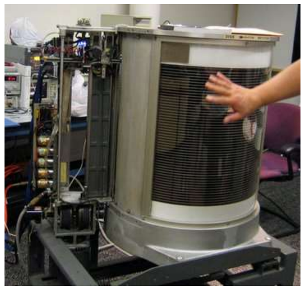
File Systems

Sirak Kaewjamnong

517-312 Operating Systems

History of storage technology


1980: IBM 3380, first GB disk (1.26G), > $100K

1980: ST-560, first 5.25 inch drive, 5MB, $1500

Tape (DECtape): primary storage for main-frames and mini-computers (1950 ~ 70s)

NextCom notebook 2006: first laptop with SSD storage

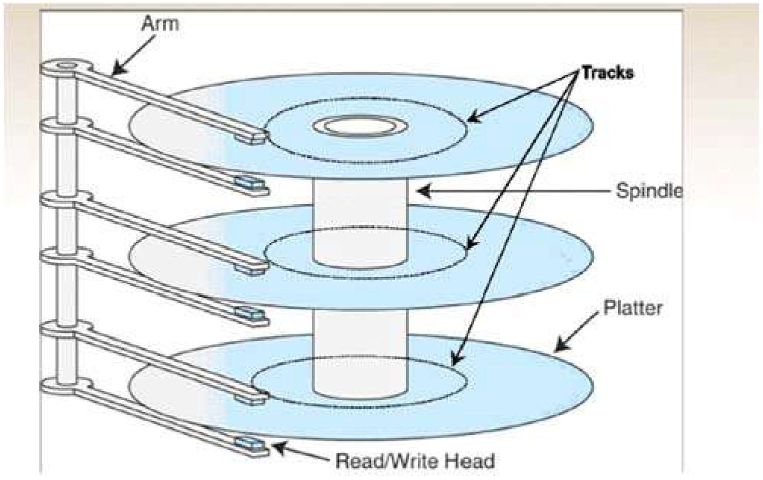
History of storage technology

Disks and the OS

- Disks are messy physical devices
  - Errors, bad blocks, missed seeks, etc.
- The job of the OS is to hide this mess from higher level software
  - Low-level device control (initiate a disk read, etc.)
  - Higher-level abstractions (files, databases, etc.)

How hard disk work?

- Disk components
  - Platters
  - Surfaces
  - Tracks
  - Cylinders
  - Sectors
  - Arm
  - Heads

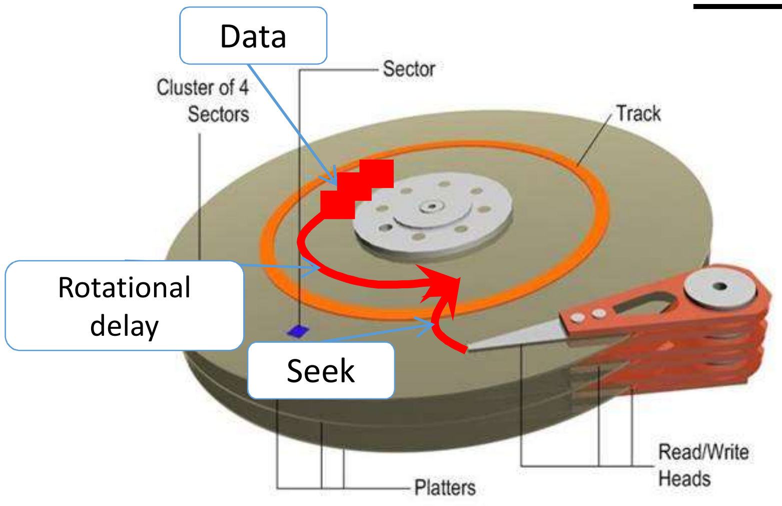


Disk Interaction

- Specifying disk requests requires a lot of info:
  - Cylinder #, surface #, sector #, transfer size…
- Older disks required the OS to specify all of this
  - The OS needed to know all disk parameters
- Modern disks are more complicated
  - Not all sectors are the same size, sectors are remapped, etc.
- Current disks provide a higher-level interface (SCSI)
  - The disk exports its data as a logical array of blocks [0…N]
    - Disk maps logical blocks to cylinder/surface/track/sector
  - Only need to specify the logical block # to read/write
  - But now the disk parameters are hidden from the OS

Disk Performance

- Random disk access is ***SLOW***!



Access data sequentially: only suffer one seek and rotational delay

Random disk access: suffers one seek and rotational delay every time!

Disk Performance

- Disk request performance depends upon three steps
  - Seek – moving the disk arm to the correct cylinder
    - Depends on how fast disk arm can move (increasing very slowly)
  - Rotation – waiting for the sector to rotate under the head
    - Depends on rotation rate of disk (increasing, but slowly)
  - Transfer – transferring data from surface into disk controller electronics, sending it back to the host
    - Depends on density (increasing quickly)
- When the OS uses the disk, it tries to minimize the cost of all of these steps
  - Particularly seeks and rotation

Solid State Drive (SSD)

- NAND-based flash memory, a non-volatile type of memory
- An SSD does not have a mechanical arm to read and write data
- the data storage integrity will be maintained for well over 200 years
- The controller is a very important factor in determining the speed of the SSD

SSD Vs HDD Comparison

SSD

- Less power draw, averages 2 – 3 watts
- Expensive, 4,000 Baht per TB
- Around 10-13 seconds average OS bootup time
- There are no moving parts and as such no sound, no vibration
- Mean time between failure rate of 2.0 million hours
- An SSD is safe from any effects of magnetism
- Up to 30% faster than HDD

HDD

- More power draw, averages 6 – 7 watts
- Cheap, 1500 Baht per TB
- Around 30-40 seconds average OS bootup time
- Audible clicks and spinning can be heard and result in vibration
- Mean time between failure rate of 1.5 million hours
- Magnets can erase data
- Slower than SSD

Type of SSD in the market

mSATA
6Gb/sec

M.2
32 Gb/sec

PCI-Express SSDs

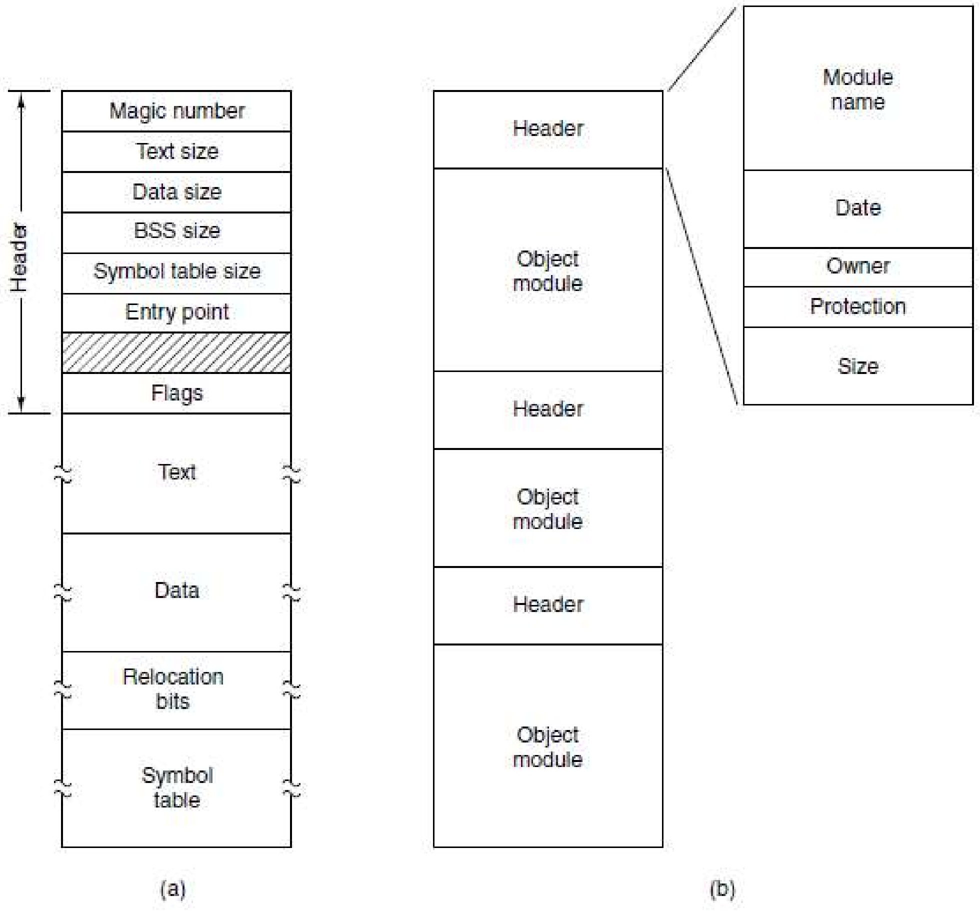
Files

- A file is an abstraction mechanism to provide a way to store information on the disk and read it back later
- Information stored in files must be persistent
- Processes can read files and create new ones
- **File system** is the part of the OS dealing with files
  - Implement an abstraction (files) for secondary storage
  - Organize files logically (directories)
  - Permit sharing of data between processes, people, and machines
  - Protect data from unwanted access (security)

File naming

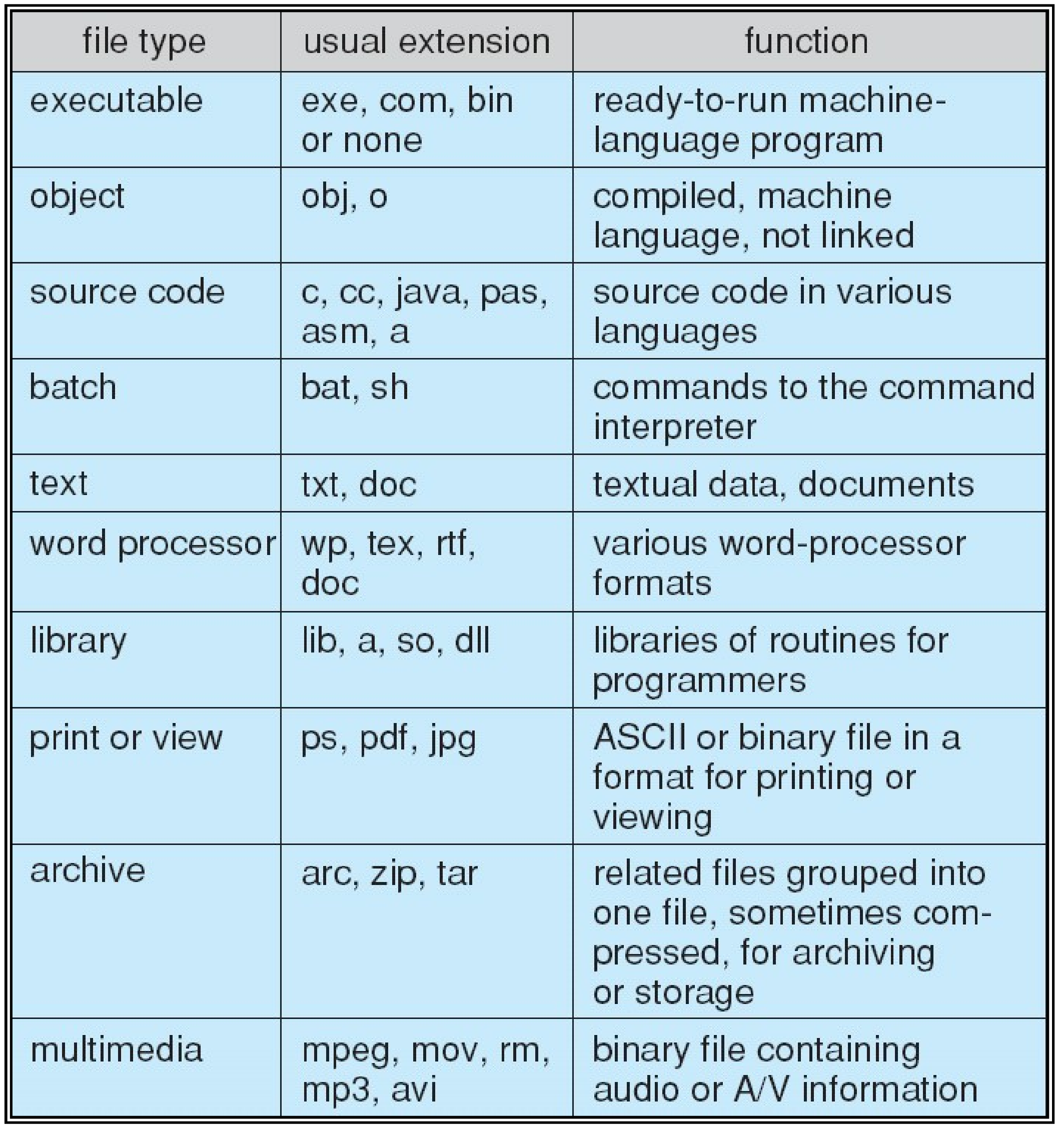
- Rules for file naming vary from system to system, but all current OSs allow strings
- Some file systems distinguish between upper an lowercases letters (Unix) while some are not (Windows)
- Many OSs support two-part file names, with the two parts separated by a period
  - The following part is called the file extension and usually indicates something about the file, .com, .exe, .bat, .dll, .jpg, etc.
  - In Unix, file extension are just convention and are not enforced by the OS, but encodes type in contents
- The OS will execute a file only if it has the proper format

Simple UNIX file



(a) An executable file
(b) An archive

Common File Types



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes com-pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Basic File Operations

Unix

- creat(name)
- open(name, how)
- read(fd, buf, len)
- write(fd, buf, len)
- sync(fd)
- seek(fd, pos)
- close(fd)
- unlink(name)

Windows

- CreateFile(name, CREATE)
- CreateFile(name, OPEN)
- ReadFile(handle, …)
- WriteFile(handle, …)
- FlushFileBuffers(handle, …)
- SetFilePointer(handle, …)
- CloseHandle(handle, …)
- DeleteFile(name)
- CopyFile(name)
- MoveFile(name)

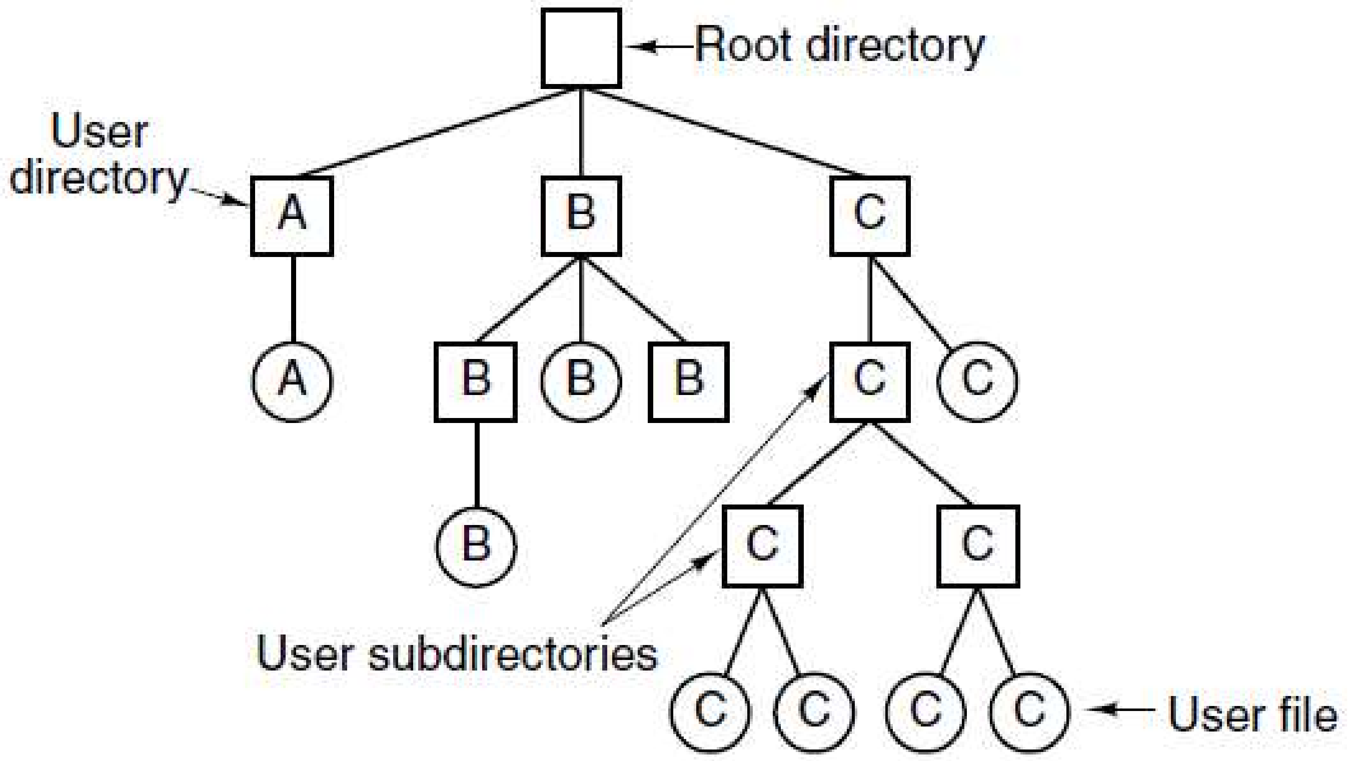
Directories

- Directories serve two purposes
  - For users, they provide a structured way to organize files
  - For the file system, they provide a convenient naming interface that allows the implementation to separate logical file organization from physical file placement on the disk
- Most file systems support multi-level directories
  - Naming hierarchies (/, /usr, /usr/local/, …)
- Most file systems support the notion of a current directory
  - Relative names specified with respect to current directory
  - Absolute names start from the root of directory tree

Directory Internals

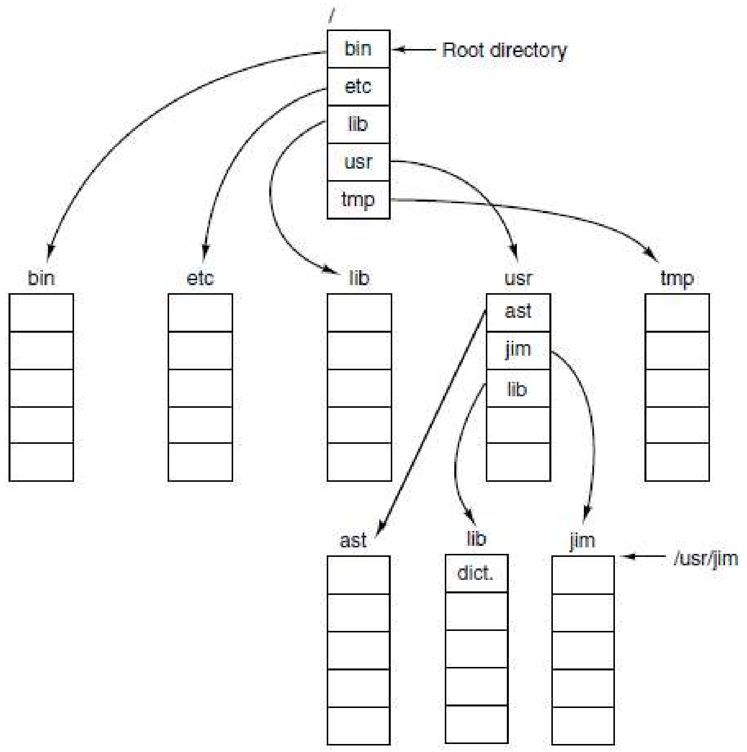
- A directory is a list of entries
  - <name, location>
  - Name is just the name of the file or directory
  - Location depends upon how file is represented on disk
- List is usually unordered (effectively random)
  - Entries usually sorted by program that reads directory
- Directories typically stored in files

Hierarchical Directory



Path Names

- **Absolute path name** consisting of path from the rot directory to the file
- Let’s say you want to open “/one/two/three”
- What does the file system do?
  - Open directory “/” (well known, can always find)
  - Search for the entry “one”, get location of “one” (in dir entry)
  - Open directory “one”, search for “two”, get location of “two”
  - Open directory “two”, search for “three”, get location of “three”
  - Open file “three”
- Systems spend a lot of time walking directory paths
  - This is why open is separate from read/write
  - OS will cache prefix lookups for performance
    - /a/b, /a/bb, /a/bbb, etc., all share “/a” prefix



Basic Directory Operations

Unix

- Directories implemented in files
  - Use file ops to create dirs
- C runtime library provides a higher-level abstraction for reading directories
  - opendir(name)
  - readdir(DIR)
  - seekdir(DIR)
  - closedir(DIR)

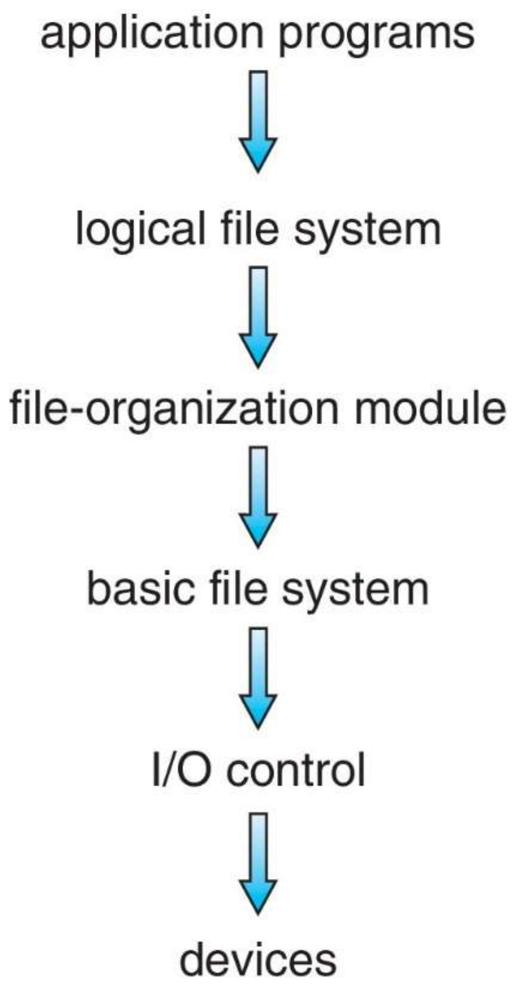
NT

- Explicit dir operations
  - CreateDirectory(name)
  - RemoveDirectory(name)
- Very different method for reading directory entries
  - FindFirstFile(pattern)
  - FindNextFile()

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- **File system** resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
  - I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)
- **File control block** (**FCB**) – storage structure consisting of information about a file
- **Device driver** controls the physical device
- File system organized into layers

Layered File System



File System Layers

- **Device drivers** manage I/O devices at the I/O control layer
  - Given commands like “read drive1, cylinder 72, track 2, sector 10, into memory location 1060” outputs low-level hardware specific commands to hardware controller
- **Basic file system** given command like “retrieve block 123” translates to device driver
- Also manages memory buffers and caches (allocation, freeing, replacement)
  - Buffers hold data in transit
  - Caches hold frequently used data
- **File organization module** understands files, logical address, and physical blocks
- Translates logical block # to physical block #
- Manages free space, disk allocation

File System Layers (Cont.)

- **Logical file system** manages metadata information
  - Translates file name into file number, file handle, location by maintaining file control blocks (**i-nodes** in UNIX)
  - Directory management
  - Protection
- Layering useful for reducing complexity and redundancy, but adds overhead and can decrease performance. To translate file name into file number, file handle, location can do by maintaining file control blocks (**i-nodes** in UNIX)
  - Logical layers can be implemented by any coding method according to OS designer

File System Layers (Cont.)

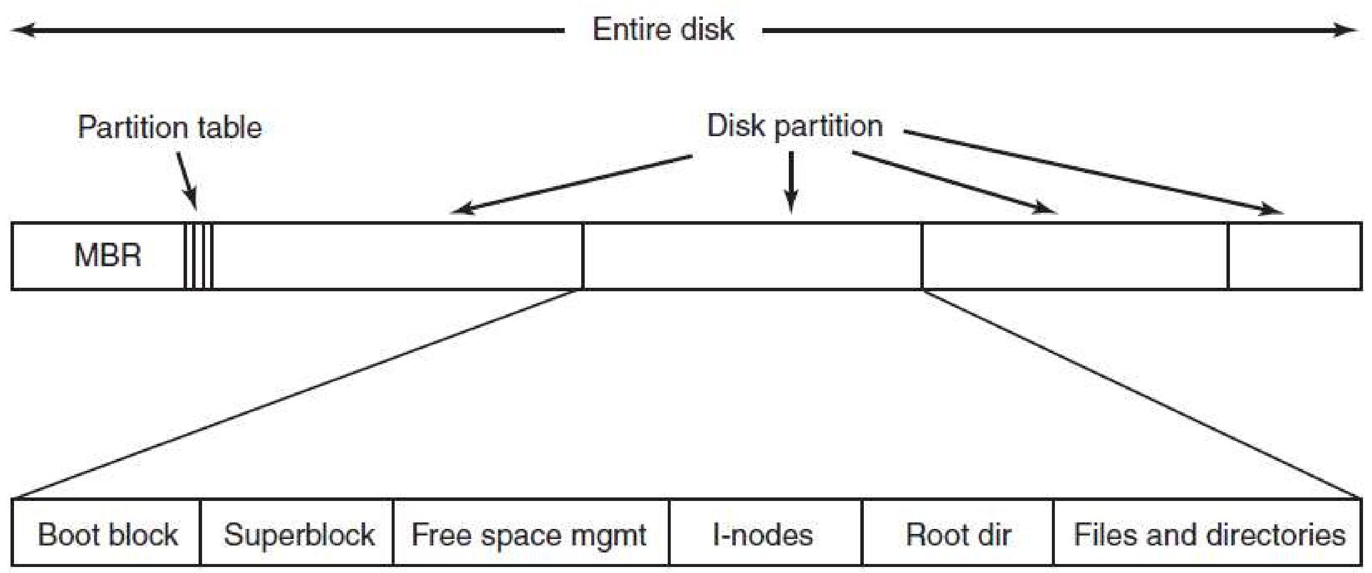
- Many file systems, sometimes many within an operating system
  - Each with its own format (CD-ROM is ISO 9660; Unix has **UFS**, FFS; Windows has FAT, FAT32, NTFS as well as floppy, CD, DVD Blu-ray, Linux has more than 130 types, with **extended file system** ext3 and ext4 leading; plus distributed file systems, etc.)
  - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE

File System Layout

How do file systems use the disk to store files?

- File systems define a block size (e.g., 4KB)
  - Disk space is allocated in granularity of blocks
- A “Master Block” determines location of root directory
  - Always at a well-known disk location
  - Often replicated across disk for reliability
- A free map determines which blocks are free, allocated
  - Usually a bitmap, one bit per block on the disk
  - Also stored on disk, cached in memory for performance
- Remaining disk blocks used to store files (and dirs)
  - There are many ways to do this

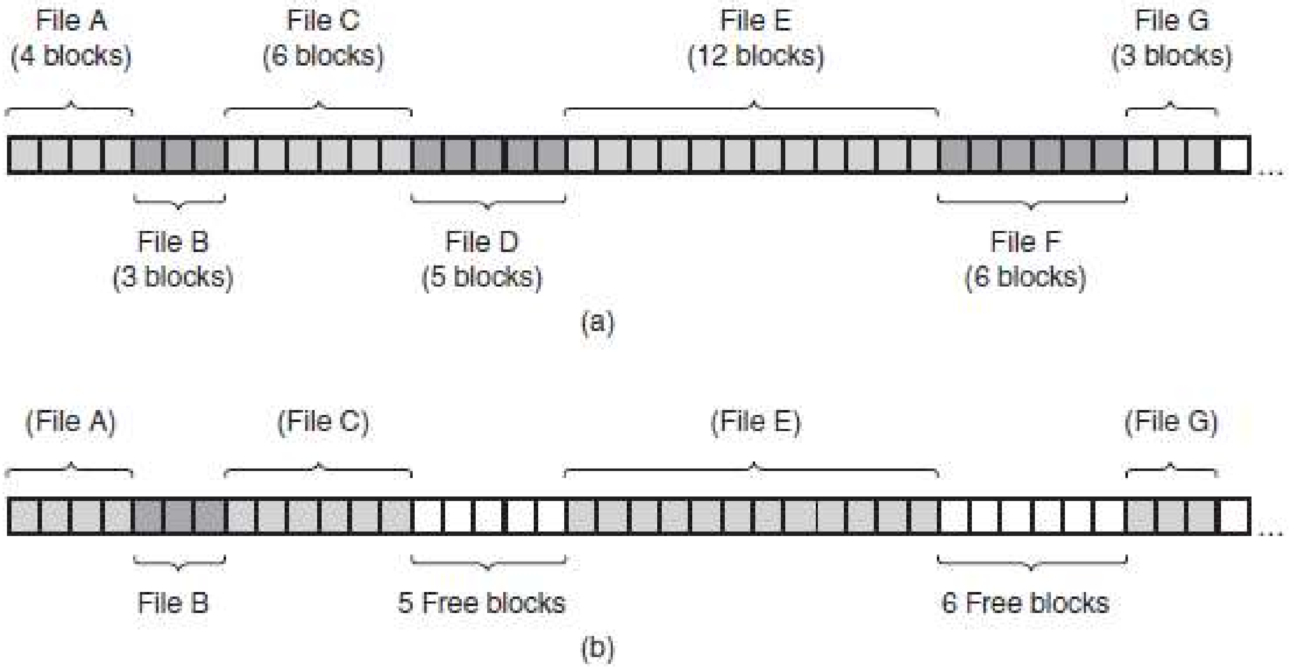
File System Layout



Disk Layout Strategies

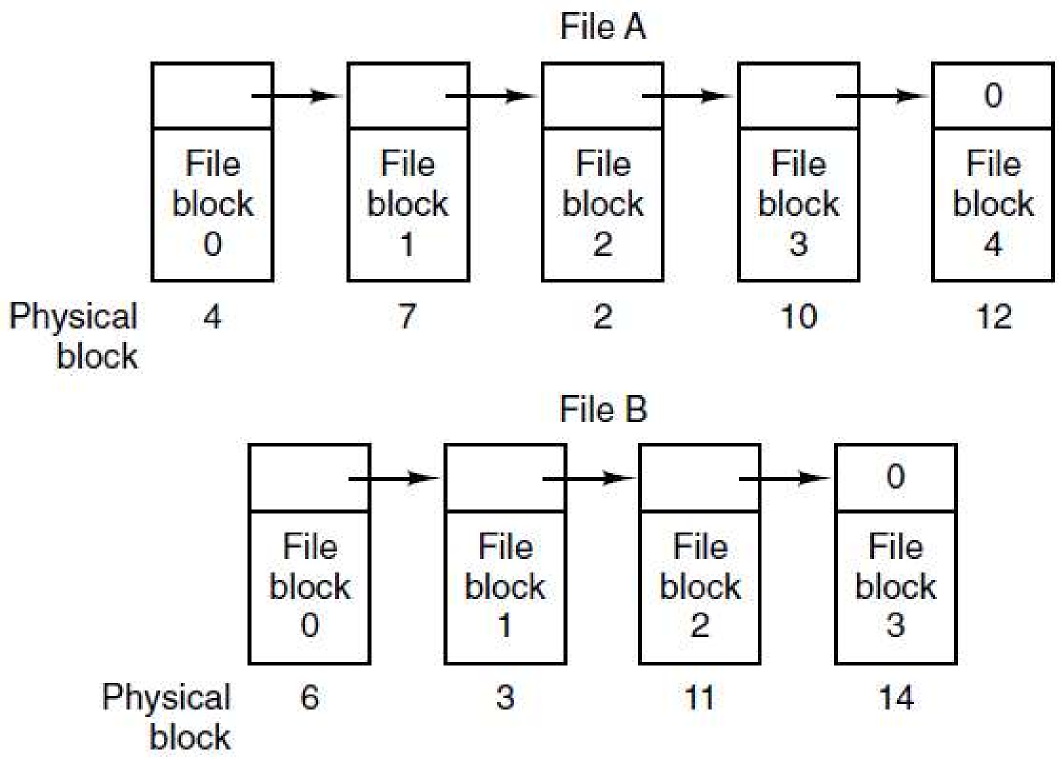
- Files span multiple disk blocks
- How do you find all of the blocks for a file?
  1. Contiguous allocation
     - Fast, simplifies directory access
     - Inflexible, causes fragmentation, needs compaction
  2. Linked structure
     - Each block points to the next, directory points to the first
     - Good for sequential access, bad for all others
     - Entire table must be in memory
  3. Indexed structure (indirection, hierarchy)
     - An “index block” contains pointers to many other blocks
     - Handles random better, still good for sequential
     - May need multiple index blocks (linked together)

Implementing Files Contiguous Layout

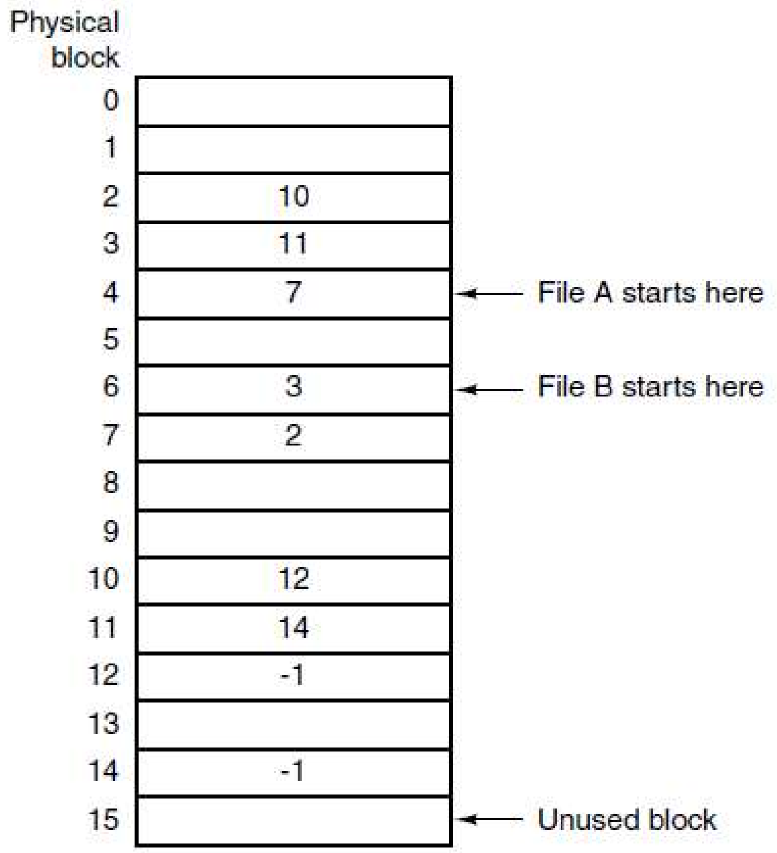


(a) Contiguous allocation of disk space for seven files
(b) The state of the disk after files *D* and *F* have been removed

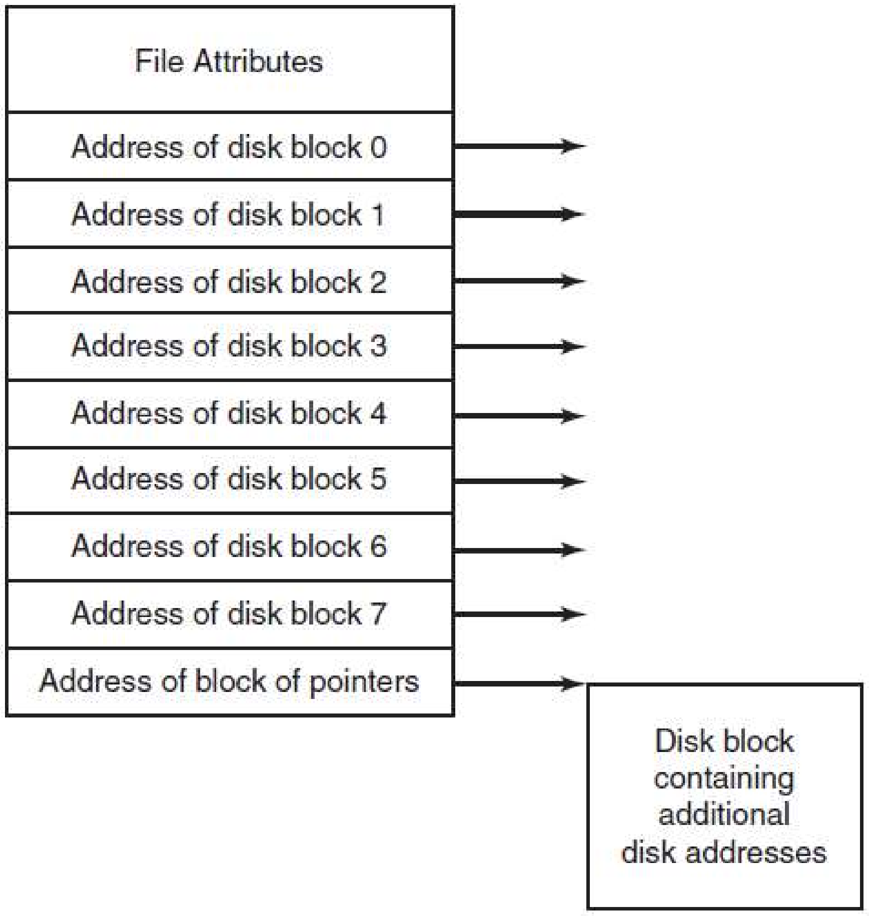
Implementing Files Linked List Allocation



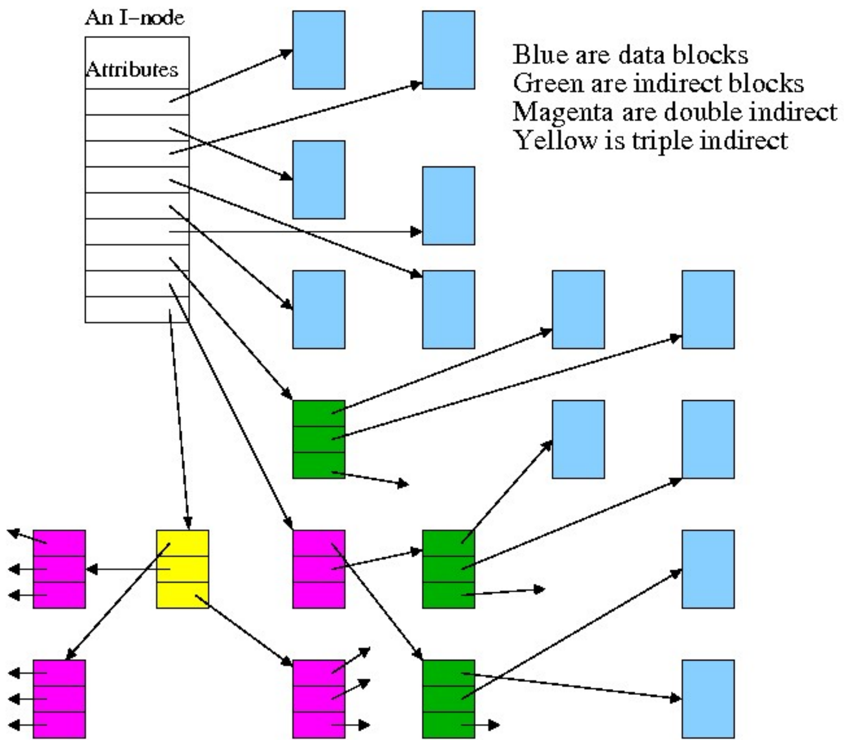
Implementing Files Linked List – Table in Memory



Implementing Files i-nodes



i-node structure

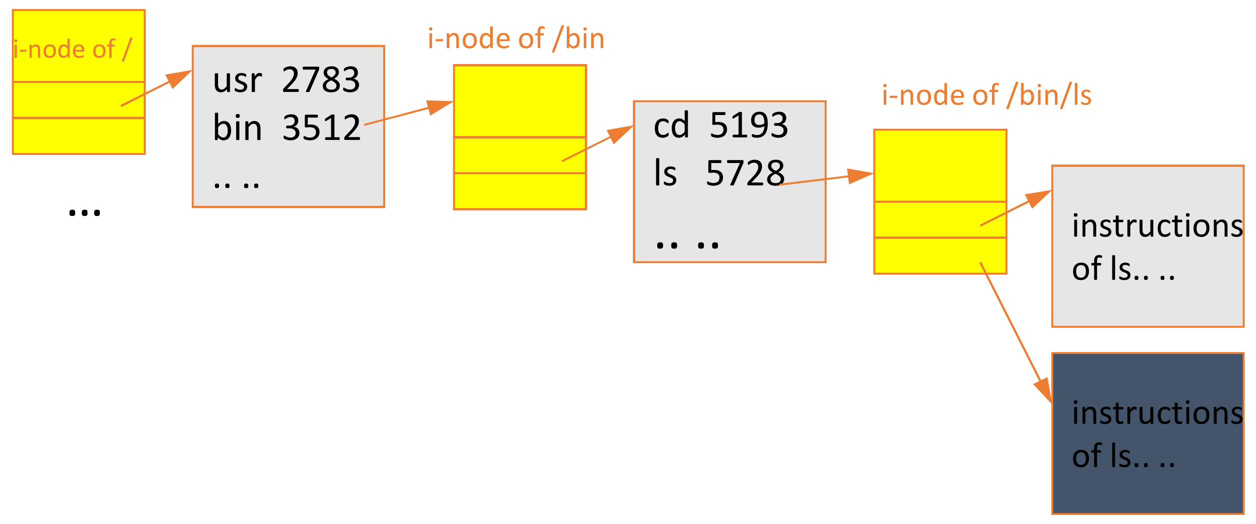

An I–node
Attributes
Blue are data blocks
Green are indirect blocks
Magenta are double indirect
Yellow is triple indirect

Unix i-nodes and Path Search

- Unix i-nodes are not directories
- i-nodes describe where on the disk the blocks for a file are placed
  - Directories are files, so i-nodes also describe where the blocks for directories are placed on the disk
- Directory entries map file names to i-nodes
  - To open “/one”, use Master Block to find i-node for “/” on disk
  - Open “/”, look for entry for “one”
  - This entry gives the i-node number for the i-node for “one”
  - Read the i-node for “one” into memory
  - The i-node says where first data block is on disk
  - Read that block into memory to access the data in the file

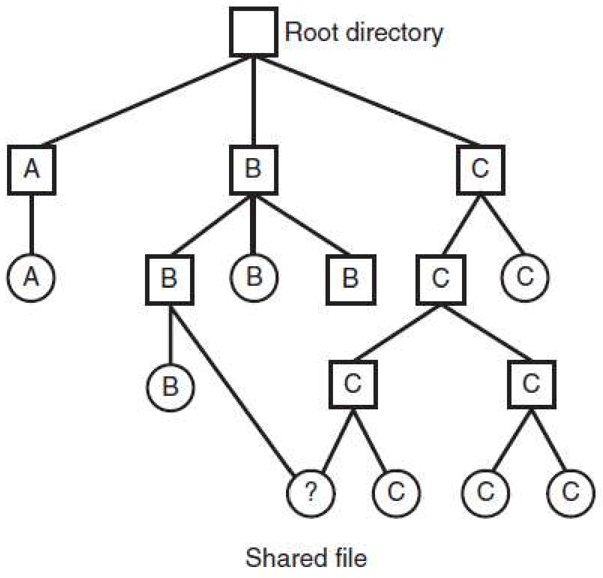
Example: read /bin/ls

- /bin/ls

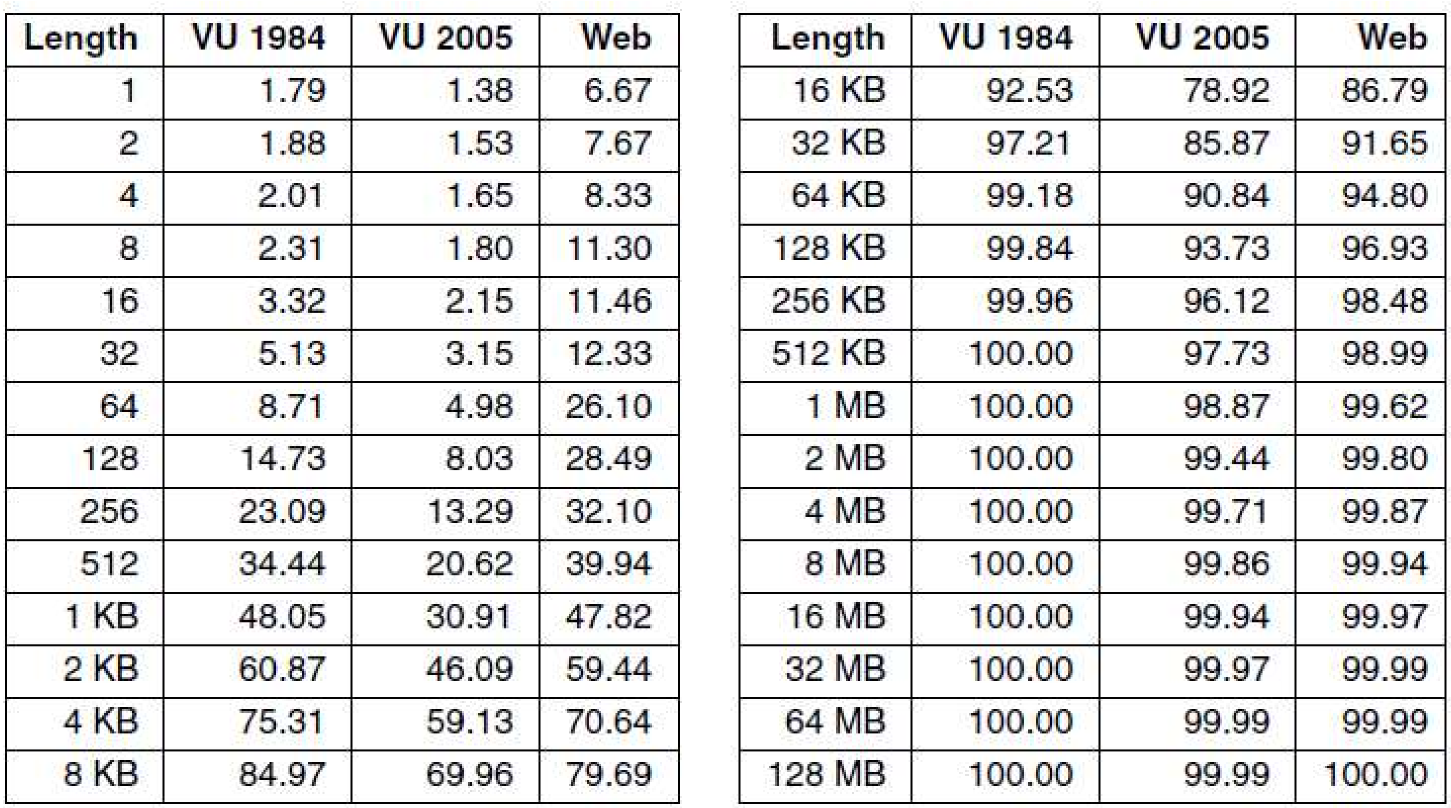


Sharing Files btw. Directories

- Links (or hard links)
  - ln source_file target_dir
    - Simply create another link from target_dir to the **inode** of source_file (the inode is not duplicated)
    - Now two directories have links to source_file
    - What if we remove one?
    - Now you understand why the system call to remove a file is named “unlink”?

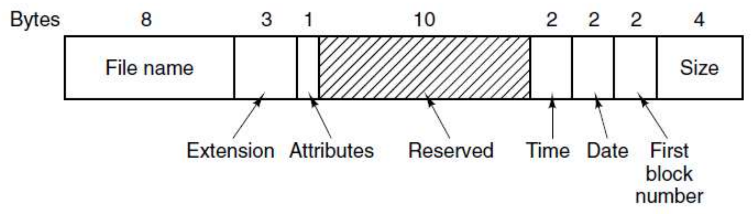


Percentage of files smaller than a given size



Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69
16 KB	92.53	78.92	86.79
32 KB	97.21	85.87	91.65
64 KB	99.18	90.84	94.80
128 KB	99.84	93.73	96.93
256 KB	99.96	96.12	98.48
512 KB	100.00	97.73	98.99
1 MB	100.00	98.87	99.62
2 MB	100.00	99.44	99.80
4 MB	100.00	99.71	99.87
8 MB	100.00	99.86	99.94
16 MB	100.00	99.94	99.97
32 MB	100.00	99.97	99.99
64 MB	100.00	99.99	99.99
128 MB	100.00	99.99	100.00

Example File System : MS-DOS File System



- The FAT file system comes in three versions, FAT-12, FAT-16, Fat-32
- Maximum partition size for different block sizes. The empty boxes represent forbidden combinations

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Sources and References

- Andrew S. Tanenbaum, Herbert Bos, Modern Operating 4th edition, Pearson, 2015
- Presentations by Ding Yuan, ECE Dept., University of Toronto
- Operating System Concepts 10th book official slides by Abraham Silberschatz, Greg Gagne, Peter B. Galvin