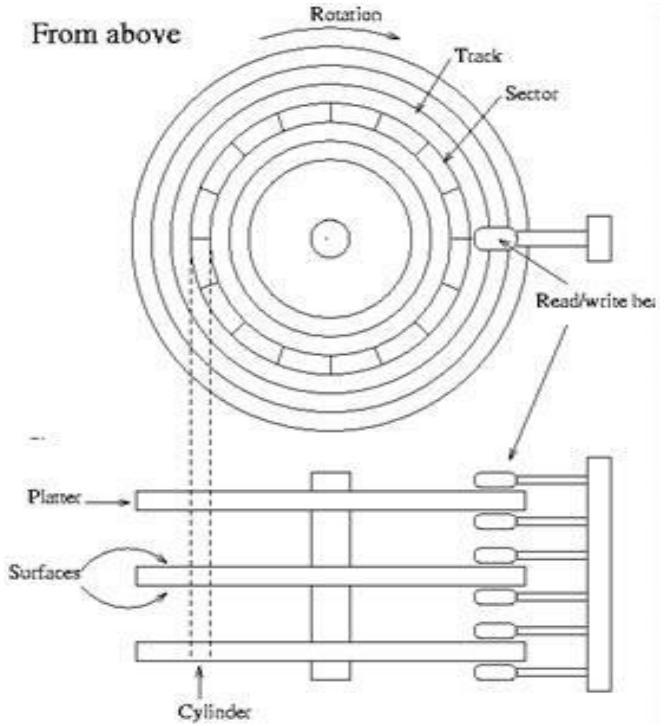

SATA
ASSEMBLY MADE
IN MALAYSIA
SAS

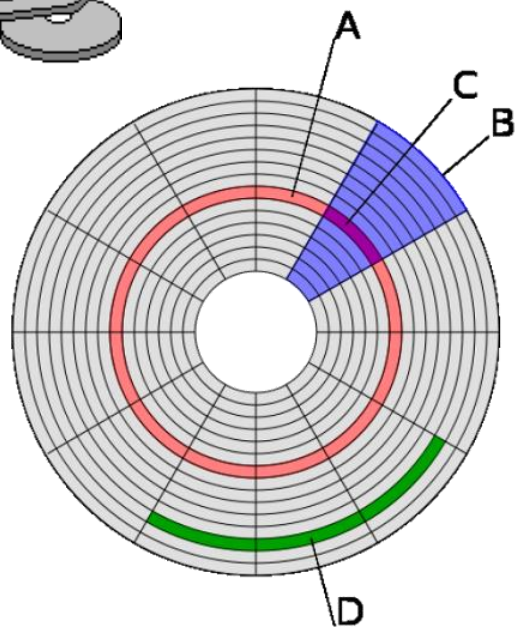
SAS Vs SATA


Full Duplex, Dual Port & 2 Active Channels
SAS
Full Duplex
Dual Port
CHANNEL 1
CHANNEL 2
Transmits
In both
directions
at one time
Two Data Channels
2 Concurrent Writes
OR
2 Concurrent Reads
OR
1 Write + 1 Read
SAS
SATA
Half Duplex
Transmits
In one
direction
at one time
One Data Channel
1 Write
OR
1 Read
SATA

Sector / Track / Cluster / Cylinder



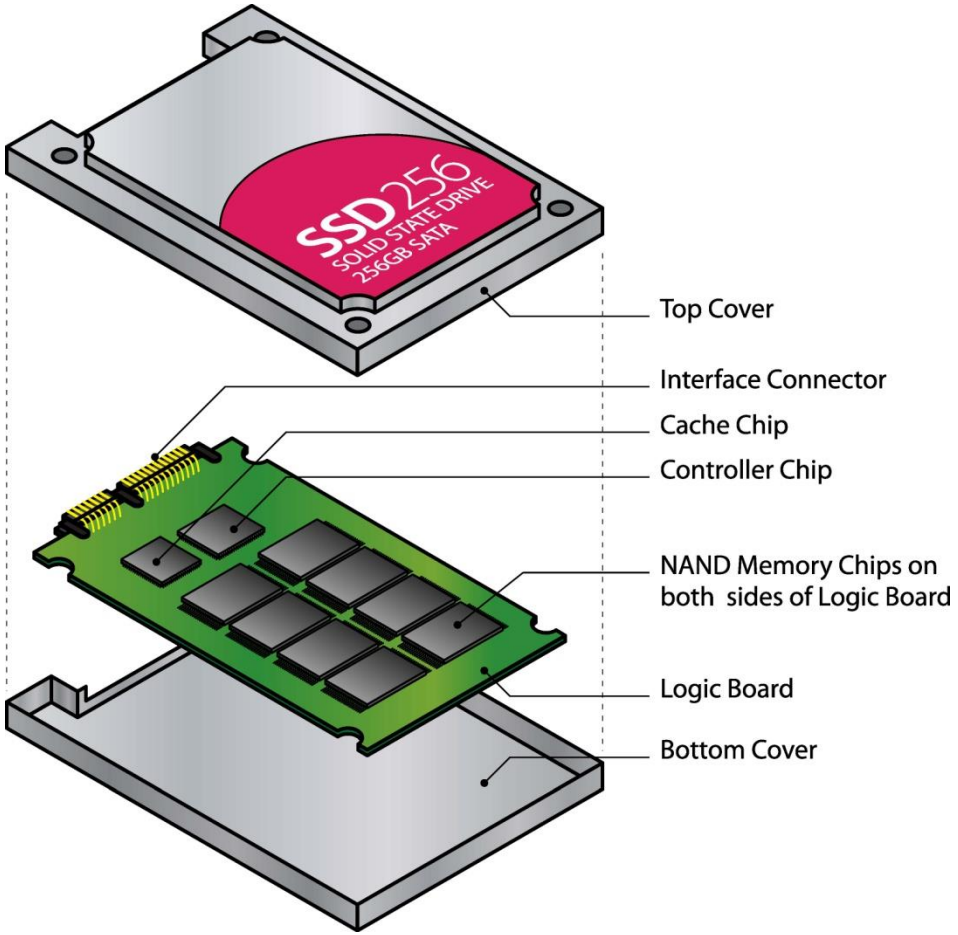


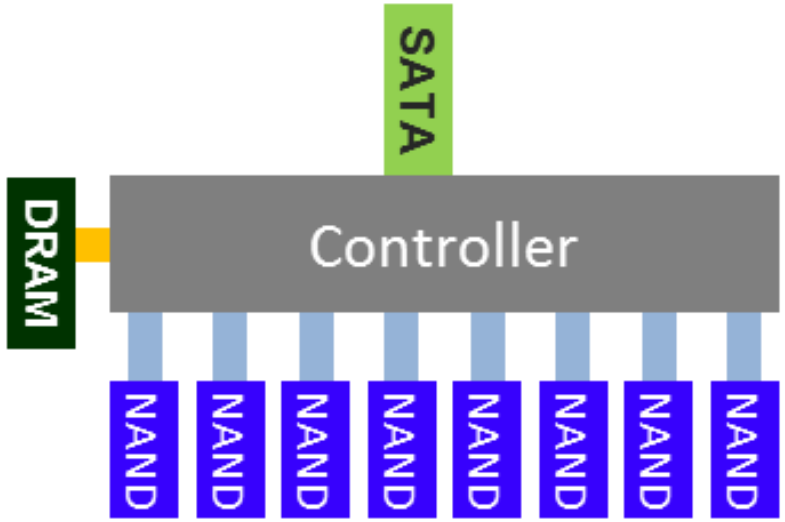


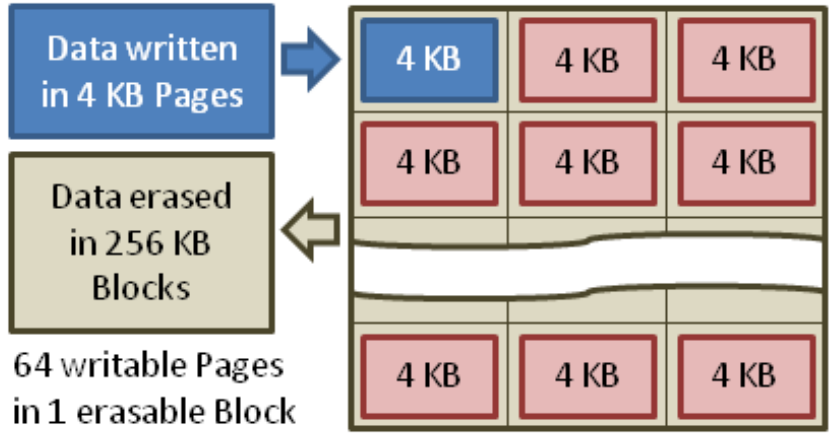
Hard Drive Structure:

A = track
B = sector
C = sector of a track
D = cluster

SSD



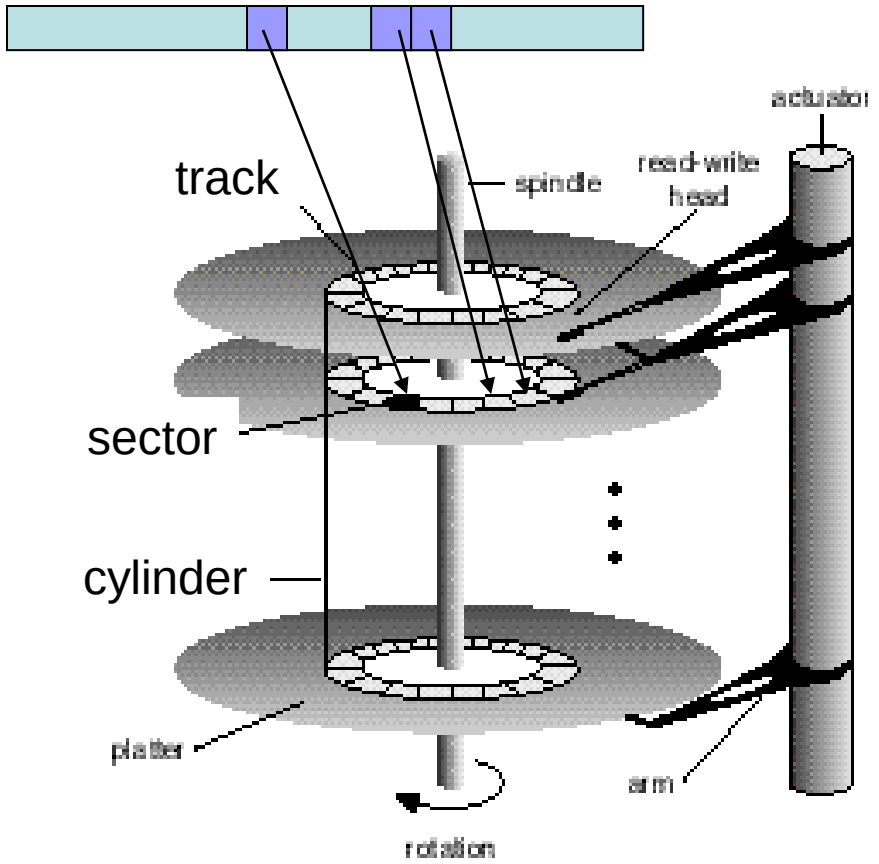




Typical NAND Flash Pages and Blocks

File System Structure

- **File system interface** provides applications with various system calls and commands such as open, write, read, seek, etc..
- **File system** maintains disk space in blocks and allocates available blocks to each stream-oriented file.
- **Basic file system (BIOS)** maintains data in physical blocks
- **Disk driver** reads from and writes to disk in a unit of block which consists of one (or more) sector(s).
- **Disk** maintains data locations with drive#, cylinder#, track# and sector#



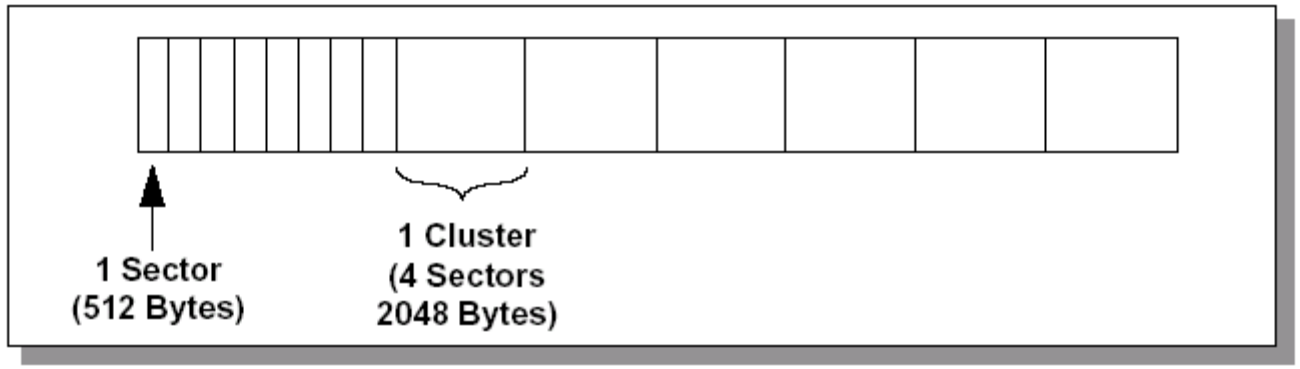
File system Units

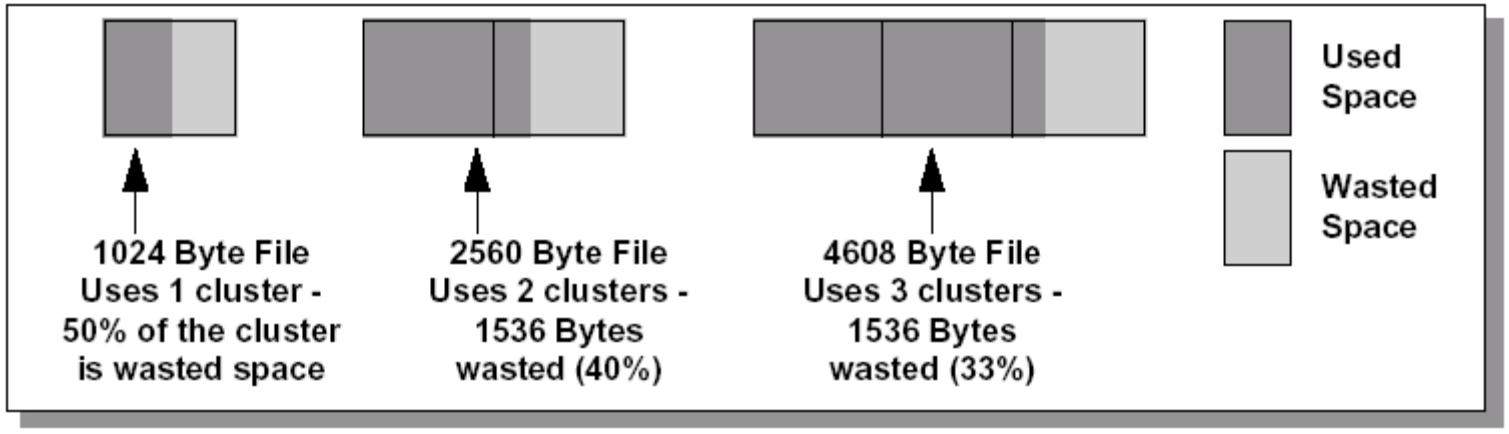
- **Sector** – the smallest unit that can be accessed on a disk (typically **512 bytes**)

- **Block(or Cluster)** – the smallest unit that can be allocated to construct a file

- What's the actual size of 1 byte file on disk?
  - takes at least one cluster,
  - which may consist of 1~8 sectors,
  - thus 1byte file may require ~4KB disk space.

Sector -> Cluster -> File layout



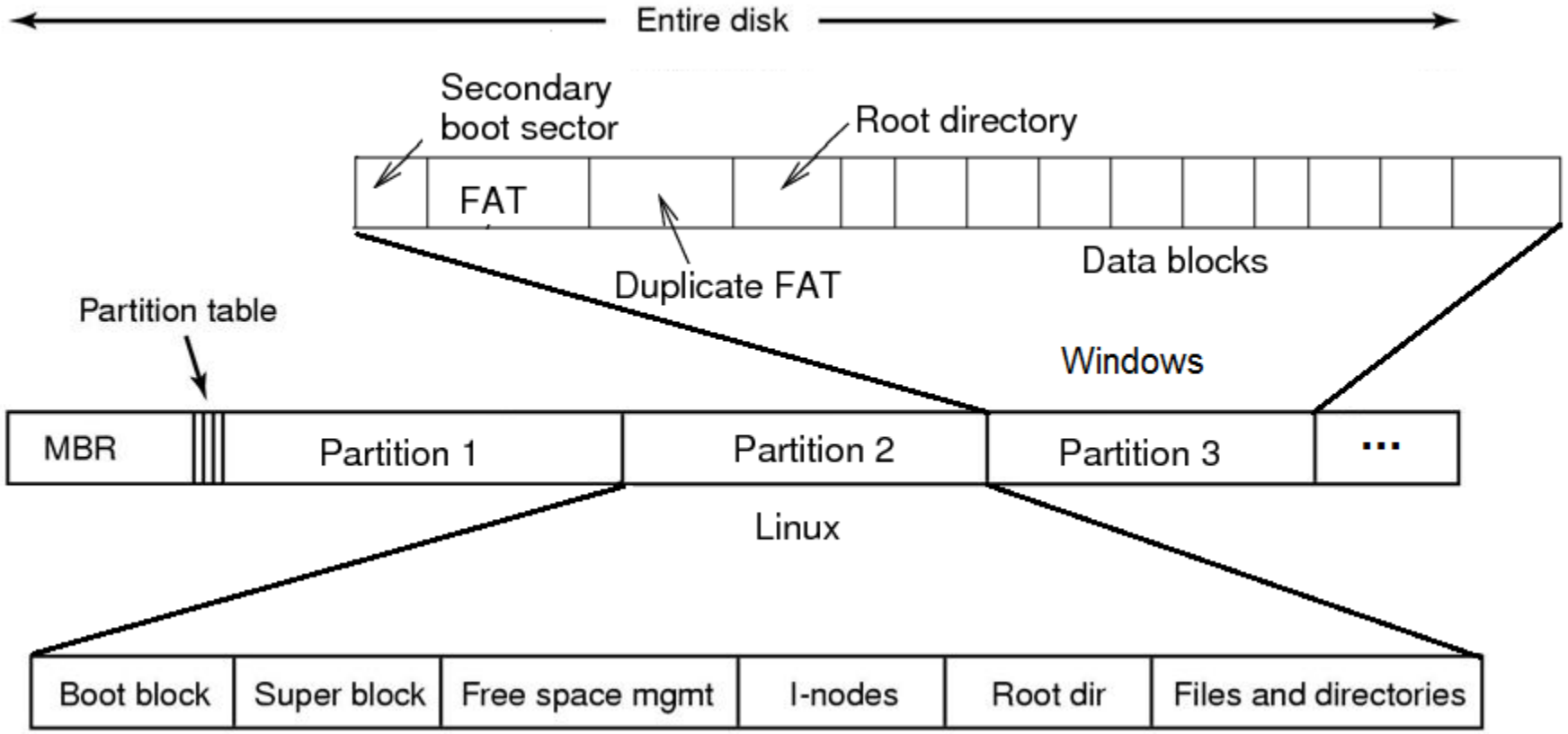


FCB – File Control Block

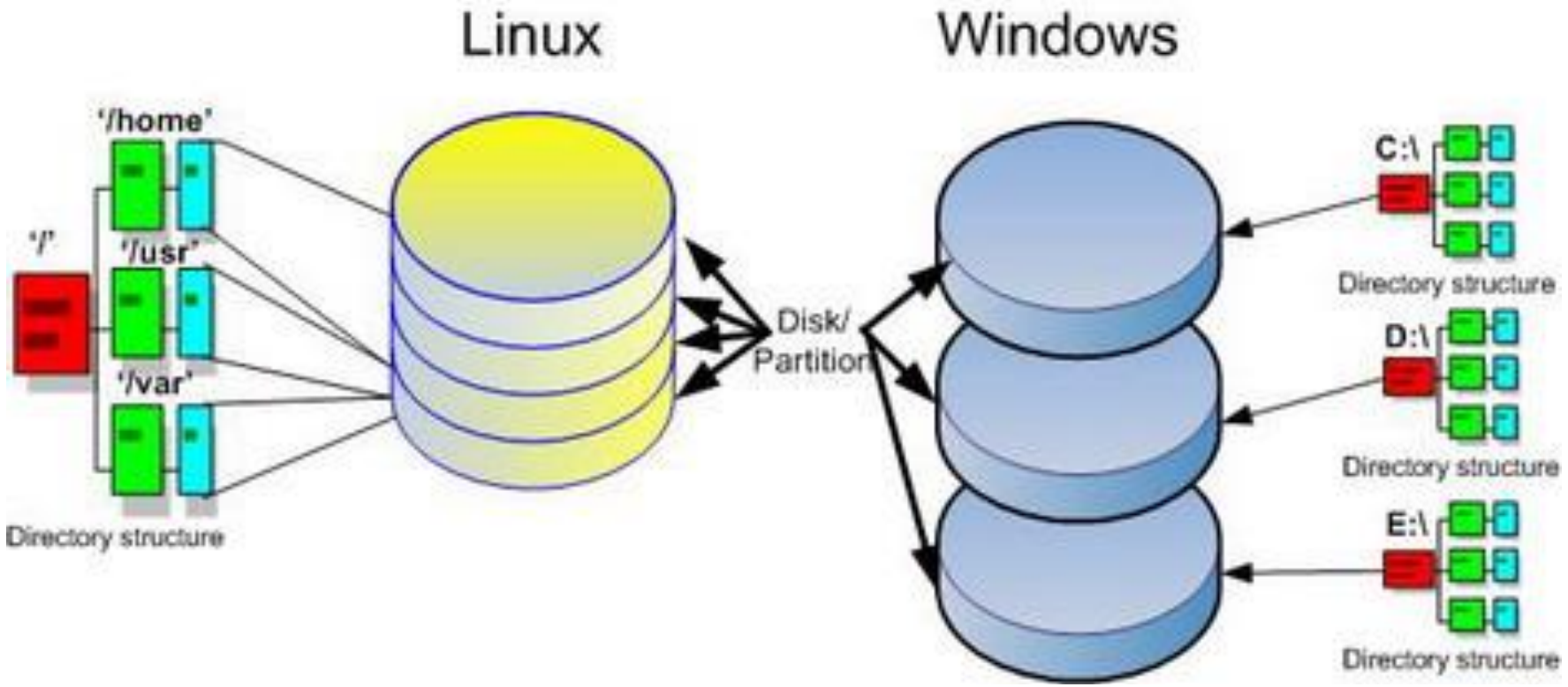
- Contains file attributes + block locations + information
  - Permissions
  - Dates (create, access, write)
  - Owner, group, ACL (Access Control List)
  - File size
  - Location of file contents
- UNIX File System ➔ **I-node**
- FAT/FAT32 ➔ part of **FAT** (File Alloc. Table)
- NTFS ➔ part of **MFT** (Master File Table)

Partitions

- Disks are broken into one or more partitions.
- Each partition can have its own file system method (Ext, FAT, NTFS, …).



Practical Partition
Linux Vs Windows



A Disk Layout for A File System

- Super block defines a file system
  - size of the file system
  - size of the file descriptor area
  - start of the list of free blocks
  - location of the FCB of the root directory
  - other meta-data such as permission and times
- Where should we put the boot image?

Boot block

- Dual Boot
  - Multiple OS can be installed in one machine.
  - How system knows what/how to boot?

- Boot Loader
  - Understands different OS and file systems.
  - Reside in a particular location in disk.
  - Read Boot Block to find boot image.

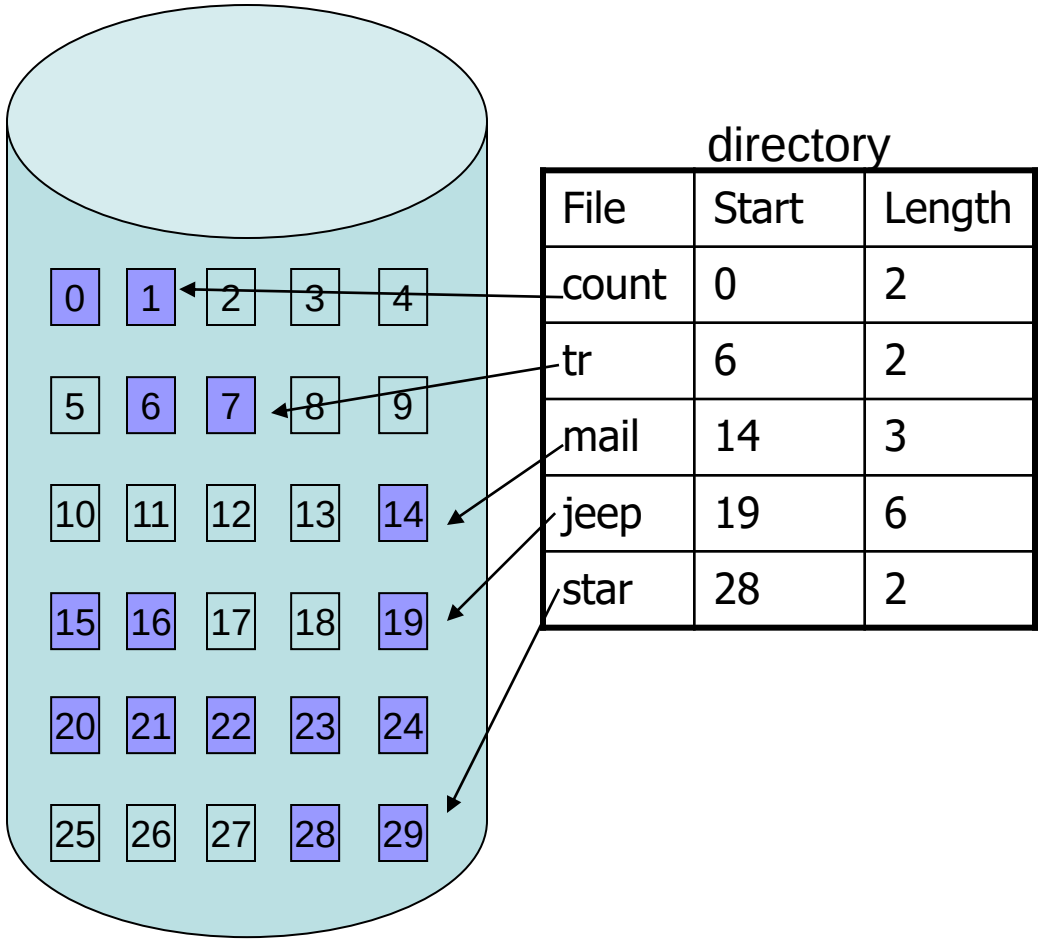
Block Allocation

- Contiguous allocation
- Linked allocation
- Indexed allocation

Disk Allocation Methods

- How should file system allocate disk blocks to each stream-oriented file?
  - Contiguous Allocation
  - Linked Allocation
    - File allocation table
  - Indexed Allocation
    - Linked scheme
    - Multilevel index
    - Combined scheme (Unix)

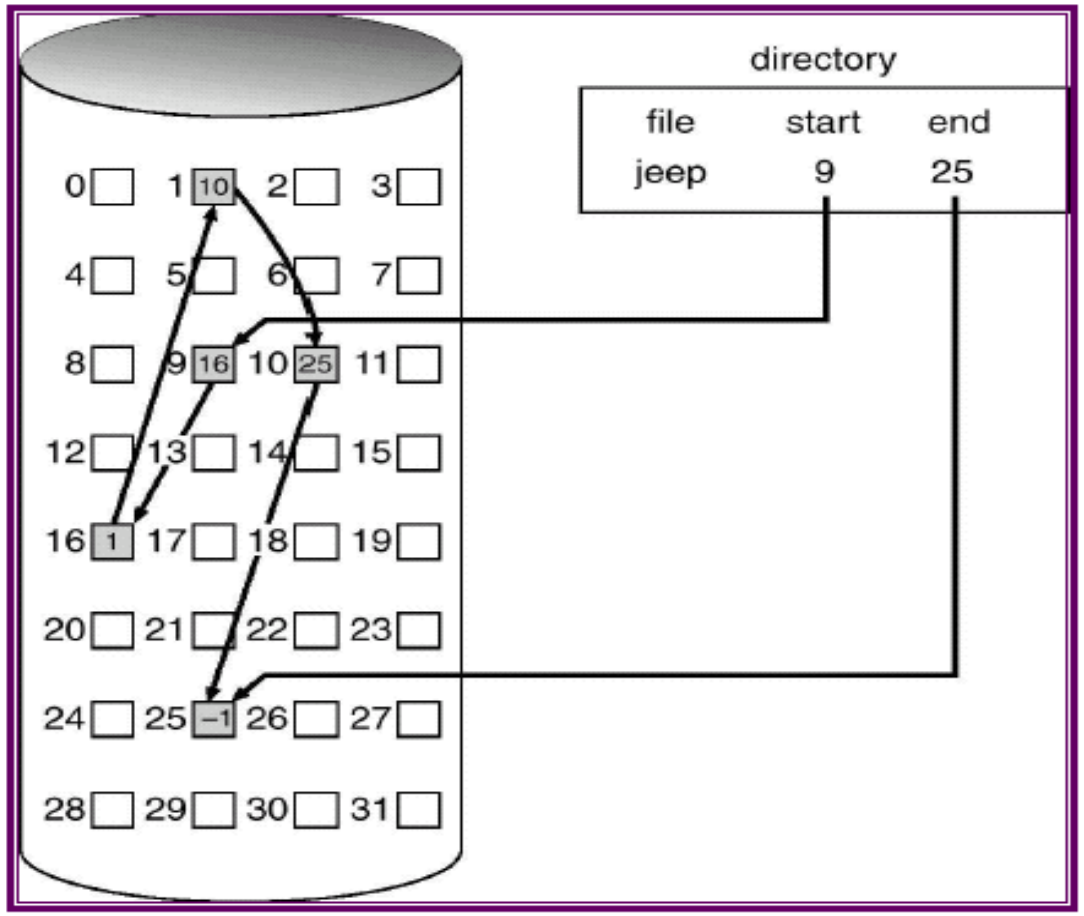
Contiguous Allocation



- Merits
  - Good performance (minimal seek time)
- Example
  - IBM VM/CMS
- Problems
  - External fragmentation
  - Determining the file space upon its creation

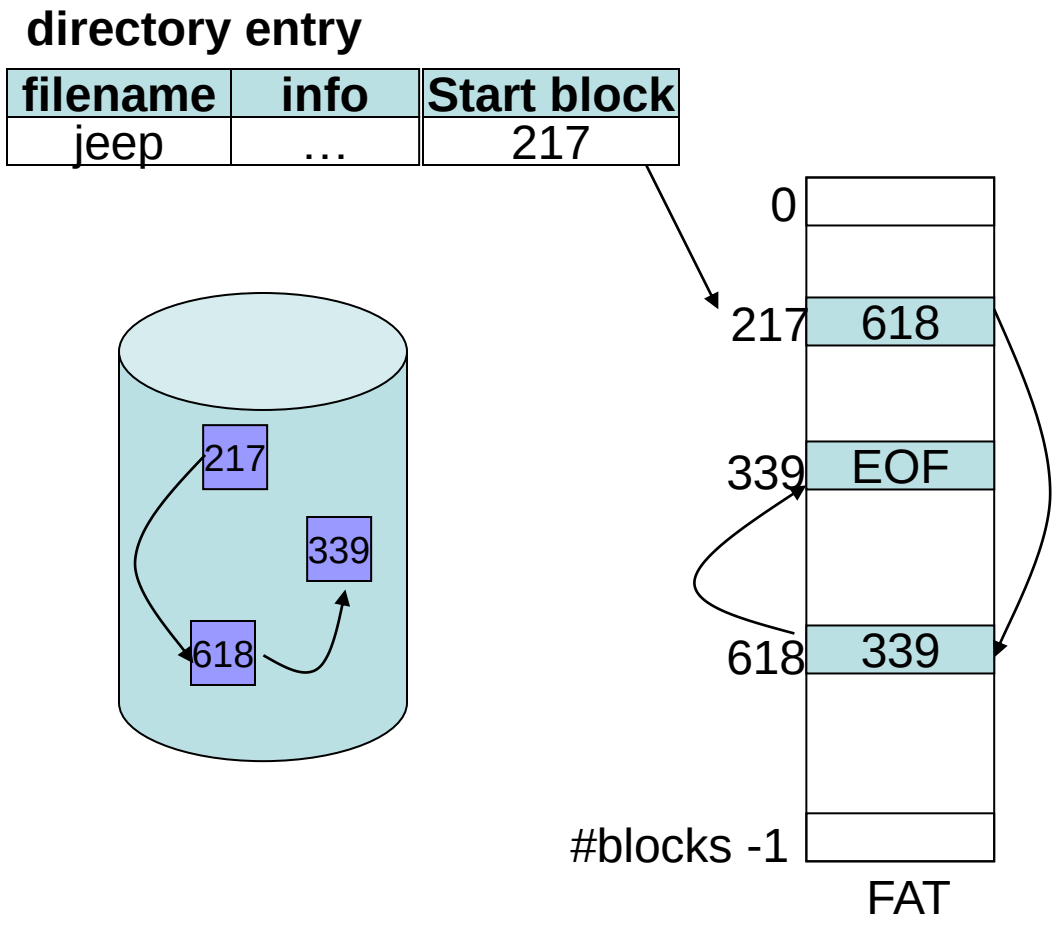
    (Can we predict the size before a file is written?)

Linked Allocation



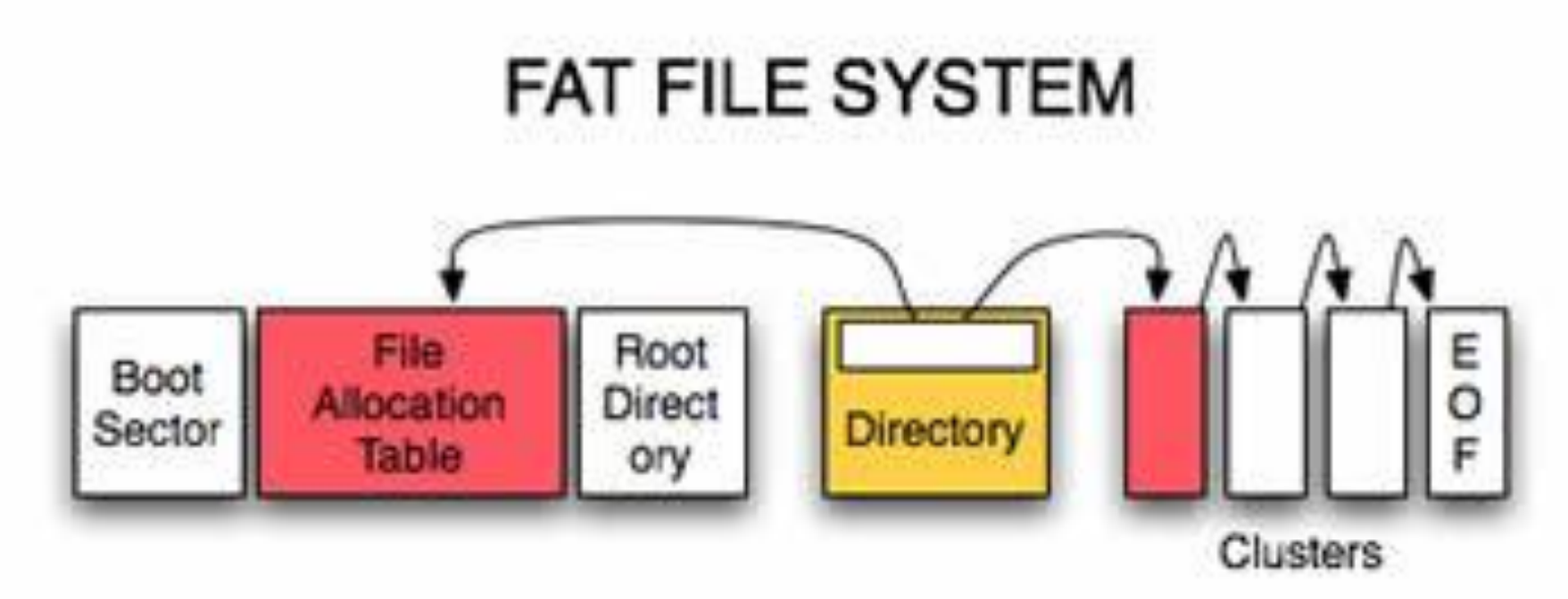
- Merits
  - Need only starting block
  - No external fragmentation
- Problems
  - Sequential access
  - Link information occupying a portion of block
  - File not recovered if its link is broken
  - O(n) time seek operation where n is number of block in the file

File Allocation Table (FAT)



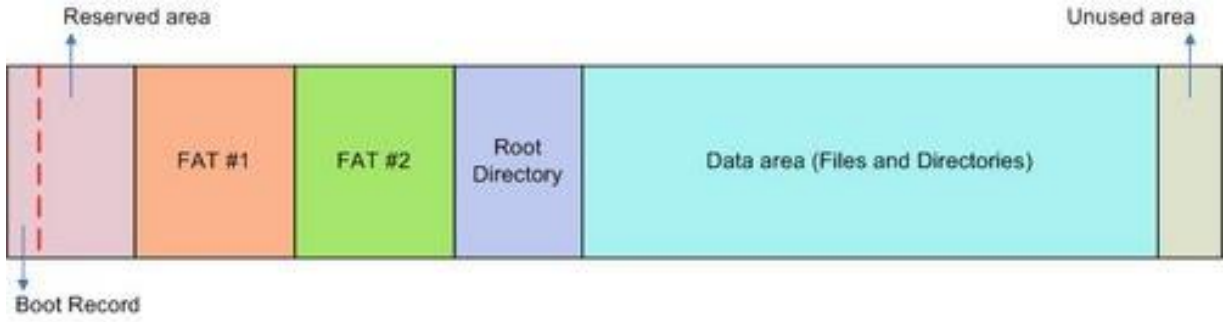
- FAT has an entry for each disk block.
- FAT entries rather than blocks themselves are **linked**.
- Example:
  - MS-DOS and OS/2
- Advantage:
  - Save disk block space
  - Faster random accesses
- Disadvantage:
  - A significant number of disk head seeks

FAT
Info Graphics

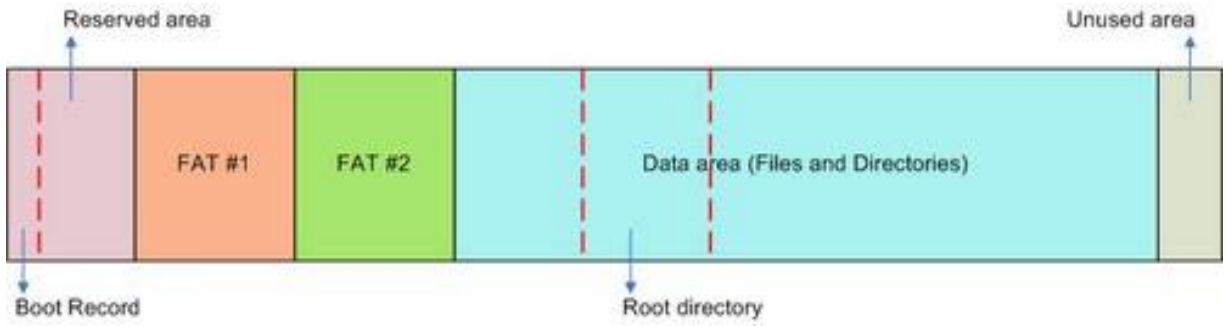


FAT

- FAT == File Allocation Table
- FAT is located at the top of the volume.
  - two copies kept in case one becomes damaged.



The structure of FAT16 file system



The structure of FAT32 file system

by **iprinceps**, <iprinceps@gmail.com>

File Allocation Table (FAT)

- FAT12
  - FAT entry size: 12bits
  - #FAT entries: 4K
  - Disk block (cluster) size: 32K (depends on each system)
  - Total disk size: 128M
- FAT16
  - FAT entry size: 16bits
  - #FAT entries: 64K
  - Disk block size: 32K
  - Total disk size: 2G
- FAT32
  - FAT entry size: 32bits, of which 28bits are used to hold blocks
  - #FAT entries: 256M
  - Disk block size: 32K
  - Total disk size: 8T, (but limited to 2T due to the use of sector counts with the 32bit entry.)

FAT Limitations

- Entry to reference a cluster is 16 bit
  - ➔Thus at most 2^16=65,536 clusters accessible.
  - ➔Partitions are limited in size to 2~4 GB.
  - ➔Too small for today's hard disk capacity!
- For partition over 200 MB, performance degrades rapidly.
  - ➔Wasted space in each cluster increases.
- Two copies of FAT…
  - ➔ still susceptible to a single point of failure!

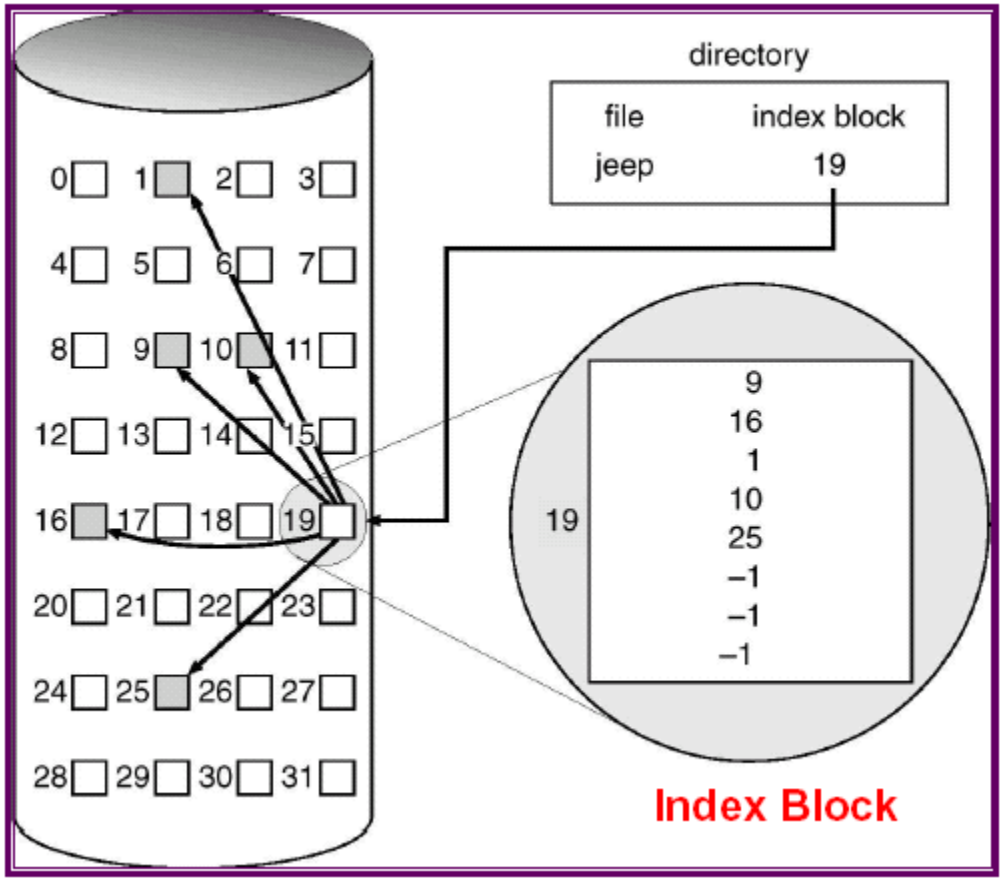
FAT32

Enhancements over FAT

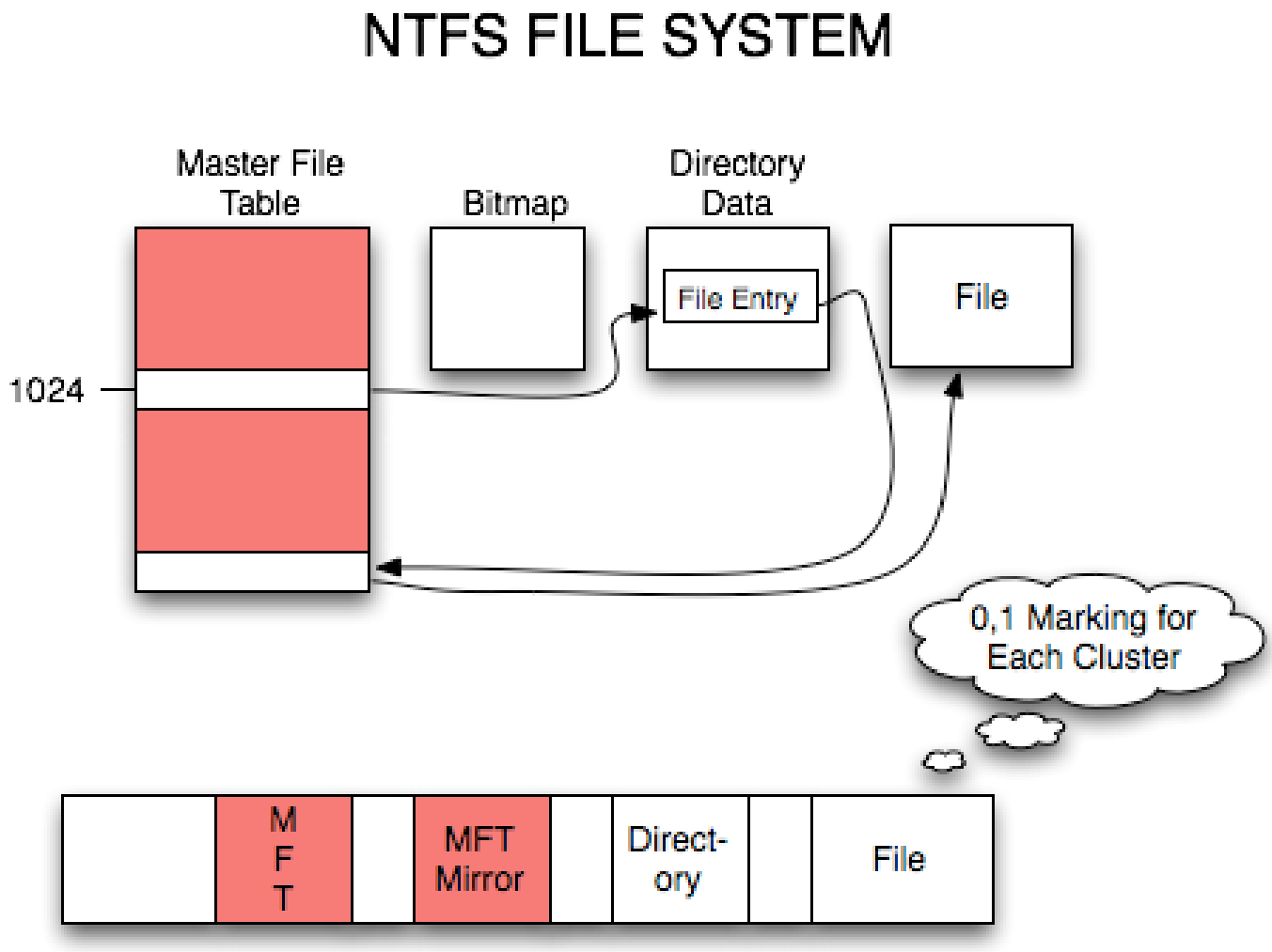
- More efficient space usage
  - By smaller clusters.
  - Why is this possible? 32 bit entry…
- More robust and flexible
  - root folder became an ordinary cluster chain, thus it can be located anywhere on the drive.
  - back up copy of the file allocation table.
  - less susceptible to a single point of failure.

Indexed Block Allocation

- Maintain an array of pointers to blocks.
- Random access becomes as easy as sequential access!
- UNIX File System

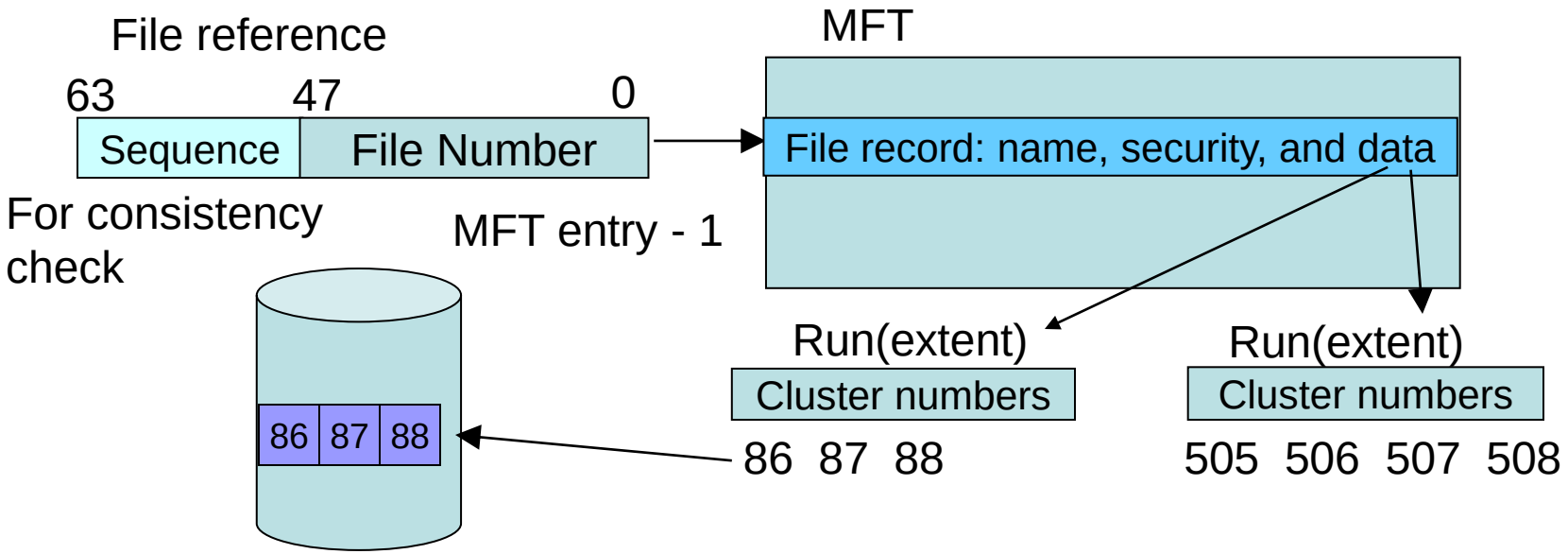


NTFS Info Graphics

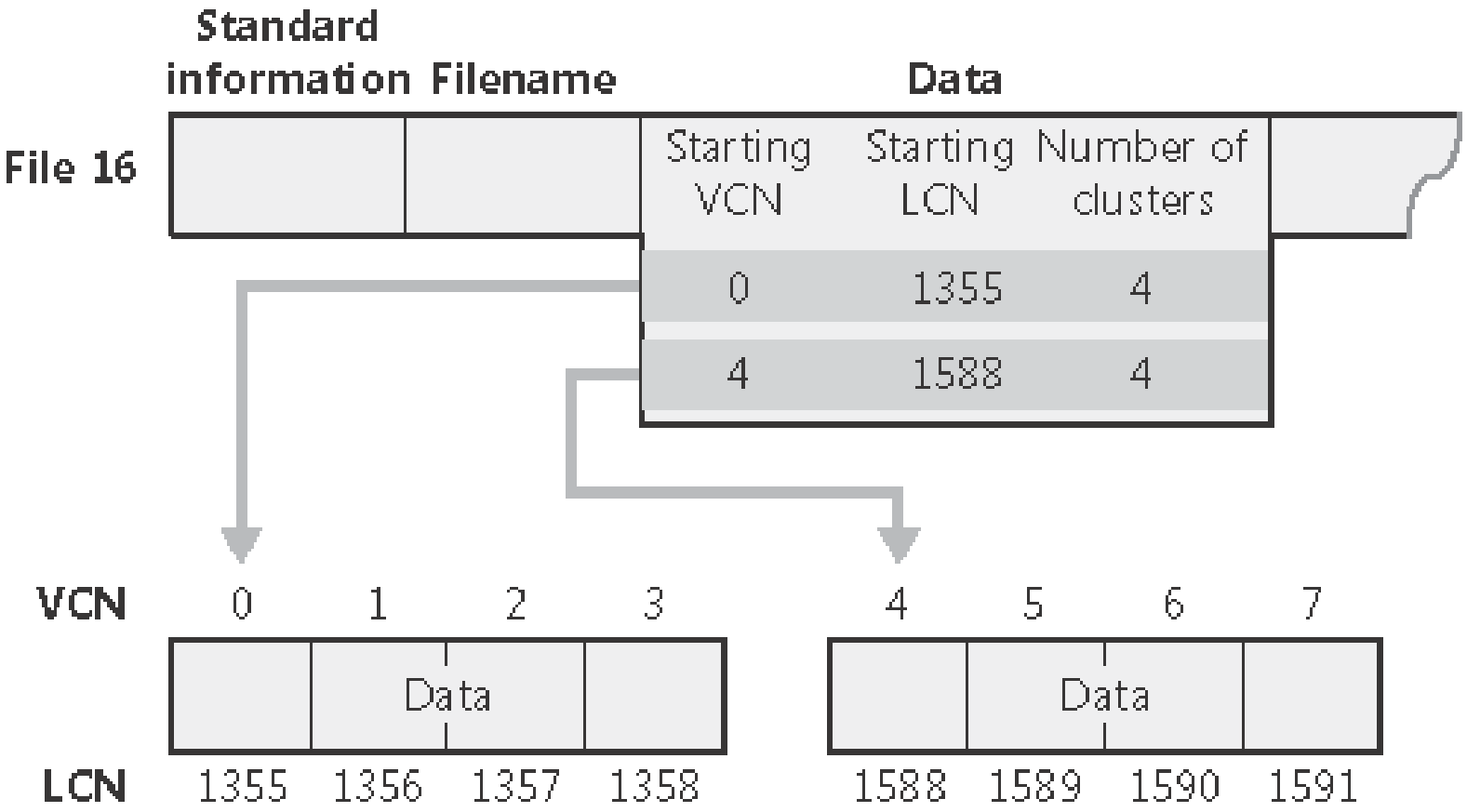


NTFS

- Index Allocation with contiguous scheme
- Uses MFT (Master File Table):
  - An array of records, each holding the attributes for a different file



NTFS


Standard information
Filename
Data
File 16
Starting VCN
Starting LCN
Number of clusters
0
1355
4
4
1588
4
VCN
0
1
2
3
4
5
6
7
Data
Data
LCN
1355
1356
1357
1358
1588
1589
1590
1591

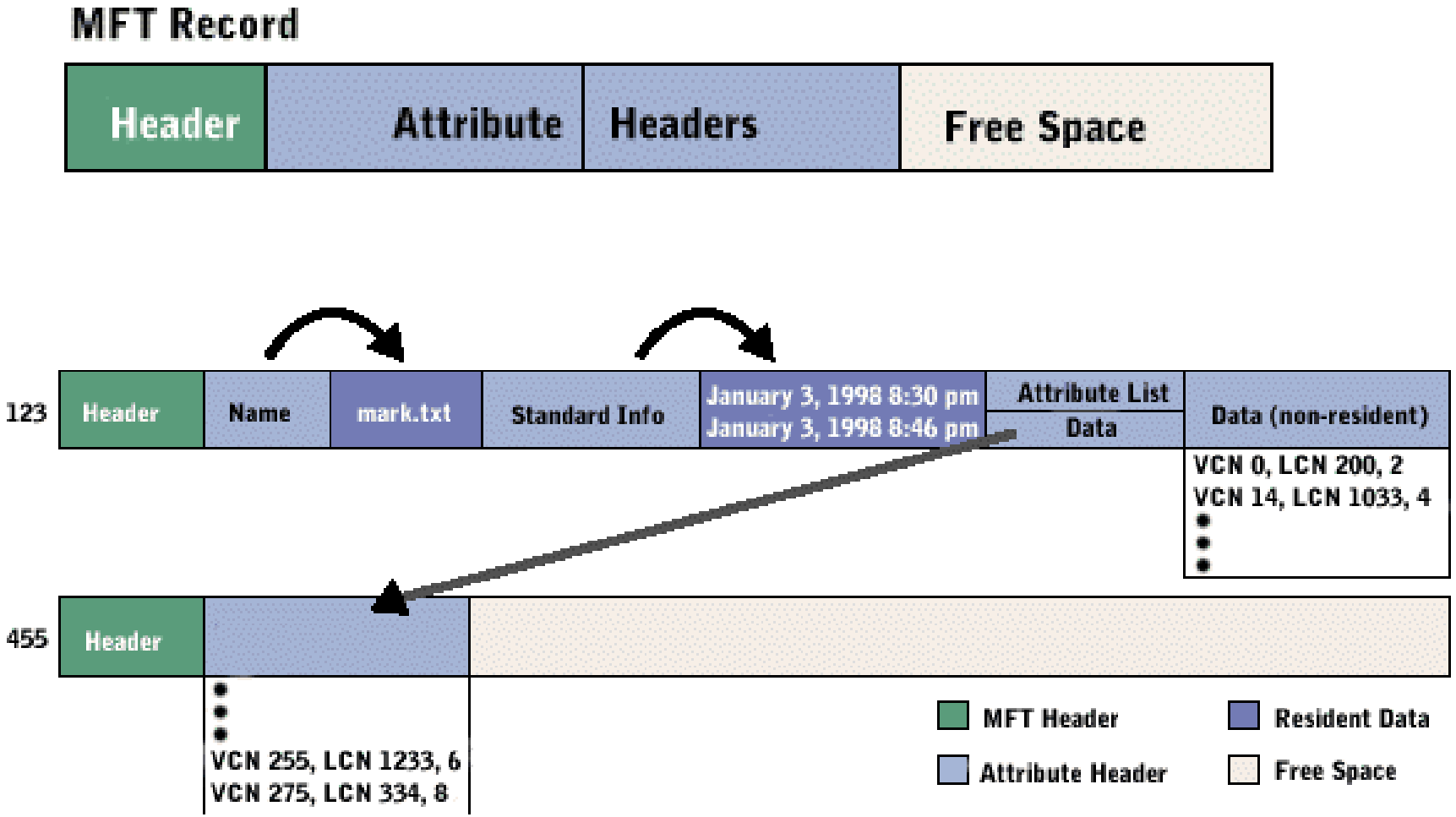
NTFS

- Scalability
  - NTFS references clusters with 64-bit addresses.
  - Thus, even with small sized clusters, NTFS can map disks up to sizes that we won't likely see even in the next few decades.
- Reliability
  - NTFS is a **journaling file system**, which maintains a log of all changes made.
  - Under NTFS, a log of transactions is maintained so that CHKDSK can roll back transactions to the last commit point in order to recover consistency within the file system.
  - Under FAT, CHKDSK checks the consistency of pointers within the directory, allocation, and file tables.

NTFS Metadata Files

NameMFT	Description
$MFT	Master File Table
$MFTMIRR	Copy of the first 16 records of the MFT
$LOGFILE	Transactional logging file
$VOLUME	Volume serial number, creation time, and dirty flag
$ATTRDEF	Attribute definitions
.	Root directory of the disk
$BITMAP	Cluster map (in-use vs. free)
$BOOT	Boot record of the drive
$BADCLUS	Lists bad clusters on the drive
$QUOTA	User quota
$UPCASE	Maps lowercase characters to their uppercase version

NTFS : MFT record



FAT Vs NTFS

FEATURE	FAT32	NTFS
Max. Partition Size	2TB	2TB
Max. File Name	8.3 Characters	255 Characters
Max. File Size	4GB	16TB
File/Folder Encryption	No	Yes
Fault Tolerance	No	Auto Repair
Security	Only Network	Local and Network
Compression	No	Yes
Conversion	Possible	Not Allowed
Compatibility	Win 95/98/2K/2K3/XP	Win NT/2K/XP/Vista/7

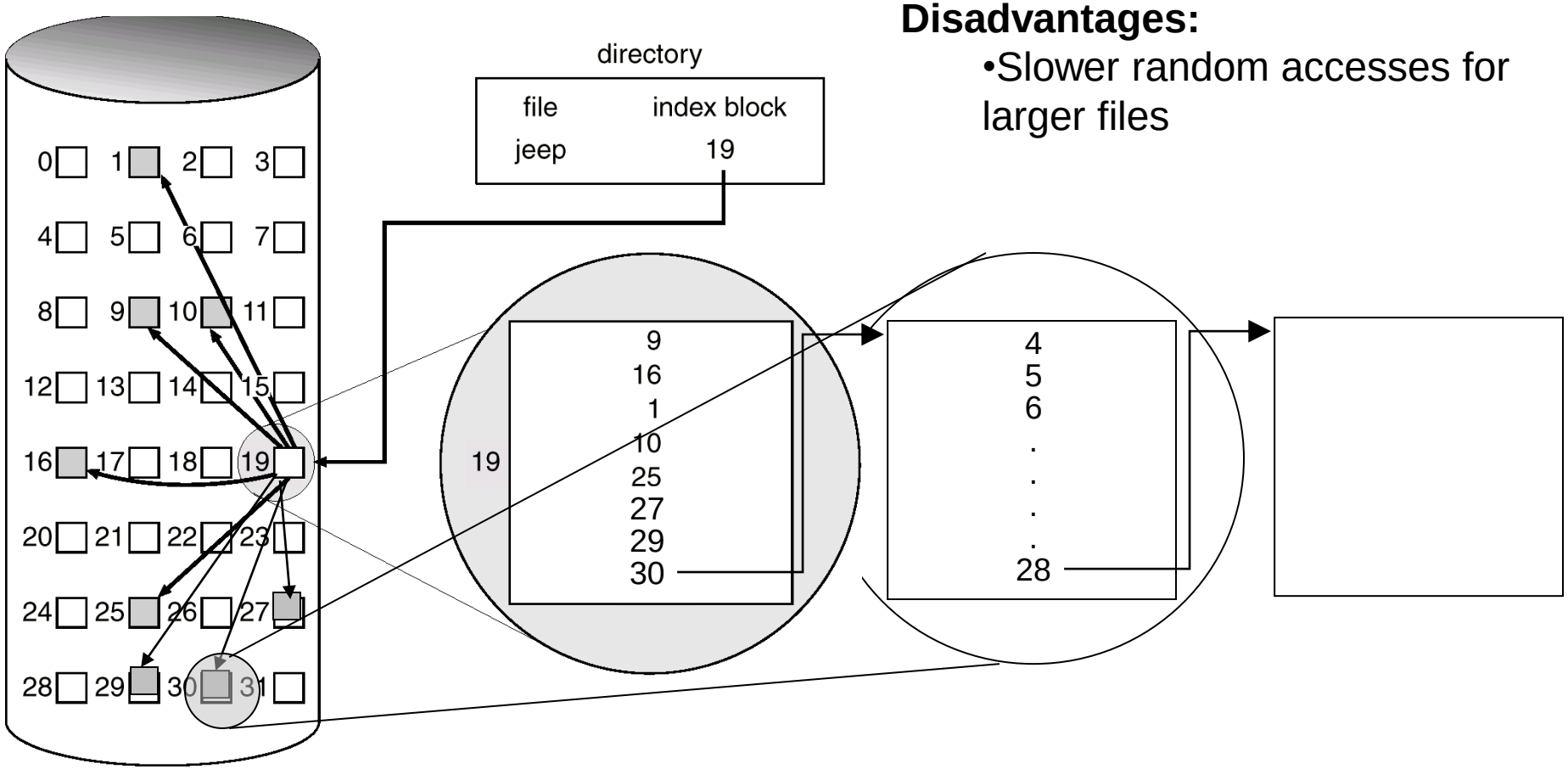
Linked Scheme in Index Allocation

Advantage:

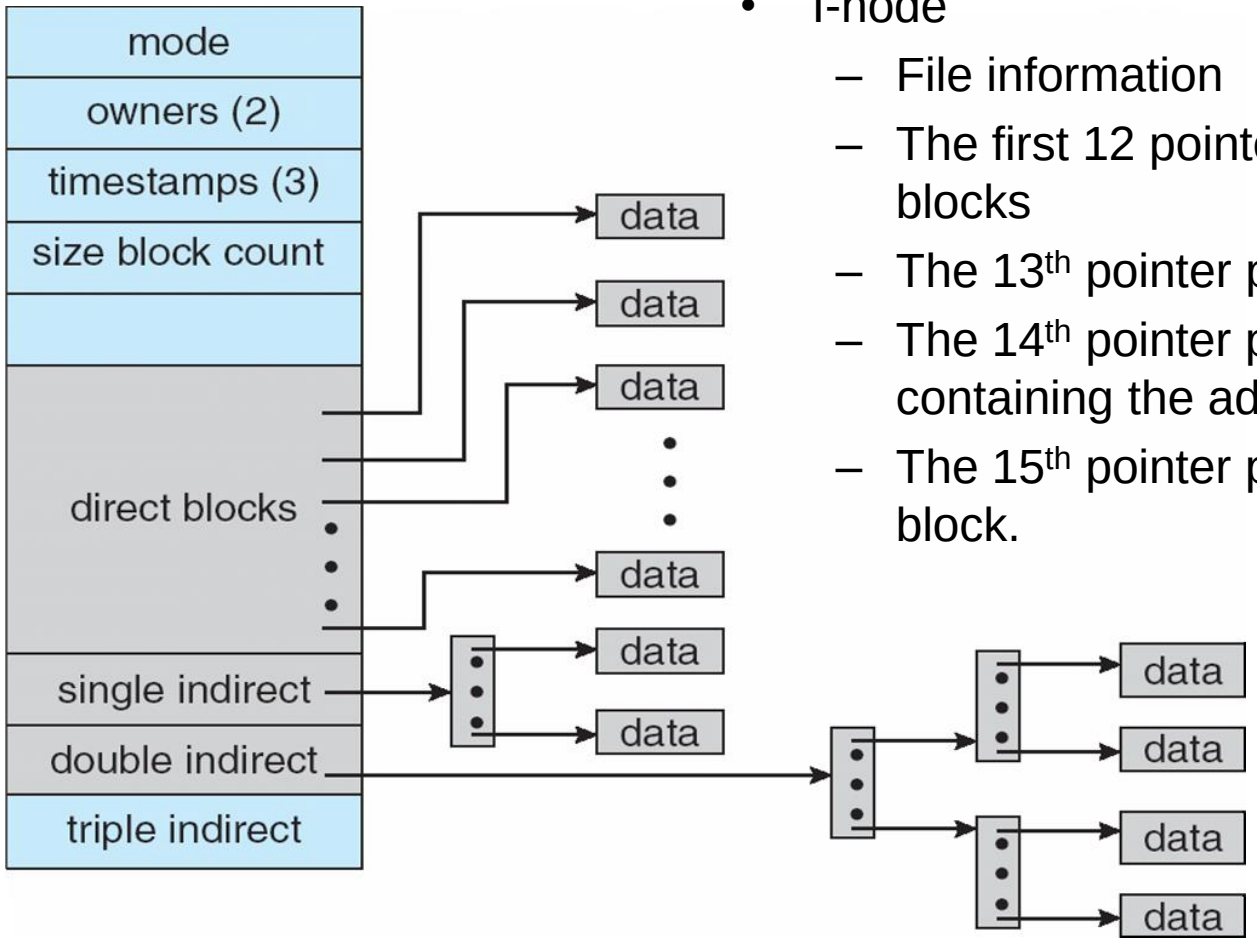
- Adjustable to any size of files

Disadvantages:

- Slower random accesses for larger files



Combined Scheme: UNIX (4K bytes per block)



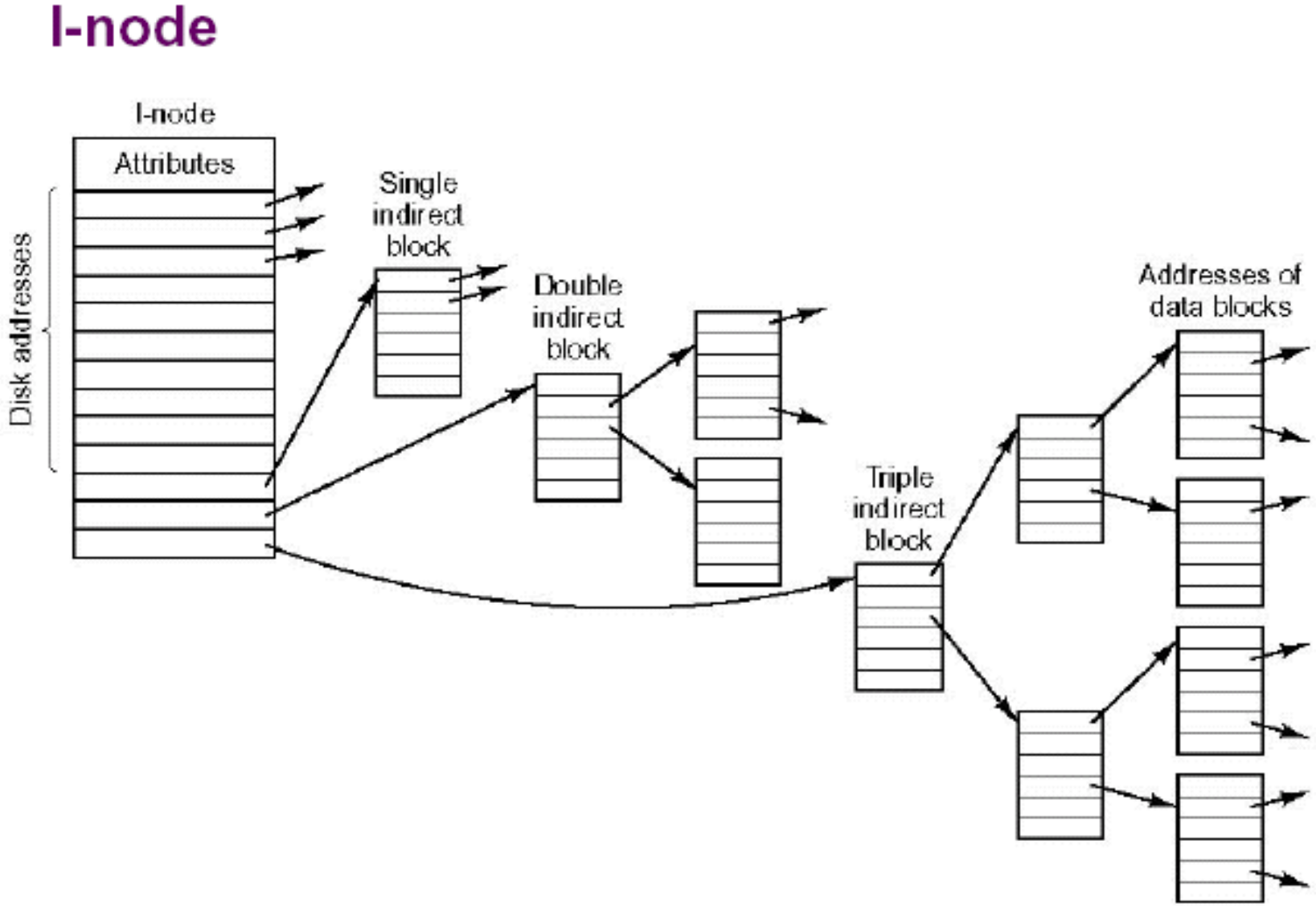
- I-node
  - File information
  - The first 12 pointers point directly to data blocks
  - The 13th pointer points to an index block
  - The 14th pointer points to a block containing the addresses of index blocks
  - The 15th pointer points to a triple index block.

I-node

- FCB(file control block) of UNIX
- Each i-node contains 15 block pointers
  - 12 direct block pointers and 3 indirect (single,double,triple) pointers.
- Block size is 4K

➔ Thus, with 12 direct pointers, first 48K are directly reachable from the i-node.

I-node block indexing



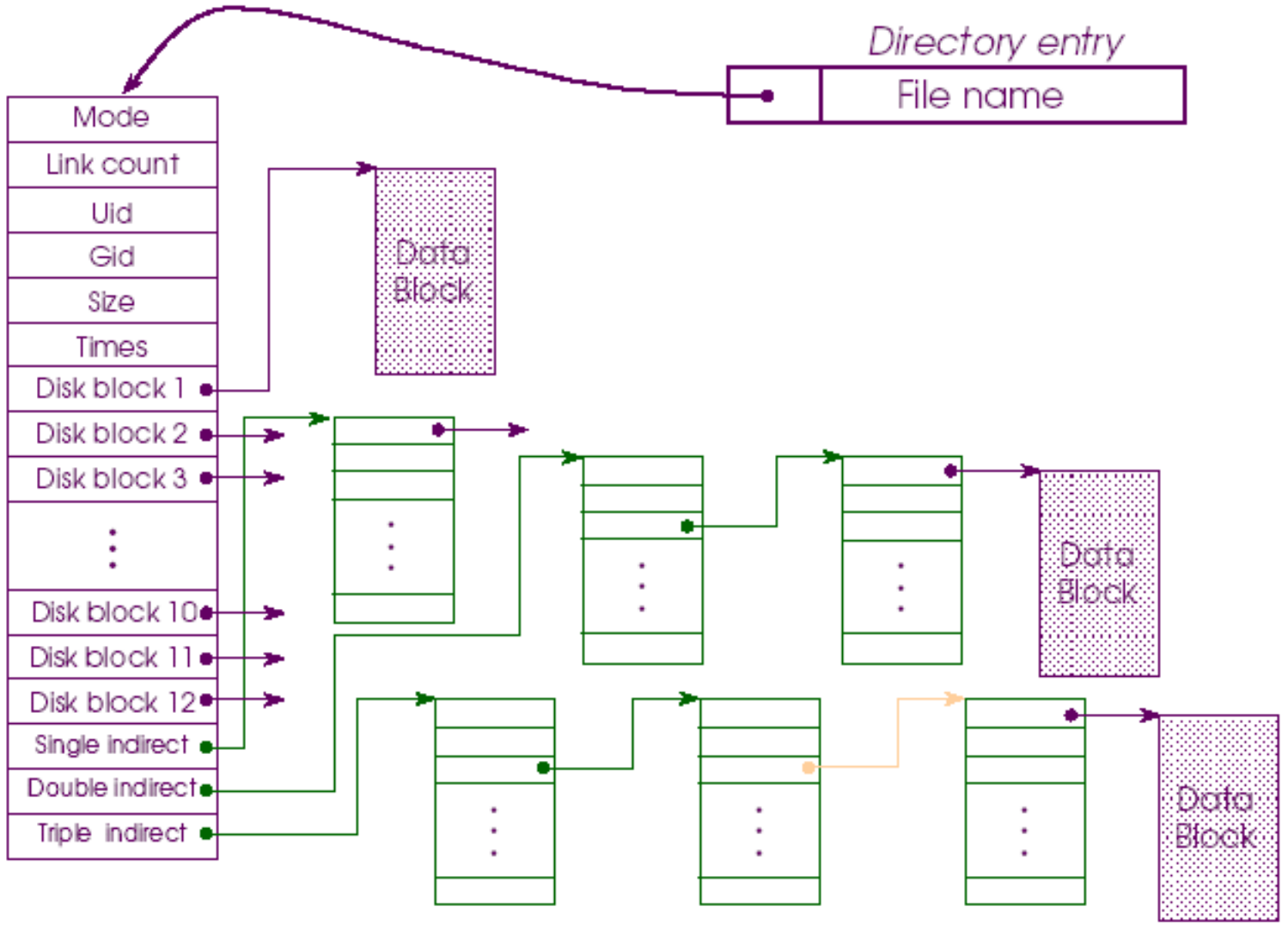
I-node addressing space

Recall block size is 4K, then
Indirect block contains 1024(=4KB/4bytes)entries

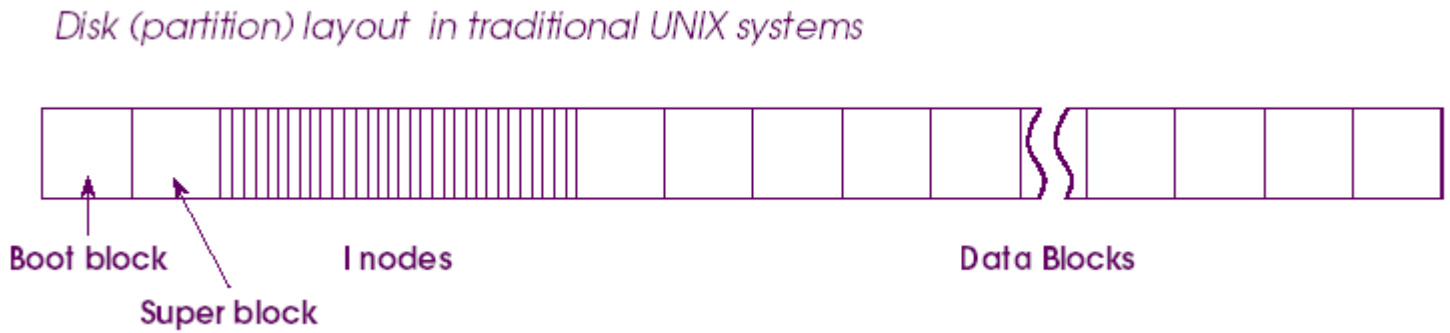
- A single-indirect block can address
  1024 * 4K = **4M** data
- A double-indirect block can address
  1024 * 1024 * 4K = **4G** data
- A triple-indirect block can address
  1024 * 1024 * 1024 * 4K = **4T** data

Any Block can be found with at most 3 indirections.

File Layout in UNIX



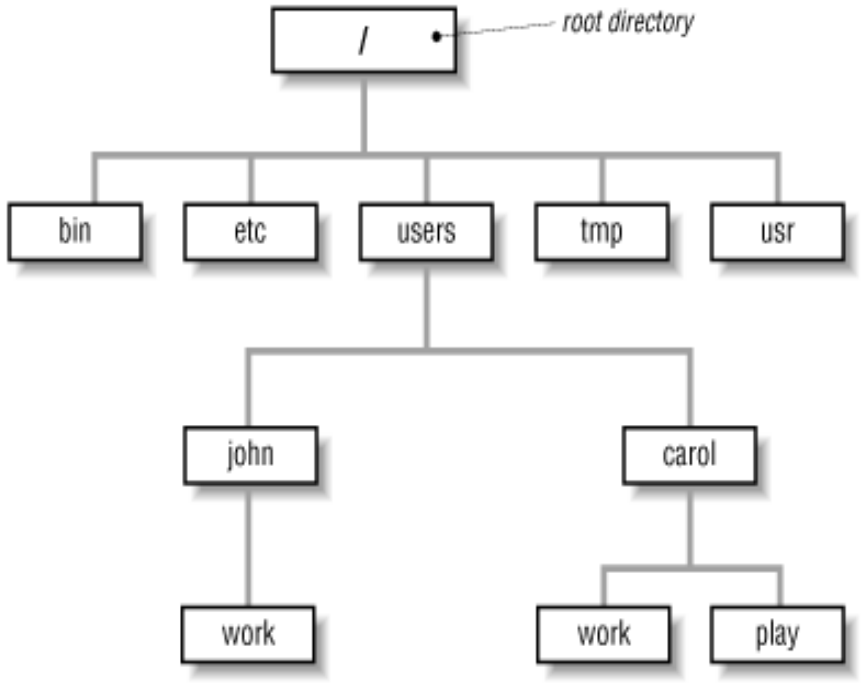
Partition layout in UNIX



- Boot block
- Super block
- FCBs
  - (I-nodes in Unix, FAT or MST in Windows)
- Data blocks

Unix Directory

- Internally, same as a file.
- A file with a type field as a directory.So that only system has certain access permissions.
- <File name, i-node number> tuples.



Unix Directory Example

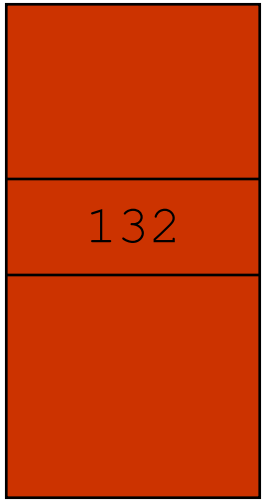
- how to look up data at /usr/bob/mbox

Root Directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up usr gives I-node 6

I-node 6



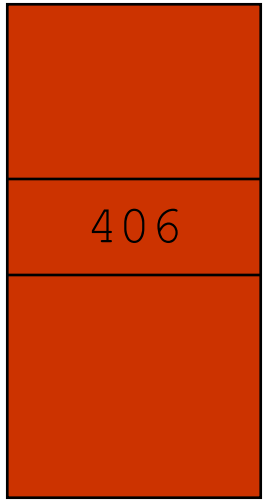
Relevant data (bob) is in block 132

Block 132

6	.
1	..
26	bob
17	jeff
14	sue
51	sam
29	mark

Looking up bob gives I-node 26

I-node 26



Data for /usr/bob is in block 406

Block 406

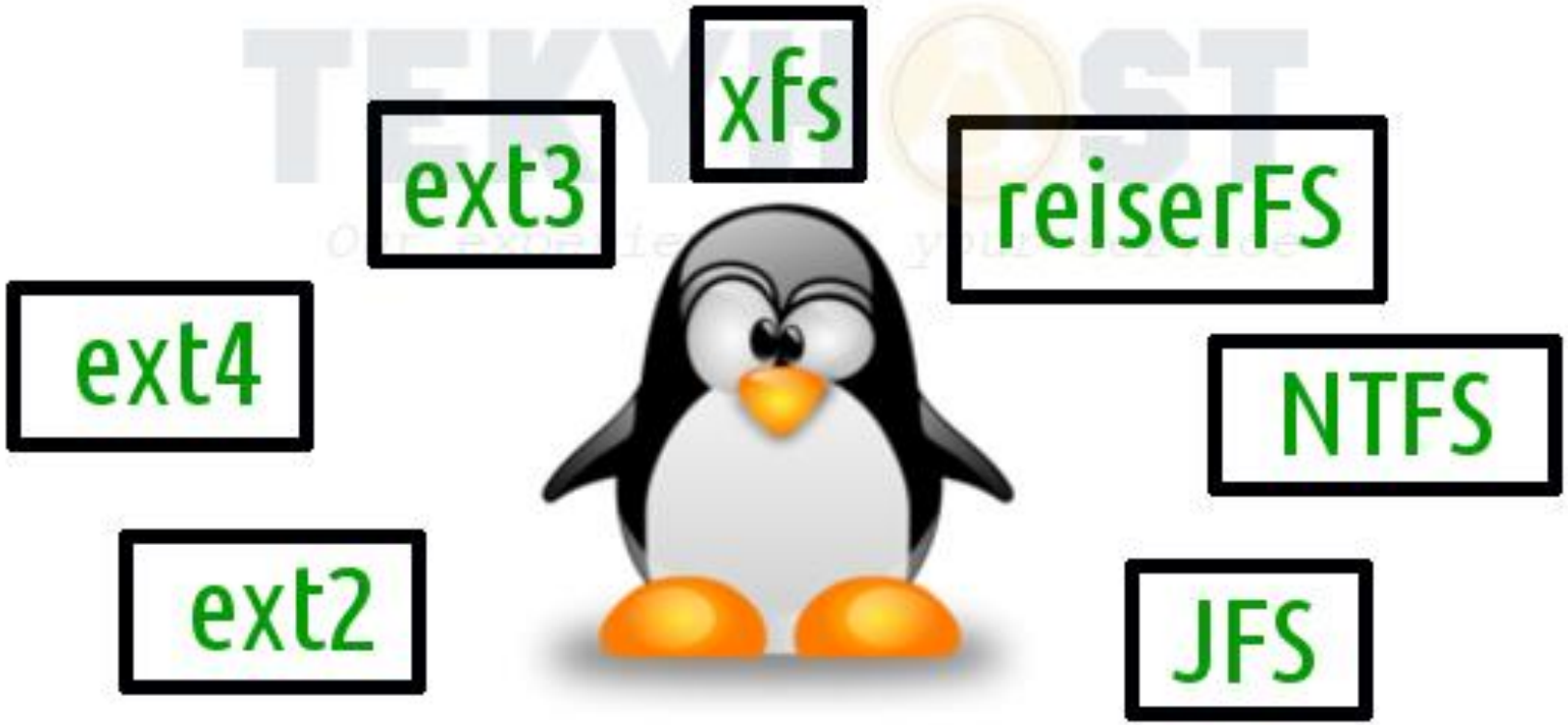
26	.
6	..
12	grants
81	books
60	mbox
17	Linux

Aha! I-node 60 has contents of mbox

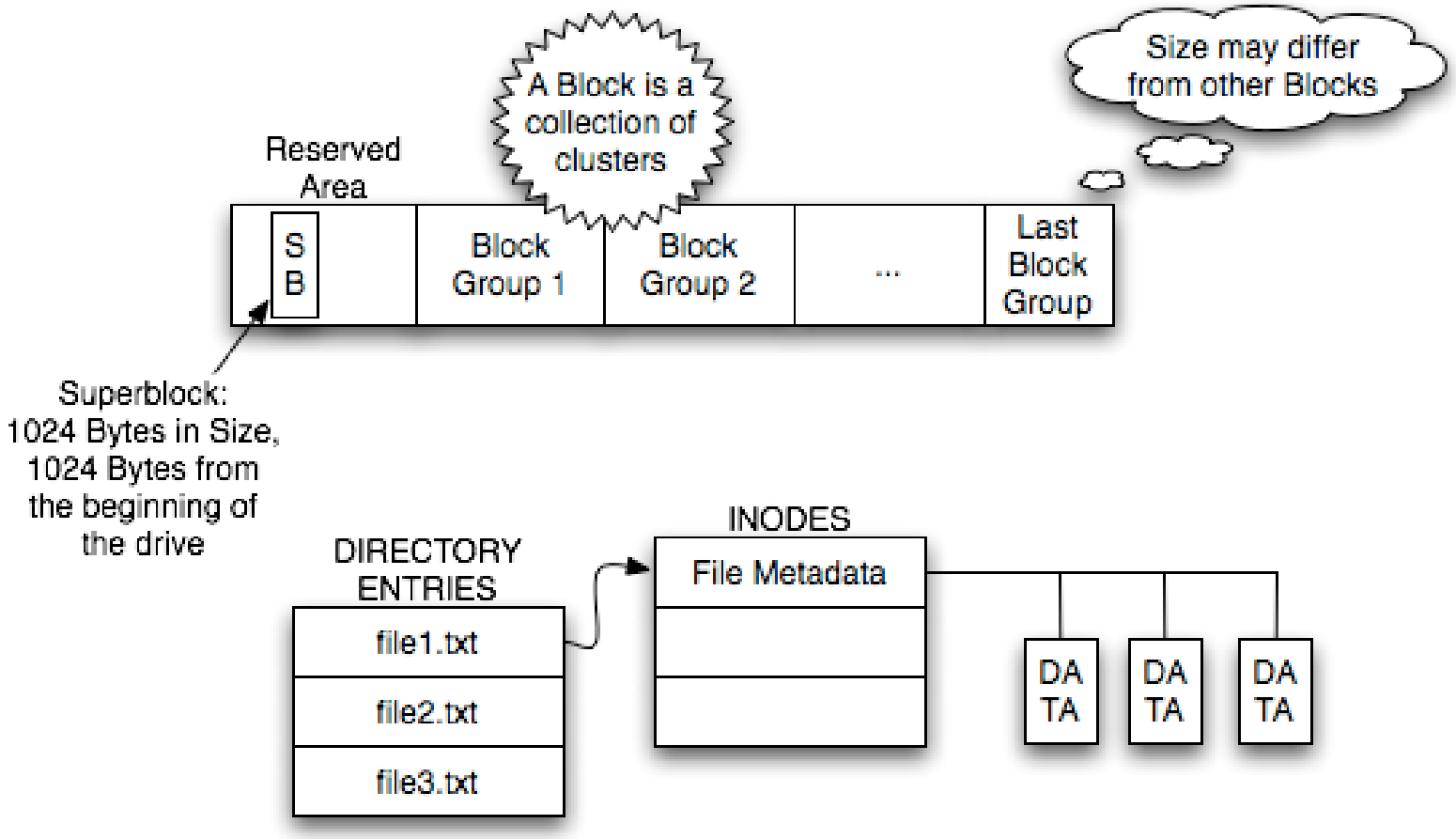
Unix File System Maintenance

- Format
  - Create file system layout: super block, I-nodes…
- Bad blocks
  - Most disks have some, increase over age
  - Keep them in bad-block list
  - “scandisk”
- De-fragmentation
  - Re-arrange blocks rather contiguously
- Scanning
  - After system crashes
  - Correct inconsistent file descriptors

Linux File Systems



EXT2/3 FILE SYSTEM



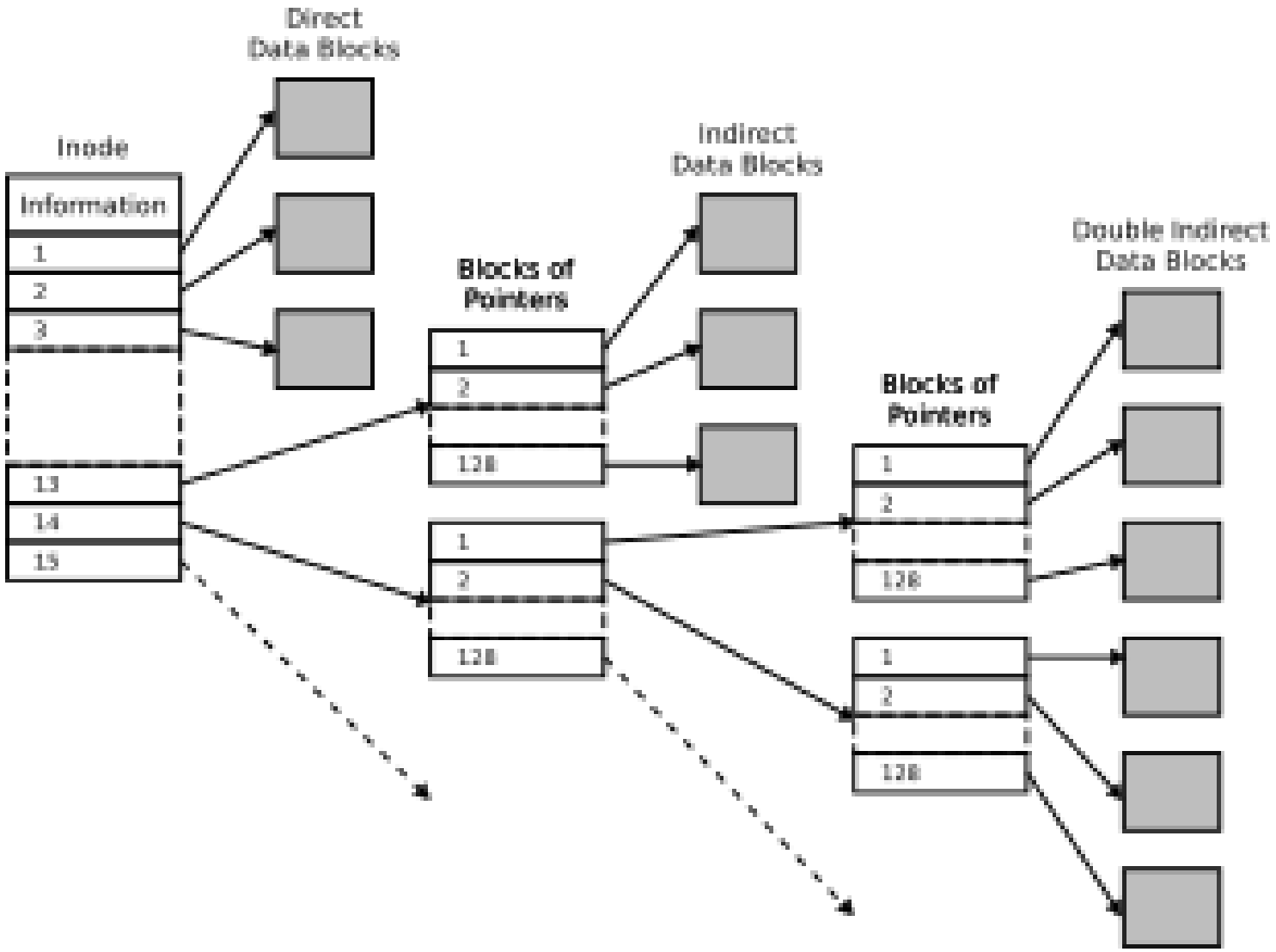
Ext file system
extended file system

- implemented in April 1992 as the first file system created specifically for the Linux kernel.
- inspired by the traditional Unix File System (UFS)
- designed by Rémy Card
- up to 2 gigabytes (GB) in size
- superseded by both ext2 and xfs

Ext2
(**second extended file system)**

- a file system for the Linux kernel
- replacement for the extended file system (ext)
- the default filesystem in several Linux distributions, including Debian and Red Hat Linux, until supplanted more recently[*when?*] by ext3
- ext2 is still the filesystem of choice for flash-based storage media (such as SD cards and USB flash drives)

i-nodes for Ext2



Ext2 limits

Block size	Maximum file size	Maximum volume size
1 KiB	16 GiB	4 TiB
2 KiB	256 GiB	8 TiB
4 KiB	2 TiB	16 TiB
8 KiB	2 TiB	32 TiB

Ext3
(**third extended file system**)

- a journaled file system that is commonly used by the Linux kernel.
- default file system for many popular Linux distributions.
- less speed than competing Linux filesystems, such as ext4, JFS, ReiserFS, and XFS.
- Benchmarks suggest that ext3 also uses less CPU power than ReiserFS and XFS.
- ext3 adds the following features to ext2:
  - A journal
  - Online file system growth
  - HTree indexing for larger directorie

Ext3 limits

Block size	Maximum file size	Maximum volume size
1 KiB	16 GiB	4 TiB
2 KiB	256 GiB	8 TiB
4 KiB	2 TiB	16 TiB
8 KiB	2 TiB	32 TiB

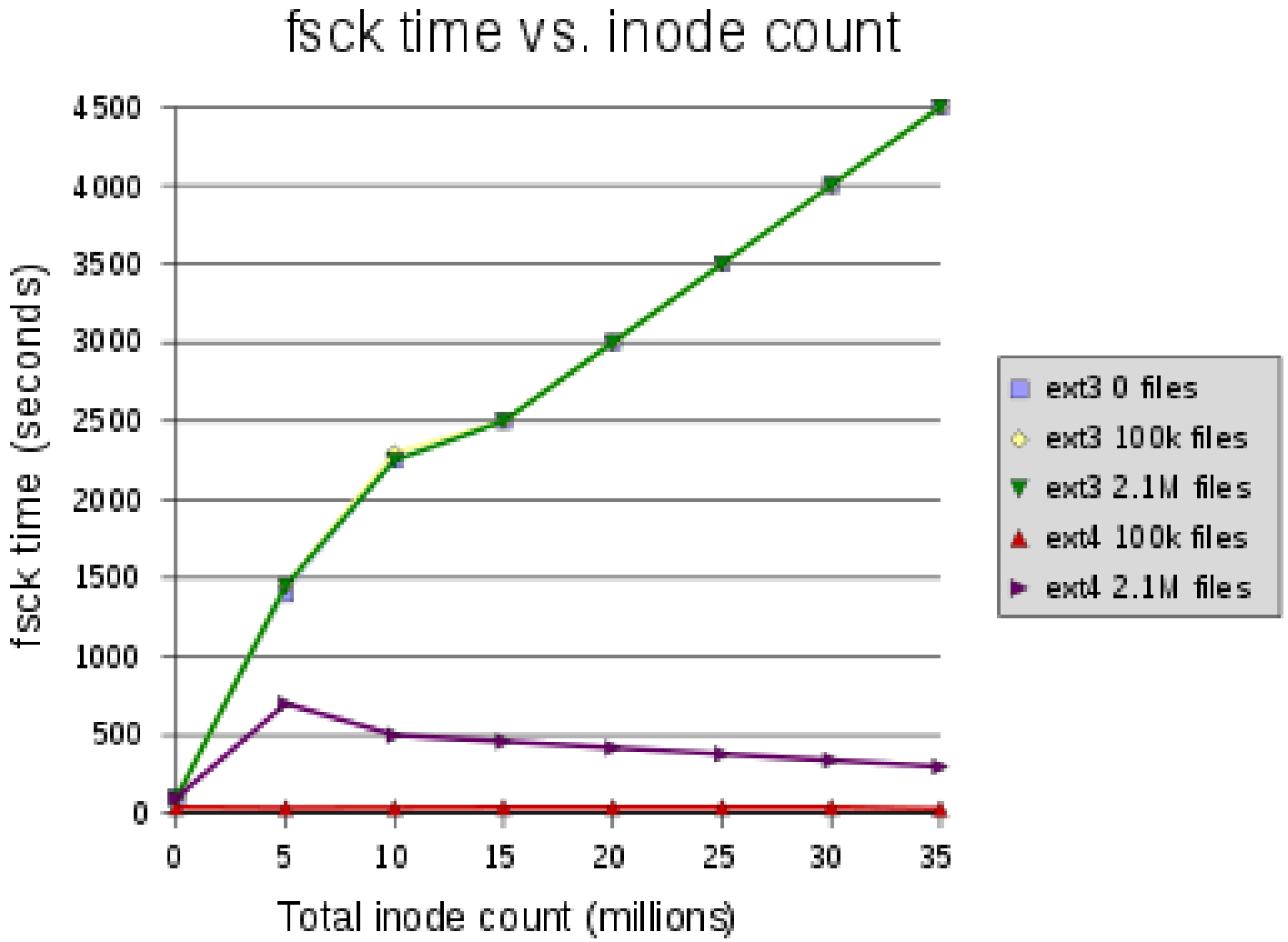
Ext4
(**forth extended file system)**

- a journaled file system that is commonly used by the Linux kernel.
- default file system for many popular Linux distributions.
- ext4 was included in version 2.6.19 of the Linux kernel
- backward-compatible extensions to ext3 and ext2
- Based file system for Google storage and Android OS

Ext2 Vs Ext3 Vs Ext4

	Ext2	Ext3	Ext4
Introduced	in 1993	in 2001 (2.4.15)	in 2006 (2.6.19) in 2008 (2.6.28)
Max file size	16GB ~ 2TB	16GB ~ 2TB	16GB ~ 16TB
Max file system size	2TB ~ 32TB	2TB ~ 32TB	1EB
Feature	no Journaling	Journaling	Extents Multiblock allocation Delayed allocation

Ext3 Vs Ext4


fsck time vs. inode count
fsck time (seconds)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
35
Total inode count (millions)
ext3 0 files
ext3 100k files
ext3 2.1M files
ext4 100k files
ext4 2.1M files

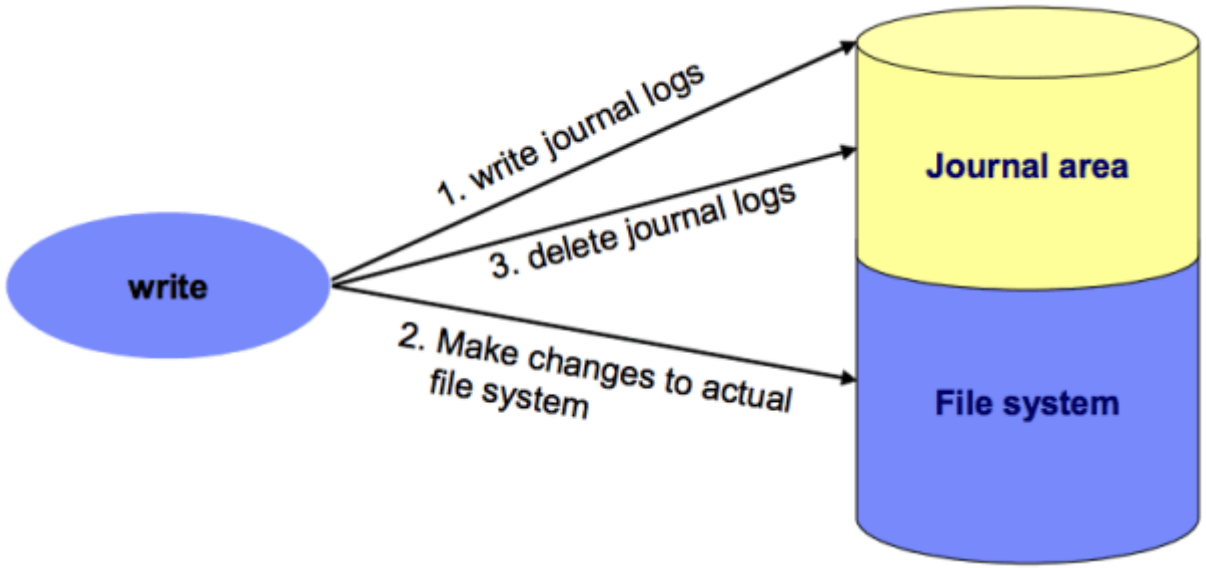
XFS

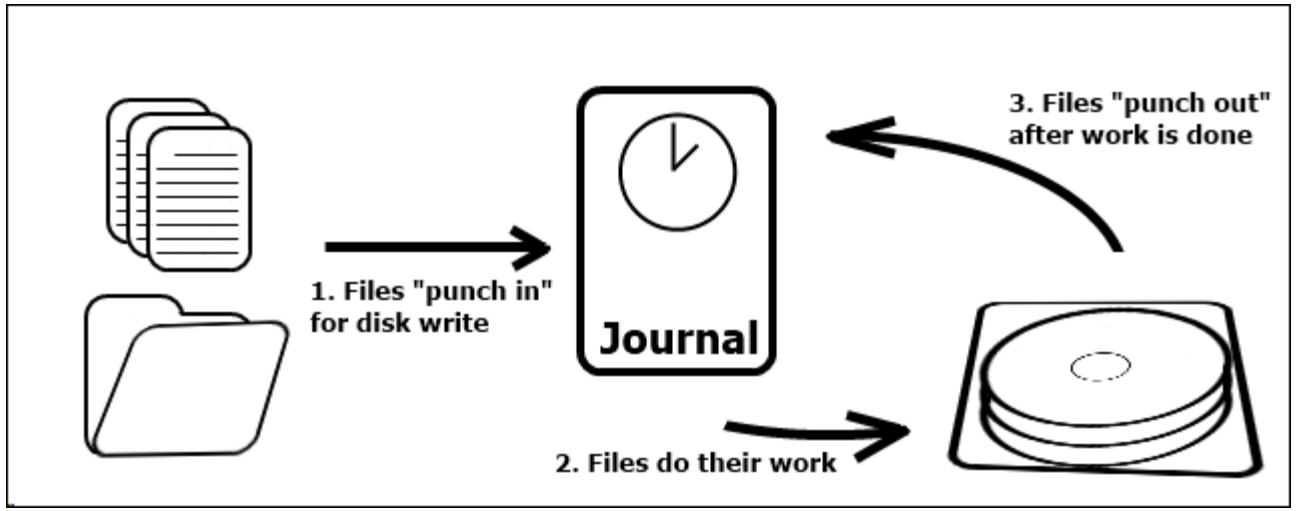
- a high-performance 64-bit journaling file system created by Silicon Graphics, Inc (SGI) in 1993
- XFS was ported to the Linux kernel in 2001; as of June 2014, XFS is supported by most Linux distributions
- execution of parallel input/output (I/O) operations due to its design
- enables extreme scalability of I/O threads, file system bandwidth, and size of files and of the file system
- 8EB for max file size and 8EB for max volume size

file systems supported by Linux / Windows

File System	Windows	Linux
FAT16	X	
FAT32	X	
NTFS	X	
EXT		X
EXT2	X	X
EXT3	X	X
EXT4	X	X
XFS		X

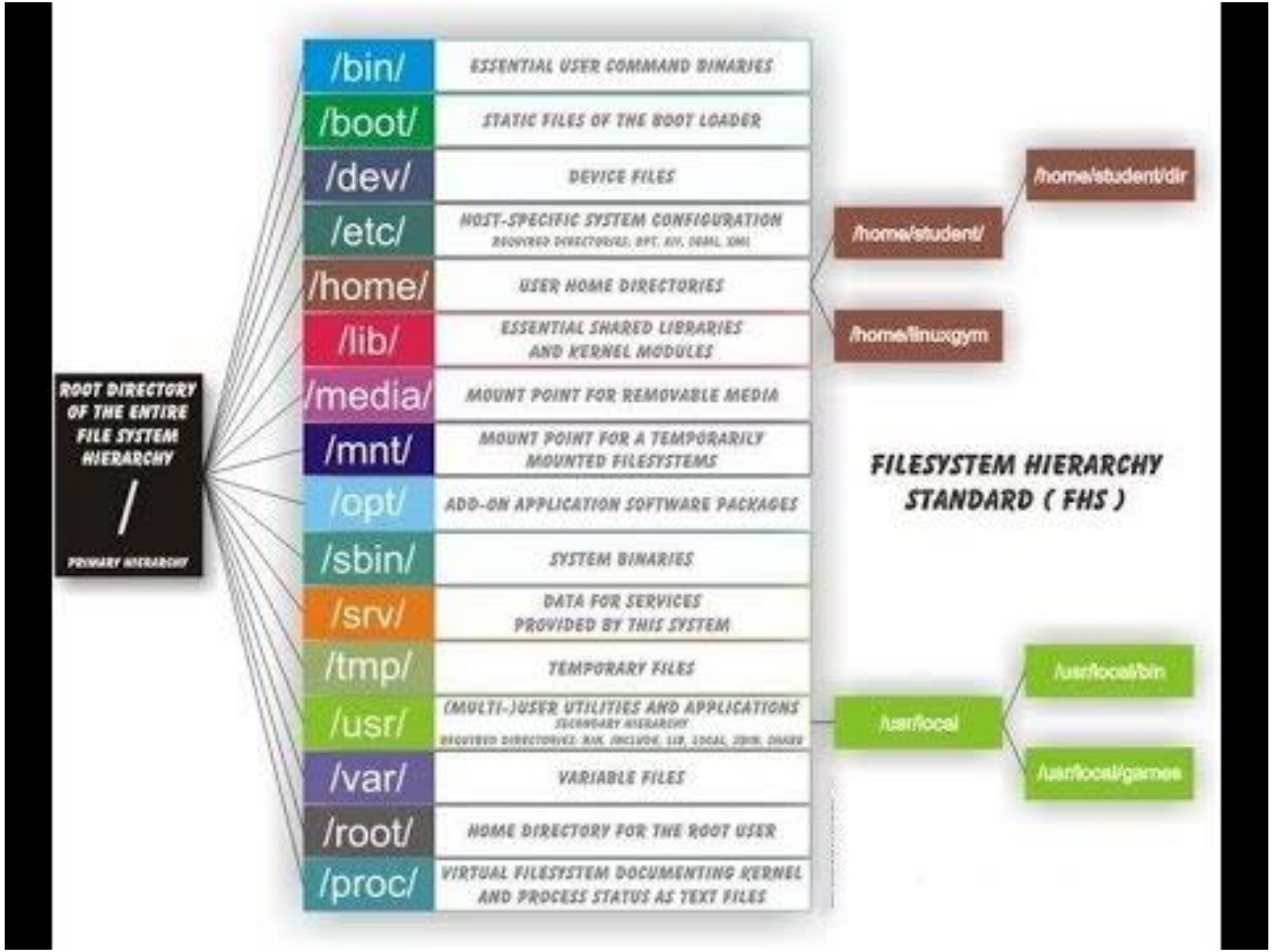
Journal File System





VFS
virtual file system

- An abstract layer on top of a more concrete file system.
- To allow client applications to access different types of concrete file systems in a uniform way.
- Used to access local and network storage devices transparently without the client application noticing the difference.
- Used to bridge the differences in Windows, classic Mac OS/macOS and Unix filesystems
- Applications can access files on local file systems of those types without having to know what type of file system they are accessing.
- First virtual file system mechanisms on Unix-like systems was introduced by Sun Microsystems in SunOS 2.0 in 1985.



https://www.youtube.com/watch?v=6PzwUhLQhP8