Modern Operating Systems

Chapter 4 – Example File Systems

Zhang Yang

Spring 2022

Content of the Lecture

- 4.5 Example File Systems

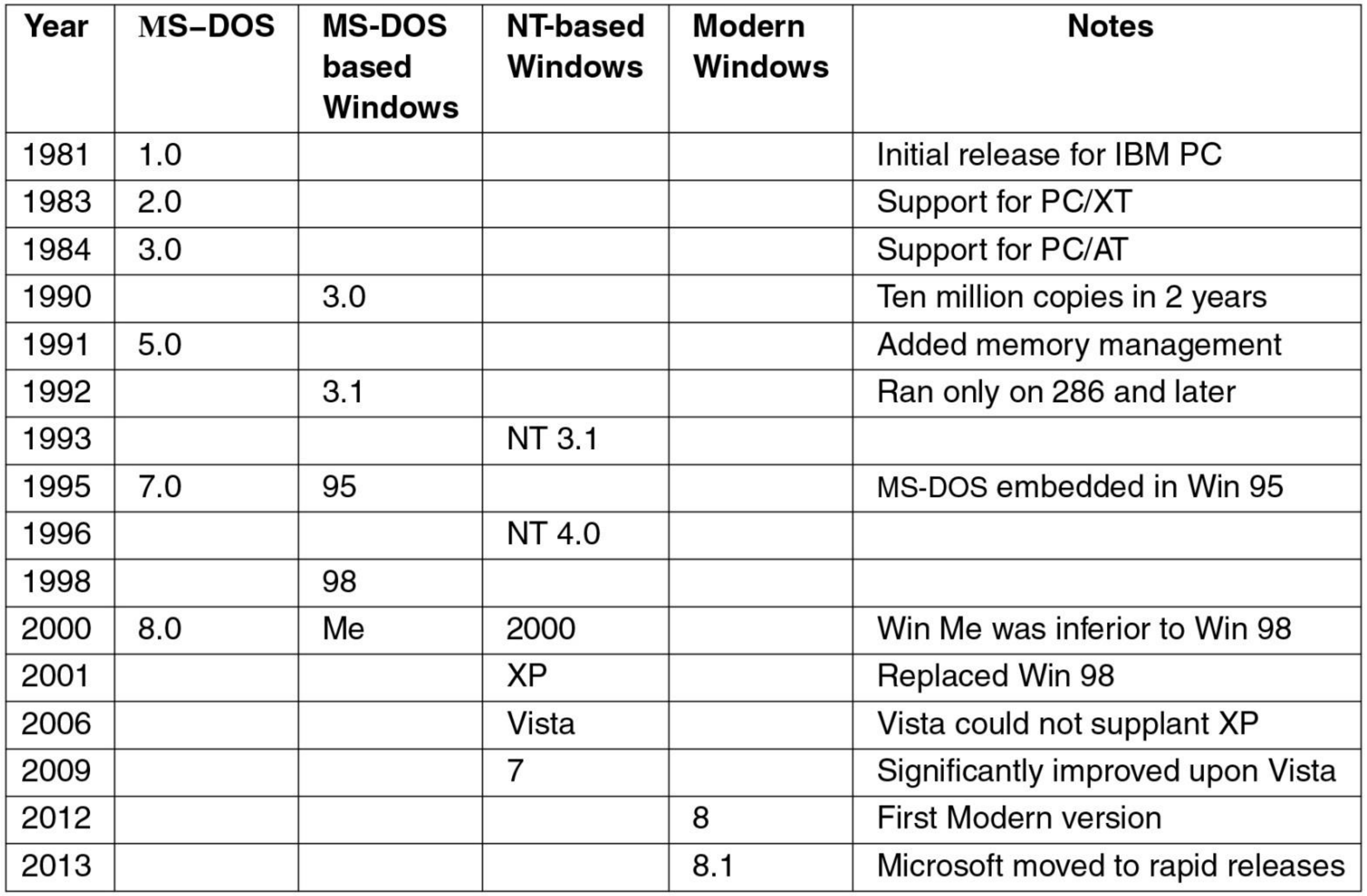
Example File Systems

- The MS-DOS File System
- Windows 98 File System (supplemented)
- UNIX V7 File System
- Linux Ext2 File System (see chapter 10.6.3)

Example File Systems Overview

- 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 40 types, with **extended file system** ext2 and ext3 leading; plus distributed file systems, etc.)
  - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE.

History of Windows



Year	MS–DOS	MS-DOS based Windows	NT-based Windows	Modern Windows	Notes
1981	1.0				Initial release for IBM PC
1983	2.0				Support for PC/XT
1984	3.0				Support for PC/AT
1990		3.0			Ten million copies in 2 years
1991	5.0				Added memory management
1992		3.1			Ran only on 286 and later
1993			NT 3.1		
1995	7.0	95			MS-DOS embedded in Win 95
1996			NT 4.0		
1998		98			
2000	8.0	Me	2000		Win Me was inferior to Win 98
2001			XP		Replaced Win 98
2006			Vista		Vista could not supplant XP
2009			7		Significantly improved upon Vista
2012				8	First Modern version
2013				8.1	Microsoft moved to rapid releases
2015				10	

Figure 11-1. Major releases in the history of Microsoft operating systems for desktop PCs.

The MS-DOS File System (1)

- The first IBM PC used.
- Main file system of Windows 98 and Windows ME.
- Still supported on Windows 2000, Windows XP, and Windows Vista.
- MS-DOS file system and its extension (FAT-32) have become widely used for many embedded systems.

The MS-DOS File System (2)

- Use 8+3 file names.
- DOS 1.0 limited to one directory.
- Later versions allows hierarchical file system.
  - Directory can nested to an arbitrary depth.
- Directories are variable sized.
- But directory entries are fixed size of 32 bytes.

The MS-DOS File System (3)

- Directory Entry
  - Attributes: indicate a file is read-only, archived, hidden, or system file.
  - Time: hours (5 bits), minutes (6 bits), seconds (5 bits)
  - Date: year (7 bits), month (4 bits), day (5 bits)
  - Size: in theory can be 4GB, but in fact 2GB or less.

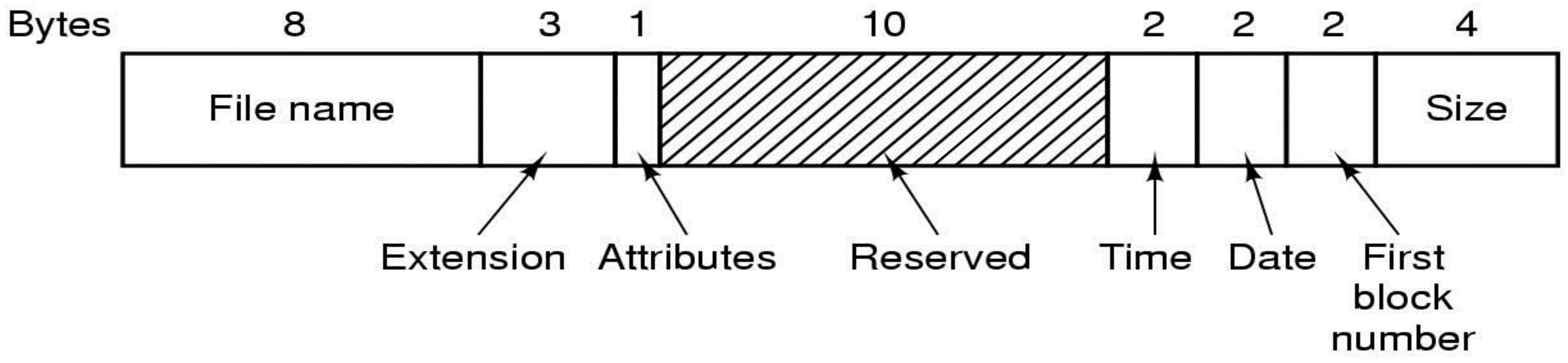


Figure 4-30 The MS-DOS directory entry

The MS-DOS File System (4)

- The highest expressible year in MS-DOS is 2107.
  - The counter starts at 1980.
- Use file allocation table to keep track of file addresses.
- Each directory entry only gives the first block number.
  - Which is used to index into the FAT table.

The MS-DOS File System (5)

- FAT file system has three versions for MS-DOS: FAT-12, FAT-16, FAT-32.
  - Each differs from the bits a disk address contains.
  - FAT-32 introduced with 2nd release of Windows 95.
- Disk blocks are multiples of 512 bytes.
  - Can be different for each partition.
- Allowable block size differing by variants.

The MS-DOS File System (6)

- FAT-32
  - 28-bit disk address
  - Partition size: theoretically max 8TB ($2^{28} \times 2^{15}$ bytes), but usually limited to 2TB.

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

Figure 4-31 Maximum partition for different block sizes. The empty boxes represent forbidden combinations

The MS-DOS File System (7)

- Advantages of FAT-32 over FAT-16
  - Support larger disks.
  - An 8-GB disk using FAT-32 can be a single partition. Using FAT-16 has to be four partitions.
  - For a given size disk partition, a smaller block size can be used.

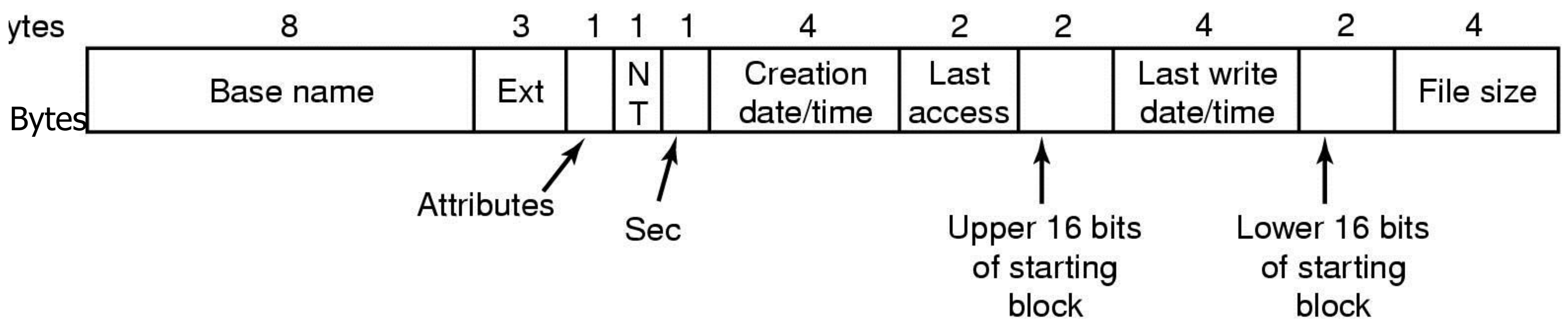
The MS-DOS File System (8)

- Exercise
  - Problem: The MS-DOS FAT-16 table contains 64K entries. Suppose that one of the bits had been needed for some other purpose and that the table contained exactly 32768 entries instead. With no other changes, what would the largest MS-DOS file have been under this condition?
  - Solution: The largest block is 32KB. With 32,768 of these blocks, the biggest file would be 1 GB.

The Windows 98 File System (1)

- To accommodate long file names.
  - Up to 255 characters.
- To provide backwards compatibility to MS-DOS files.
  - Must use the MS-DOS directory structure.
- Idea: keep two names for each file.
  - Create a MS-DOS name from the given (long) name.
  - Both name are then stored in the directory structure.

The Windows 98 File System (2)



The extended MOS-DOS directory entry used in Windows 98

NT: compatibility with Windows NT

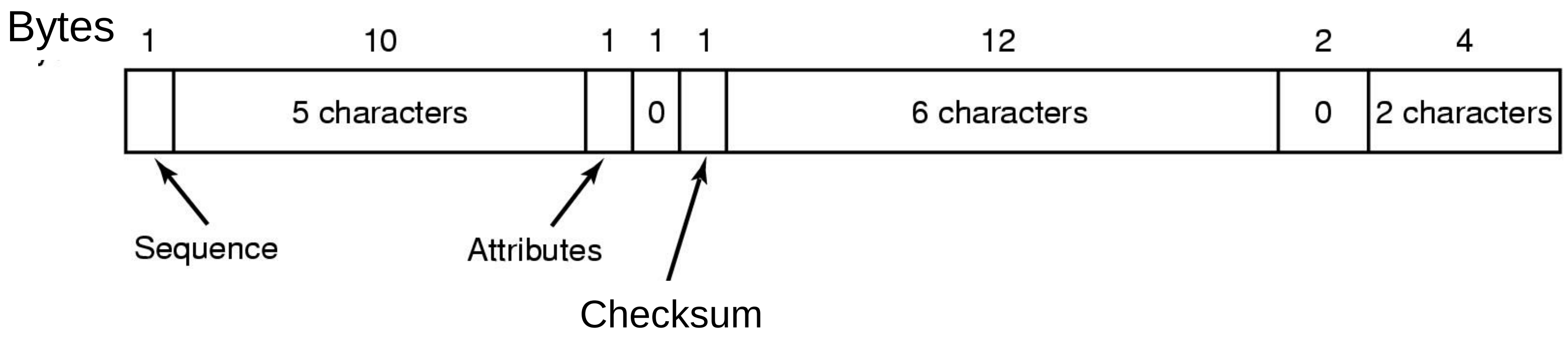
Sec: solve the problem that it is not possible to store the time of day in a 16-bit field. (Creation time is accurate to 10 msec)

Creation time

Last access

Upper 16bits of starting block

The Windows 98 File System (3)

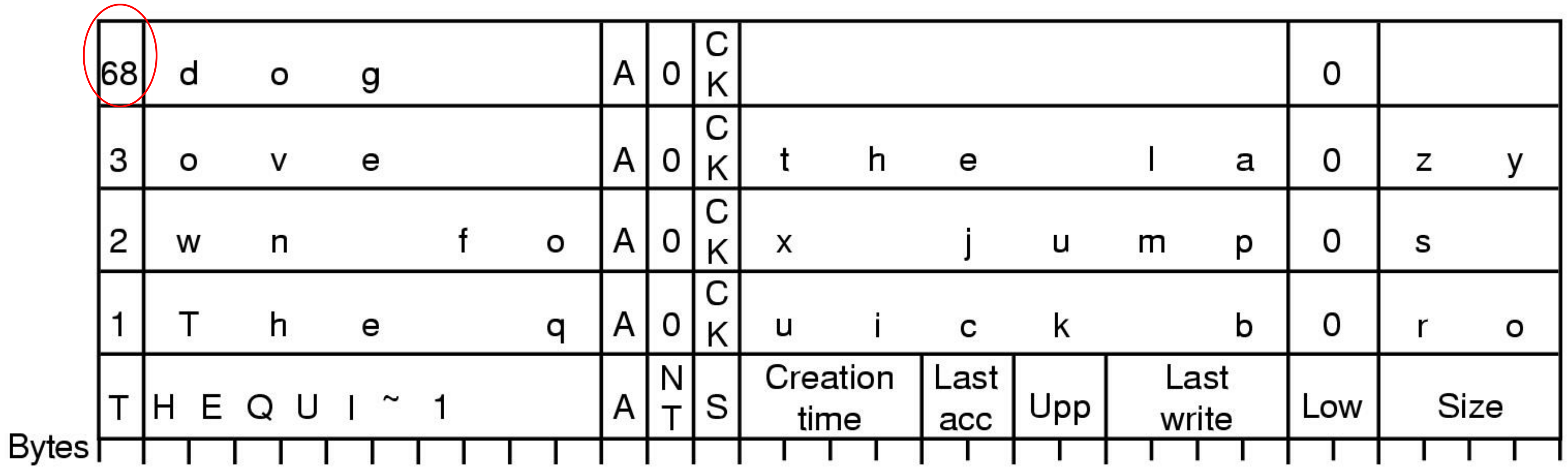


An entry for (part of) a long file name in Windows 98

sequence: 6 bit. The maximum file name = 63 * 13 = 819. But in fact limited to 260 chars for historical reasons.

Attributes field = 0x0f for long file name

The Windows 98 File System (4)



68	d o g	A	0	C K					0	
3	o v e	A	0	C K	t h	e		l a	0	z y
2	w n f o	A	0	C K	x	j	u	m p	0	s
1	T h e q	A	0	C K	u i	c	k	b	0	r o
T	H E Q U I ~ 1	A	N T	S	Creation time	Last acc	Upp	Last write	Low	Size

Bytes

An example of how a long name is stored in Windows 98

The UNIX V7 File System (1)

- Introduced in PDP-11.
- Organized in the form of a tree starting from root.
- File names are up to 14 characters and can contain any ASCII characters except / and NUL.
- Each directory entry has two fields:
  - File name, and the # of i-node for that file (2 bytes)
- Number of files in the file system limited to 64K.

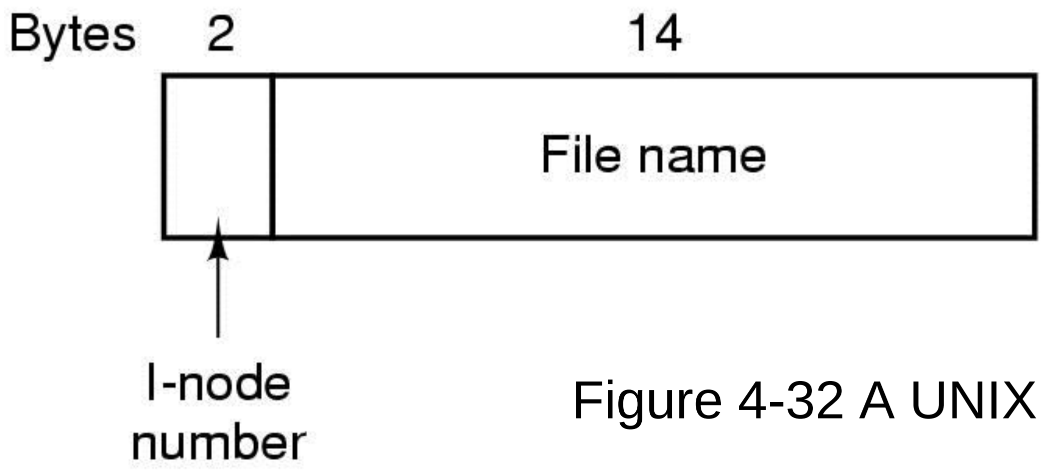
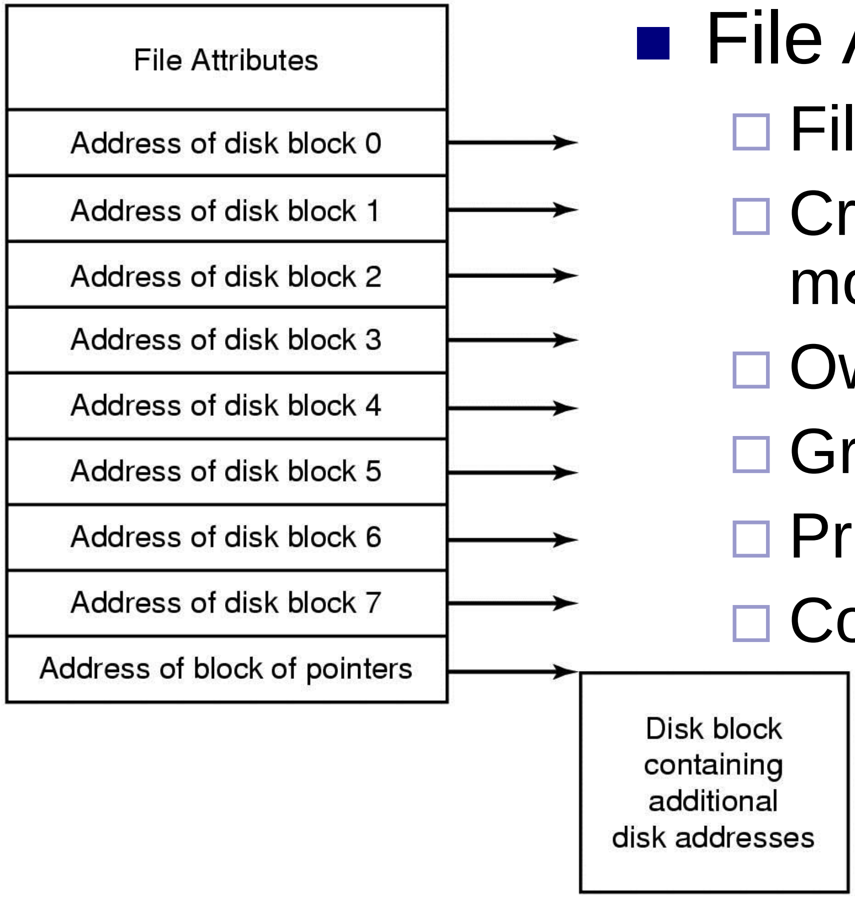


Figure 4-32 A UNIX V7 directory entry

The UNIX V7 File System (2)



- File Attributes:
  - File size
  - Creation, last access, last modification time
  - Owner
  - Group
  - Protection
  - Count

Figure 4-13 An Example i-node

The UNIX V7 File System (3)

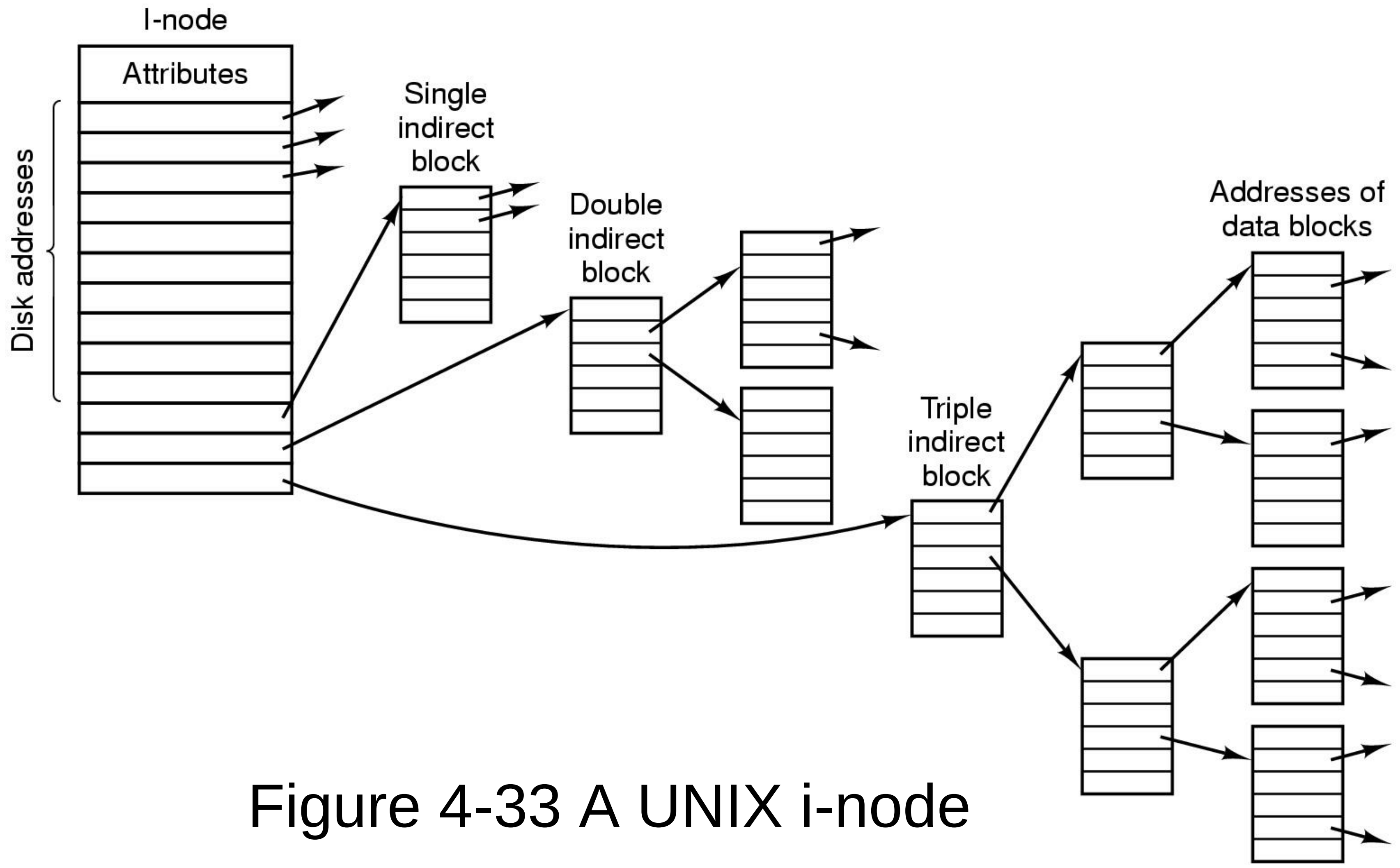
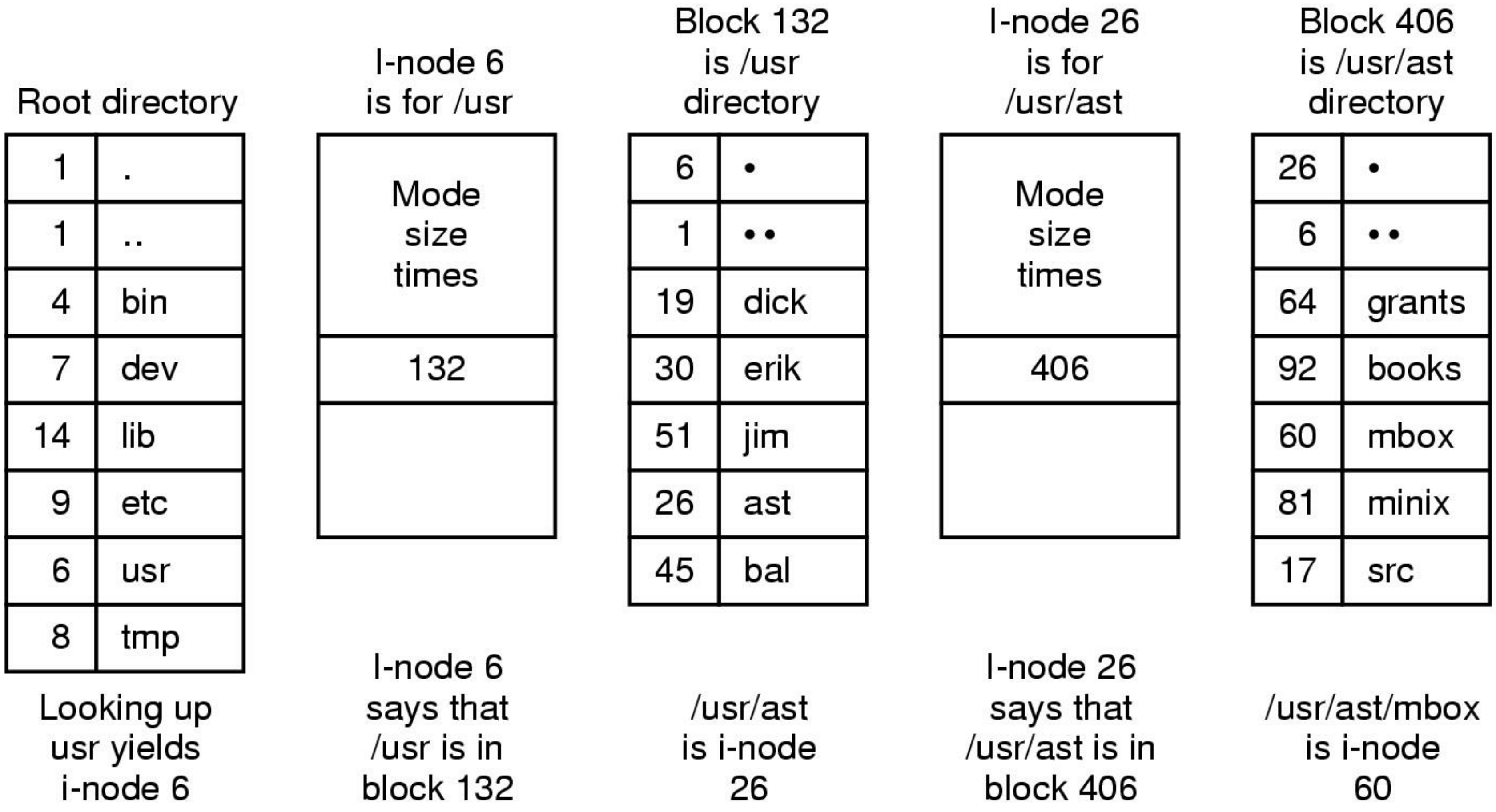


Figure 4-33 A UNIX i-node

The UNIX V7 File System (4)

- The steps in looking up */usr/ast/mbox:*
  - The file system locates the root directory. It can be anywhere on the disk, but its i-node is located at a fixed place.
  - The file system looks up the first component of the path (usr) and find its i-node number. From this i-node, the system locates the directory for /usr and looks up the next component (ast) in it.
  - When the entry for ast is found in directory /usr, i-node for the directory /usr/ast can be obtained. From this i-node it can find the directory itself and lookup mbox.
  - The i-node for this file is then read into memory and kept there until the file is closed.

The UNIX V7 File System (5)



Root directory
1 .
1 ..
4 bin
7 dev
14 lib
9 etc
6 usr
8 tmp
Looking up usr yields i-node 6
I-node 6 is for /usr
Mode size times
132
I-node 6 says that /usr is in block 132
Block 132 is /usr directory
6 •
1 ••
19 dick
30 erik
51 jim
26 ast
45 bal
/usr/ast is i-node 26
I-node 26 is for /usr/ast
Mode size times
406
I-node 26 says that /usr/ast is in block 406
Block 406 is /usr/ast directory
26 •
6 ••
64 grants
92 books
60 mbox
81 minix
17 src
/usr/ast/mbox is i-node 60

Figure 4-35 The steps in looking up */usr/ast/mbox*

The UNIX V7 File System (6)

- Exercise
  - Problem1: A Unix file system has 1KB blocks and 4 byte disk address. What is the maximum file size if i-nodes contain 10 direct entries, and one single, double and triple indirect entry each?
  - Solution: The i-node holds 10 pointers. The single indirect block holds 256 pointers. The double indirect block is good for 256^2 pointers. The triple indirect block is good for 256^3 pointers. Adding these up, we get a maximum file size of 16,843,018 blocks, which is about 16.06 GB.

The UNIX V7 File System (7)

- Exercise
  - Problem2:How many disk operations are needed to fetch the i-node for the file /usr/ast/course/os/handout.t? Assume that the i-node for the root directory is in memory, but nothing else along the path is in memory. Also assume that all directories fit in one disk block.

The UNIX V7 File System (8)

- Exercise
  - Solution: The following disk reads are needed:
    - directory for /
    - i-node for */usr*
    - directory for */usr*
    - i-node for */usr/ast*
    - directory for */usr/ast*
    - i-node for */usr/ast/courses*
    - directory for */usr/ast/courses*
    - i-node for */usr/ast/courses/os*
    - directory for */usr/ast/courses/os*
    - i-node for */usr/ast/courses/os/handout.t*

In total, 10 disk reads are required.

The Linux Ext2 File System (1)

- See textbook P785 10.6.3
- Linux supports many file systems, some of them are as follows: ext, ext2, xia, minix, umsdos, msdos, vfat, proc, smb, ncp, iso9660, sysv, hpfs, affs and ufs.
- Minix was the first file system of Linux.
- EXT was released in April 1992, and was the first to use the VFS API.
- The Second Extended Filesystem (Ext2) was introduced in 1994.

The Linux Ext2 File System (2)

- Disk Layout
  - In any Ext2 partition, the first block is reserved for the partition boot sector. The rest of the Ext2 partition is split into block groups.
  - The whole area is divided into several block groups, and block groups contains several blocks.
  - Files in the same directory are stored in the same block group.
  - Files in different directories are spread among the block groups.
  -

The Linux Ext2 File System (3)

- Disk Layout (ctd.)
  - Each block group is organized as follows

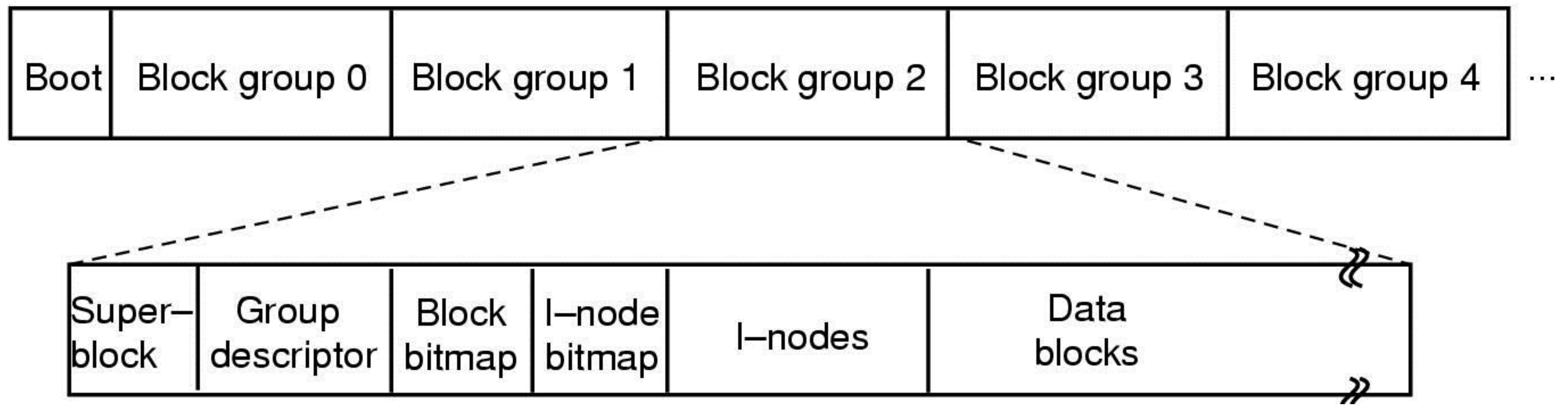


Figure 10-31 Disk Layout of the Linux ext2 file system

The Linux Ext2 File System (4)

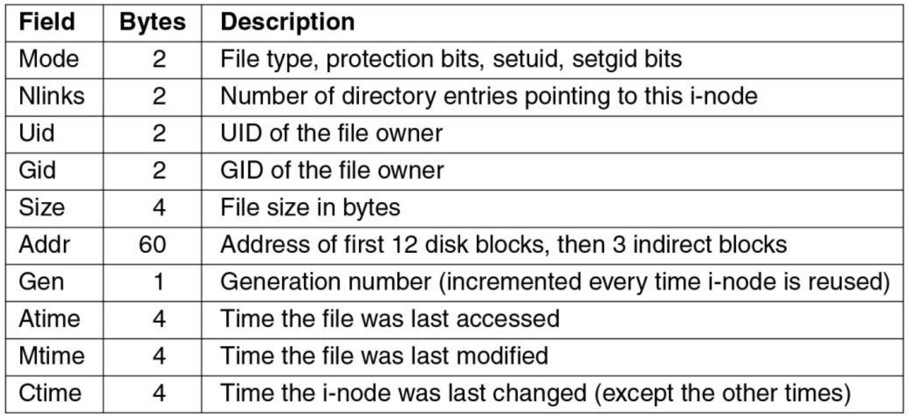
- Metadata Concepts
  - Superblock
    - The Ext2 superblock is located 1024 bytes from the start of the file system and is 1024 bytes in size. (The first two sectors are used to store boot code if necessary).
    - Back up copies are typically stored in the first file data block of each block group.
    - It contains information about the layout of the file system, including the number of i-nodes, the number of disk blocks, and the start of the list of free blocks.

The Linux Ext2 File System (5)

- Metadata Concepts (ctd.)
  - Block Group Descriptor Table
    - It contains a group descriptor data structure for every block group.
    - It contains information about the location of the bitmaps, the number of free blocks and i-nodes in the group, and the number of directories in the group.
  - Bitmaps
    - The block bitmap manages the allocation status of the blocks in the group.
    - The inode bitmap manages the allocation status of the inodes in the group.

The Linux Ext2 File System (6)

- Metadata Concepts (ctd.)
  - Inodes
    - Each inode corresponds to one file, and it stores file's primary metadata, such as file's size, ownership, and temporal information.
    - Inode is typically 128 bytes in size and is allocated to each file and directory.



Field	Bytes	Description
Mode	2	File type, protection bits, setuid, setgid bits
Nlinks	2	Number of directory entries pointing to this i-node
Uid	2	UID of the file owner
Gid	2	GID of the file owner
Size	4	File size in bytes
Addr	60	Address of first 12 disk blocks, then 3 indirect blocks
Gen	1	Generation number (incremented every time i-node is reused)
Atime	4	Time the file was last accessed
Mtime	4	Time the file was last modified
Ctime	4	Time the i-node was last changed (except the other times)

Figure 10-33. Some fields in the i-node structure in Linux

The Linux Ext2 File System (7)

- Directory

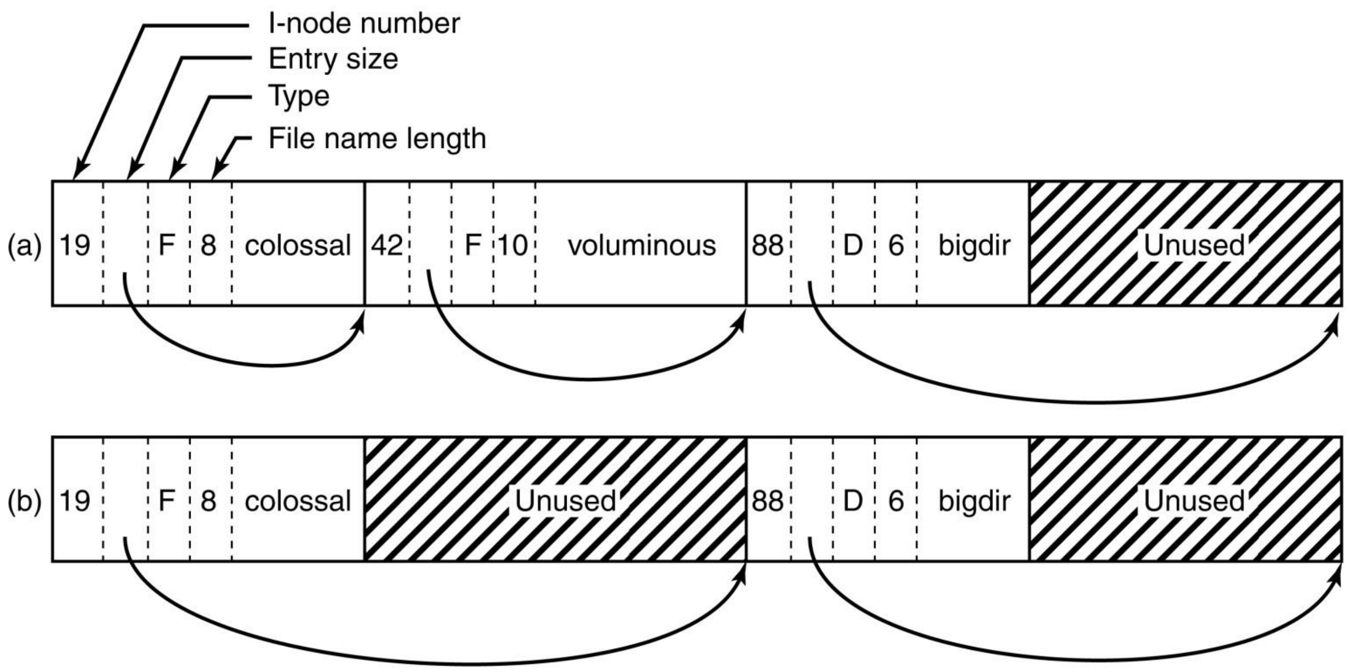


Figure 10-32. (a) A Linux directory with three files. (b) The same directory after the file voluminous has been removed.

The Linux Ext2 File System (8)

- To improve the robustness of the file system, Linux relies on journaling file systems.
- Ext3
  - Successor of the Ext2 file system
  - A journaling file system
- Ext4
  - A follow-on of Ext2
  - Also a journaling file system
  - Changes the block addressing scheme

Homework

- P335 35, 38, 41