Distributed Operating Systems

File Systems

Topics

- Files
- Directories
- File system implementation
- Example file systems
  - MS-DOS
  - Windows 98
  - UNIX

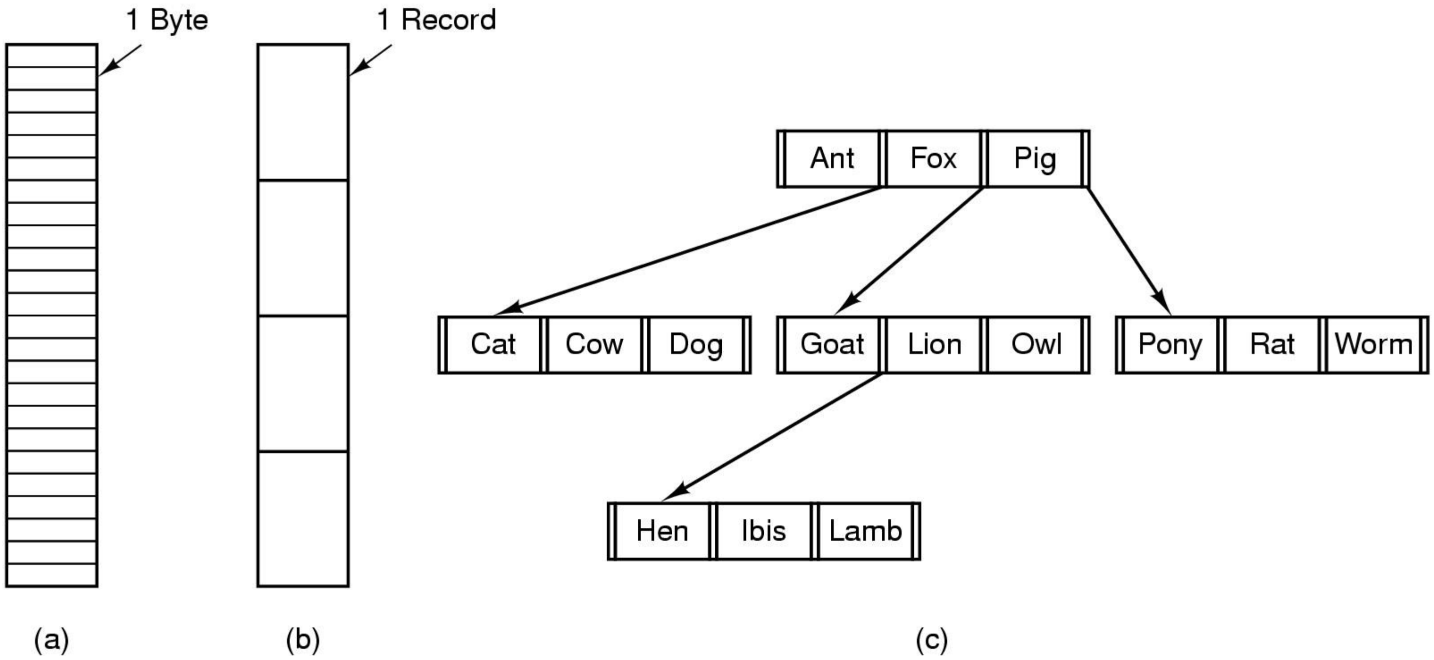
Long-term Information Storage

1. A computer system must store large amounts of data

2. Information stored must survive the termination of the process using it

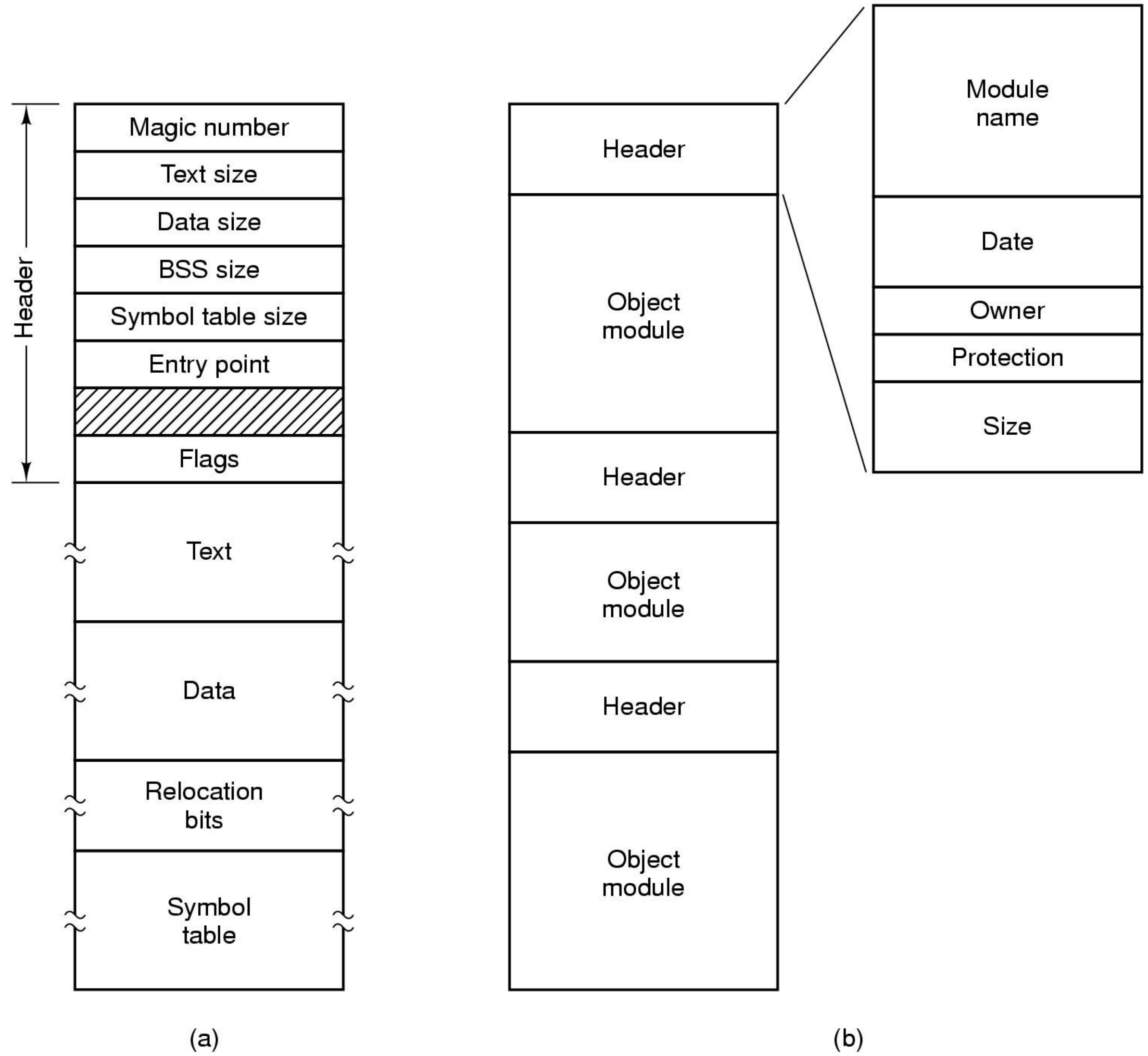
3. Multiple processes must be able to access the information concurrently

File Structure



- Three kinds of files
  - byte sequence
  - record sequence
  - tree

File Types

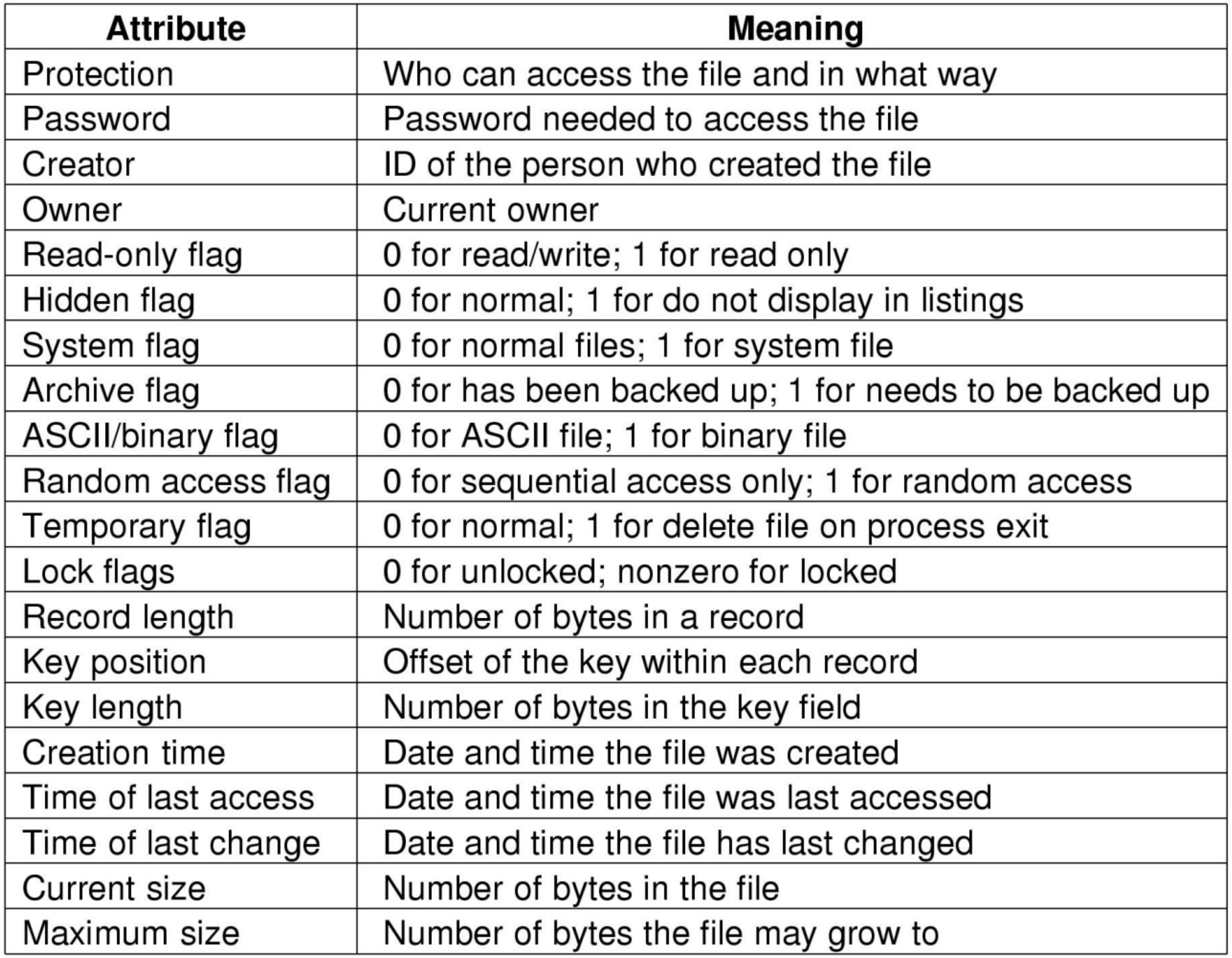


(a) An executable file (b) An archive

File Access

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was magnetic tape
- Random access
  - bytes/records read in any order
  - essential for data base systems
  - read can be …
    - move file marker (seek), then read or …
    - read and then move file marker

File Attributes



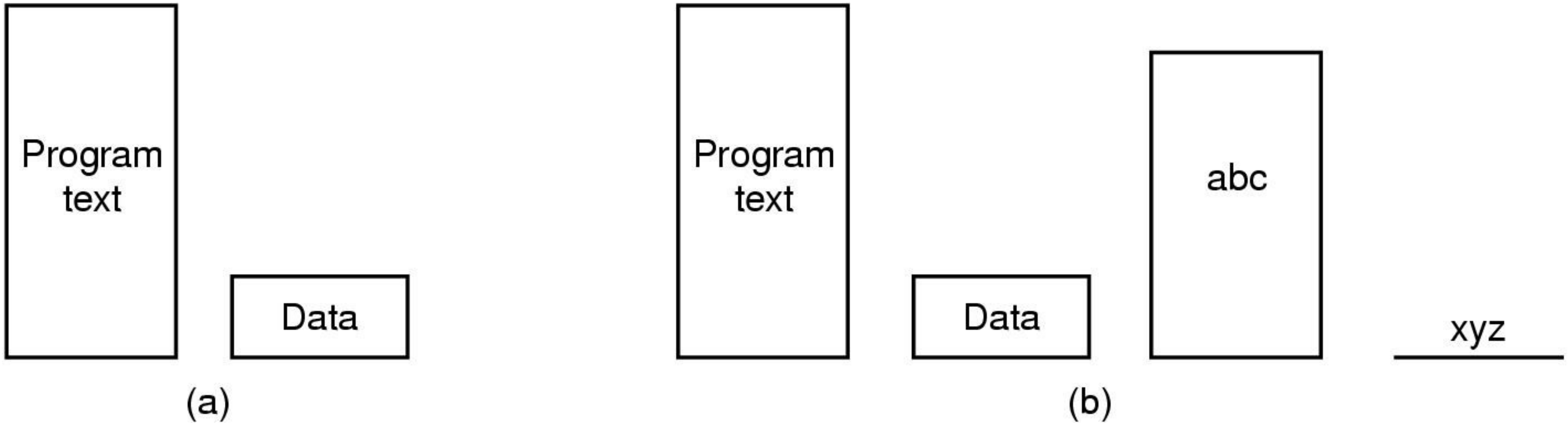
Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Possible file attributes

File Operations

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

Memory-Mapped Files



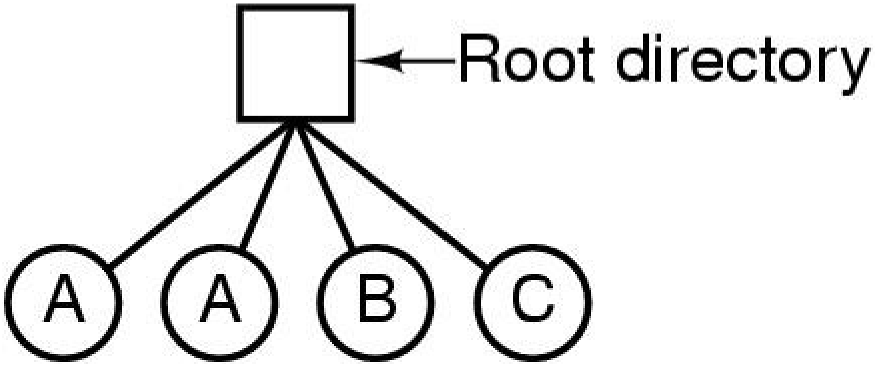
(a) Segmented process before mapping files into its address space

(b) Process after mapping

existing file *abc* into one segment

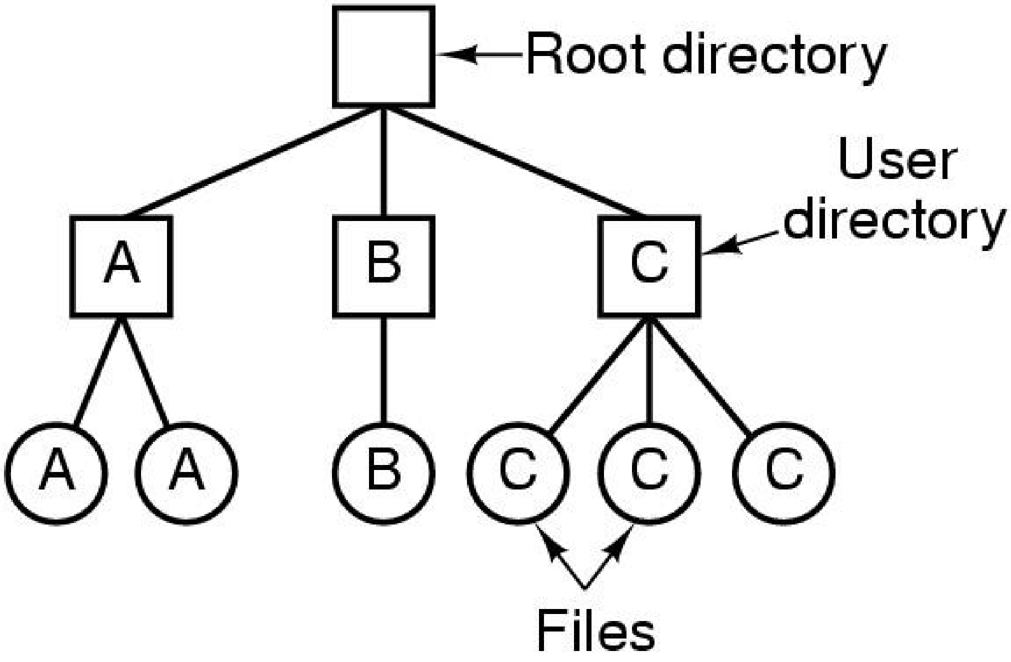
creating new segment for *xyz*

Directories
Single-Level Directory Systems



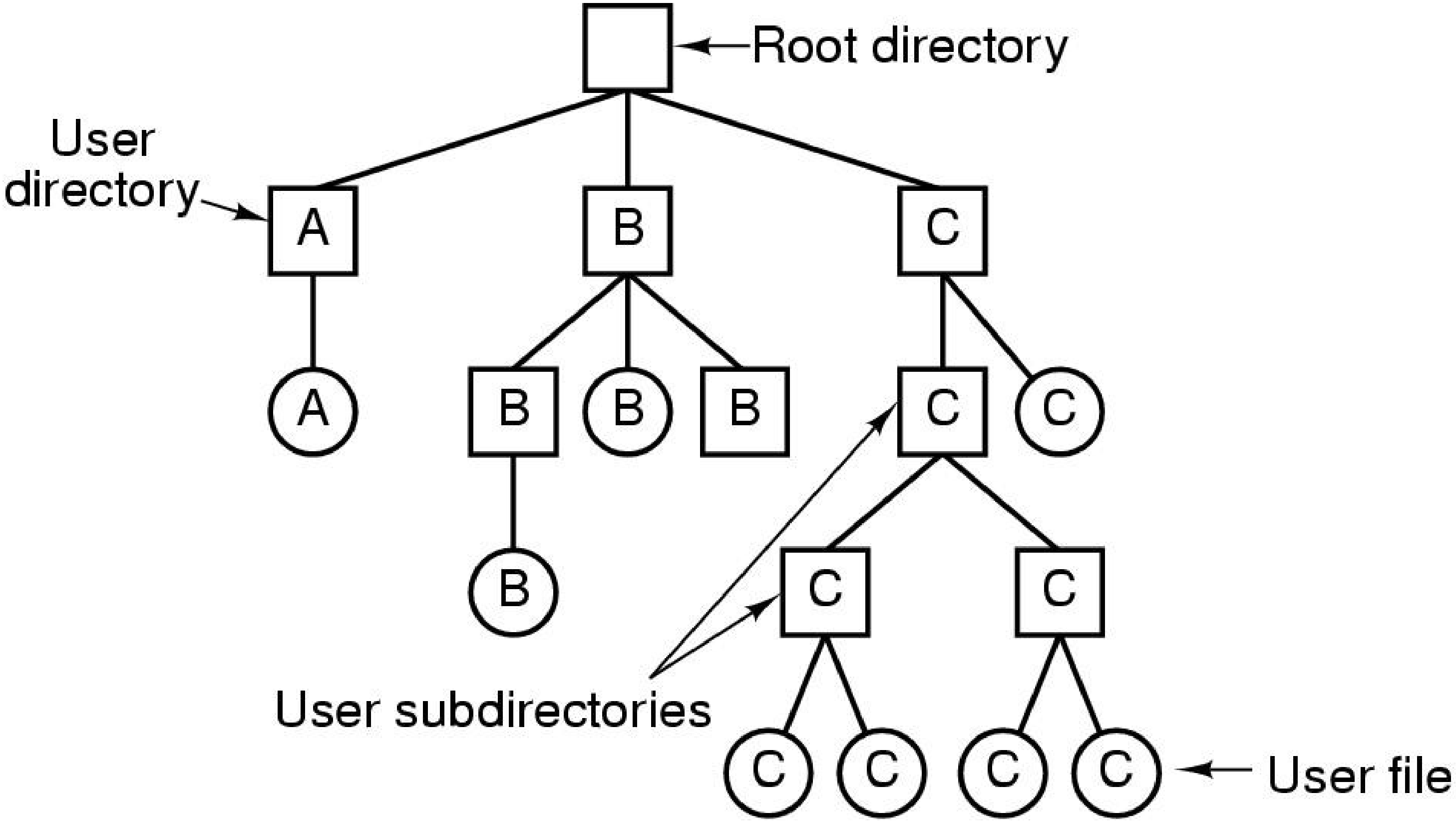
- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

Two-level Directory Systems



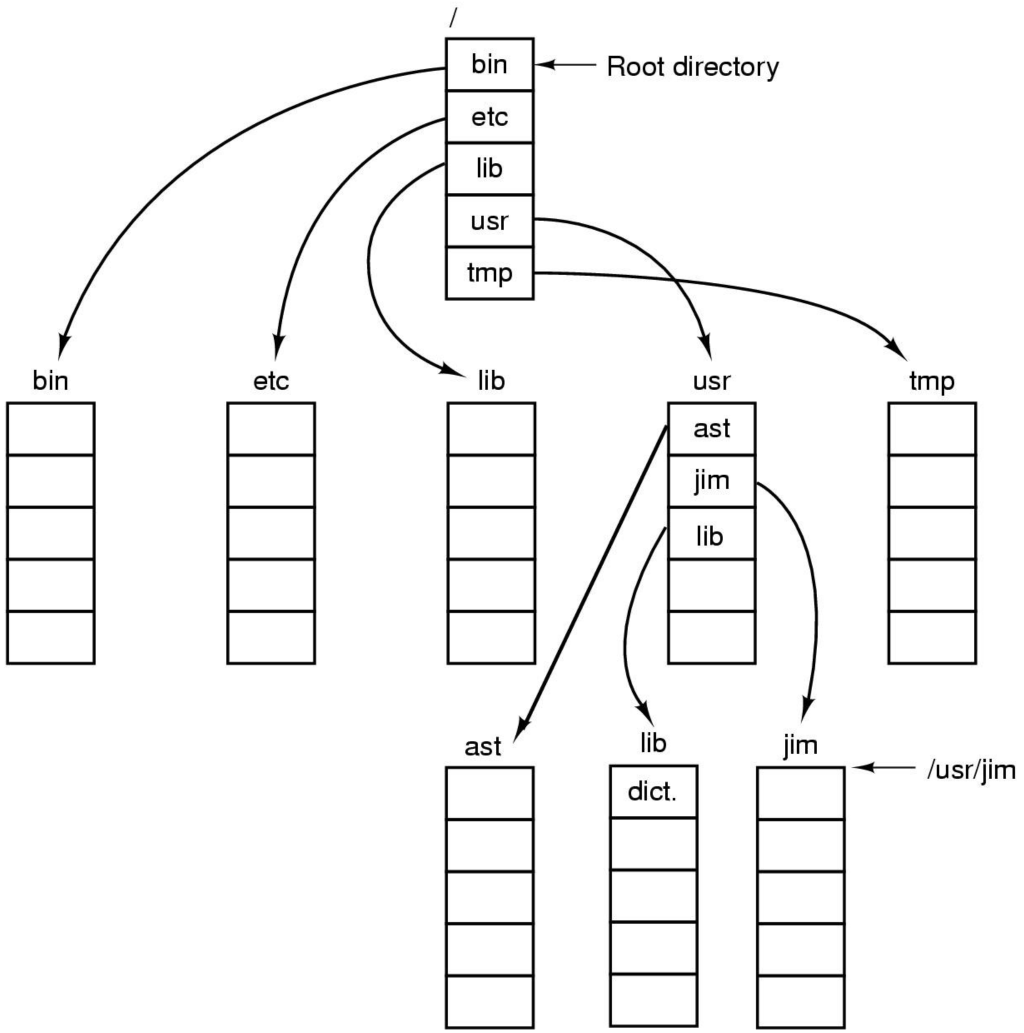
Letters indicate *owners* of the directories and files

Hierarchical Directory Systems



A hierarchical directory system

Path Names

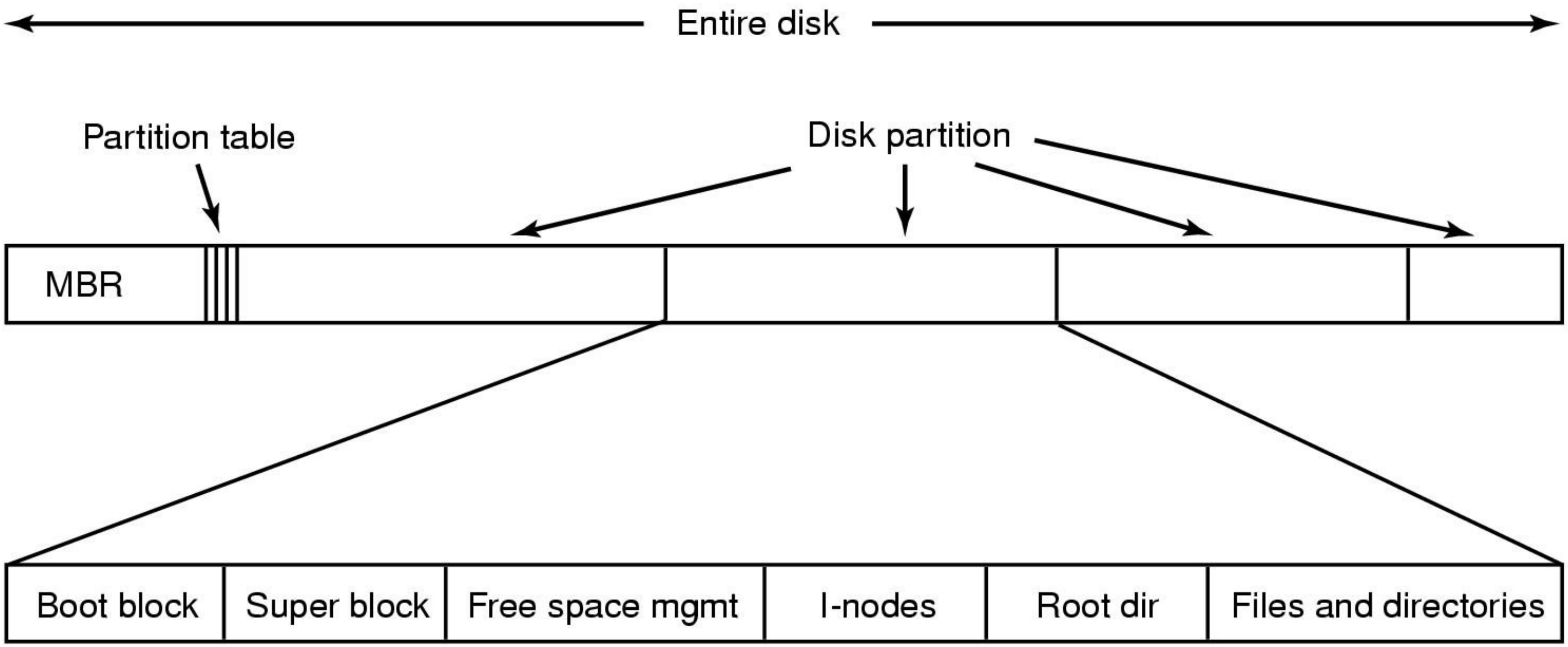


A UNIX directory tree

Directory Operations

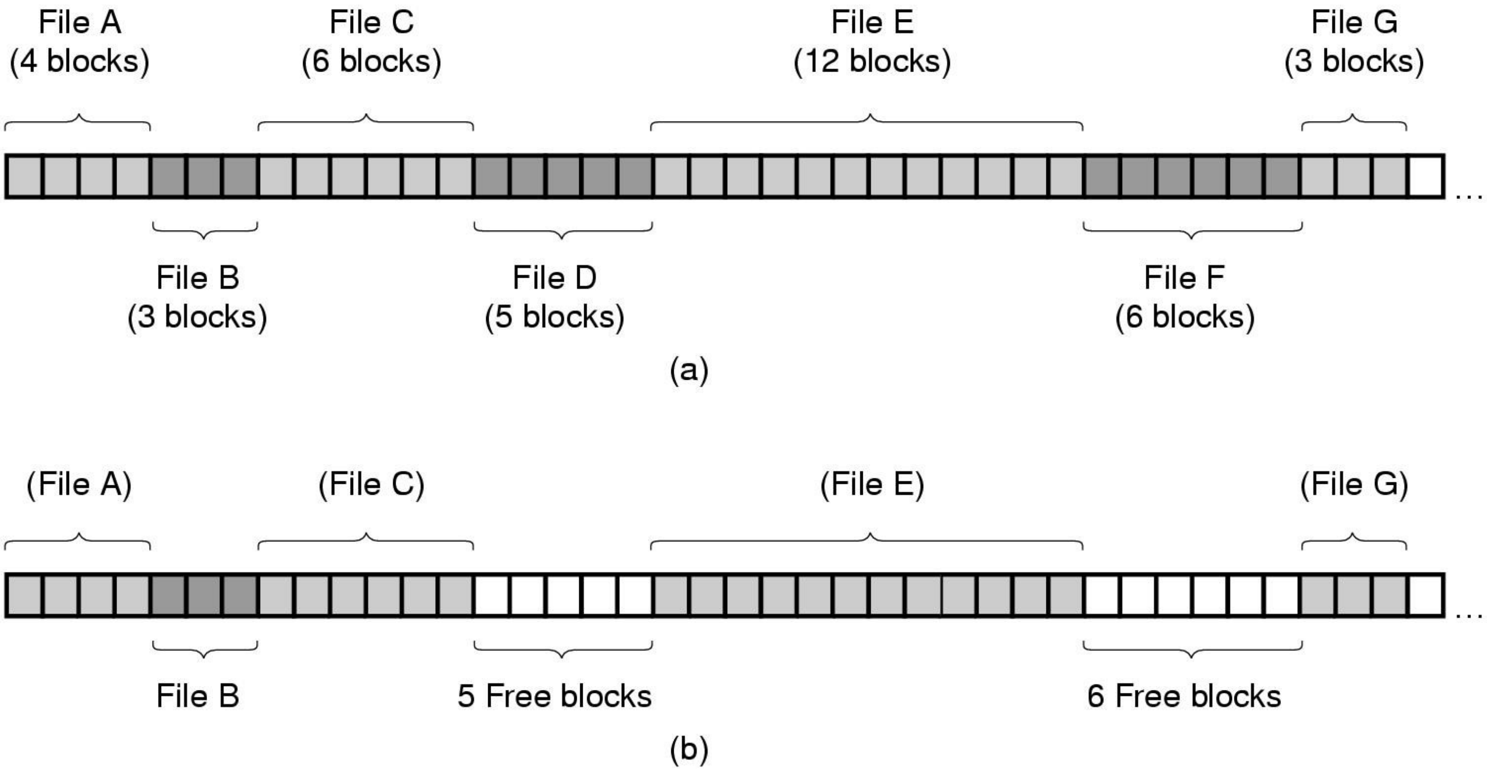
1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

File System Implementation



A possible file system layout

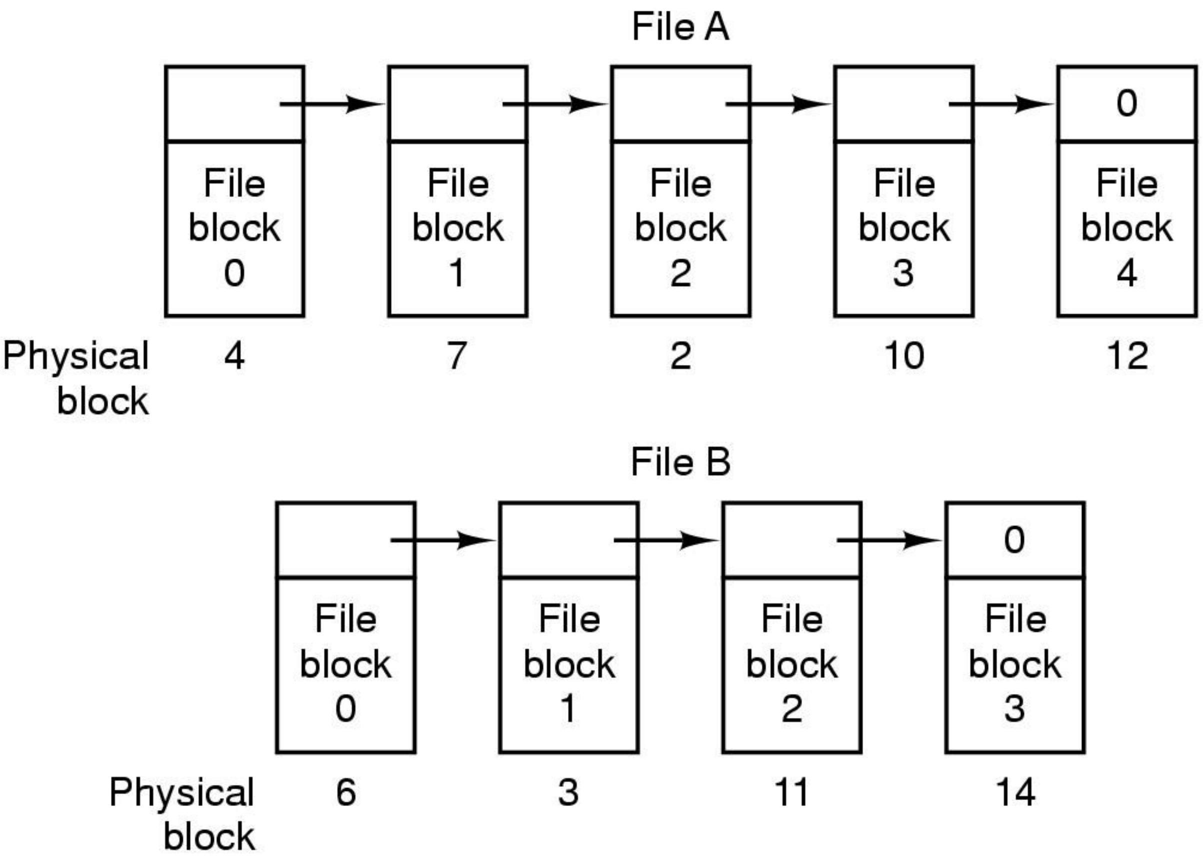
Implementing Files (1)



(a) Contiguous allocation of disk space for 7 files

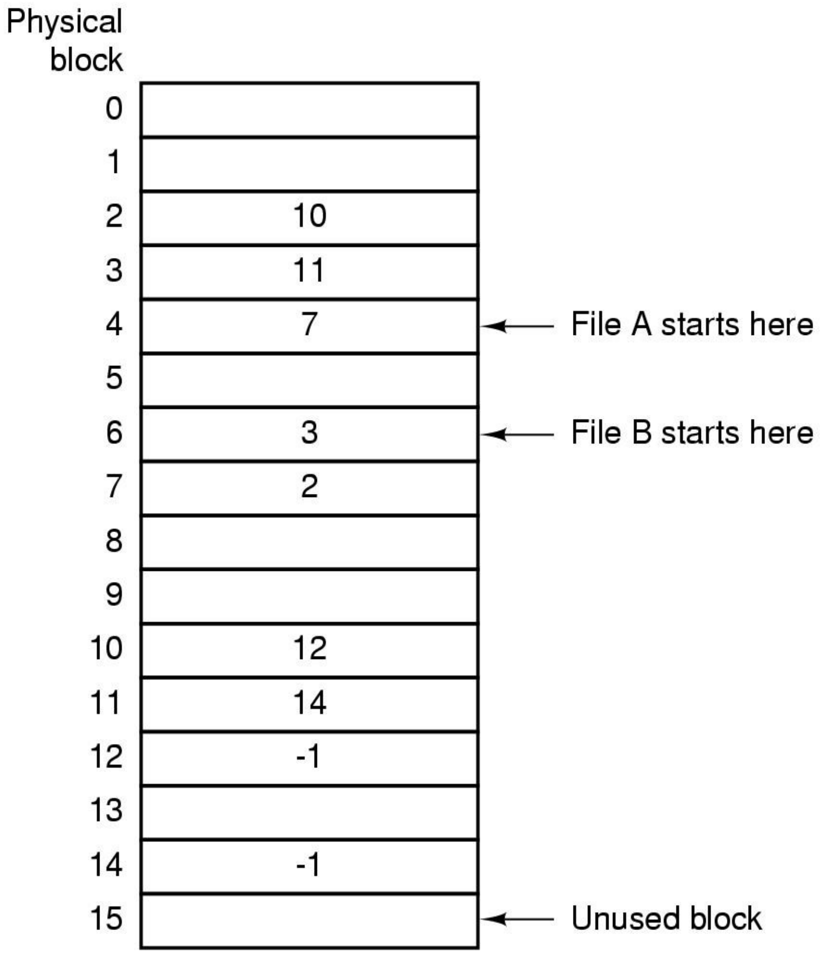
(b) State of the disk after files *D* and *E* have been removed

Implementing Files (2)



Storing a file as a linked list of disk blocks

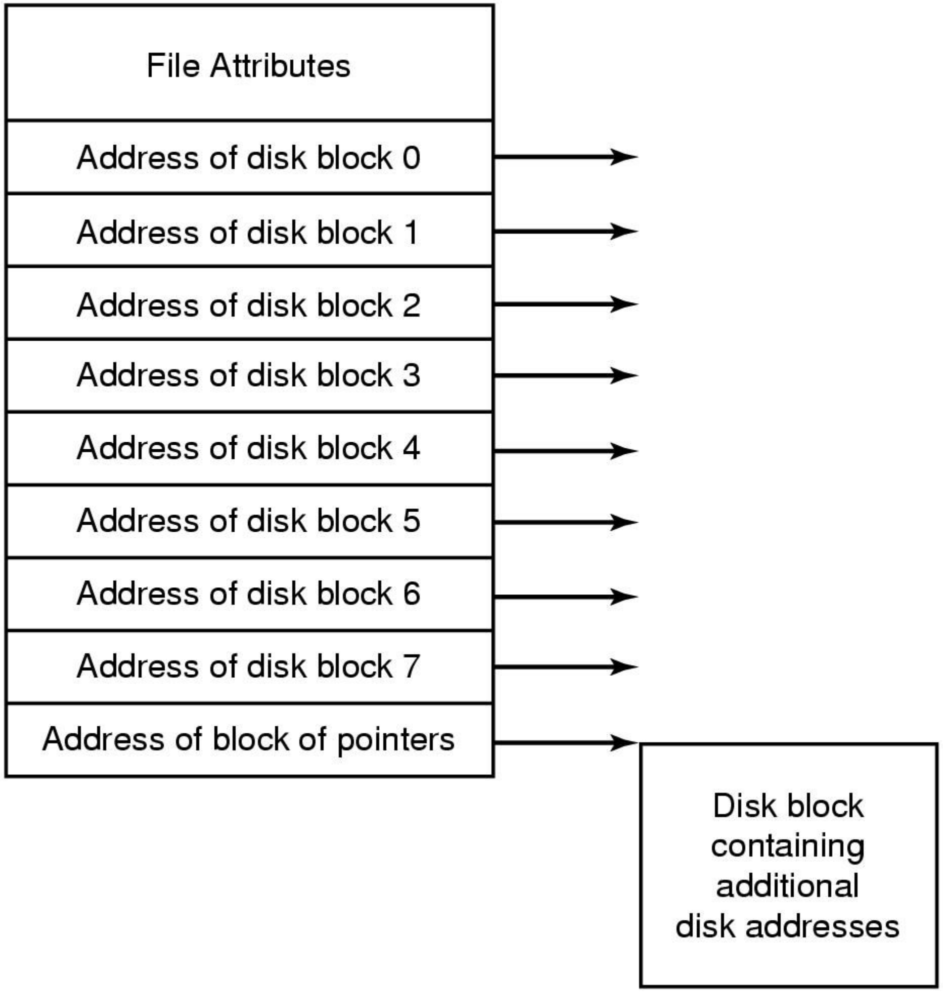
Implementing Files (3)



Physical block		
0		
1		
2	10	
3	11	
4	7	File A starts here
5		
6	3	File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		Unused block

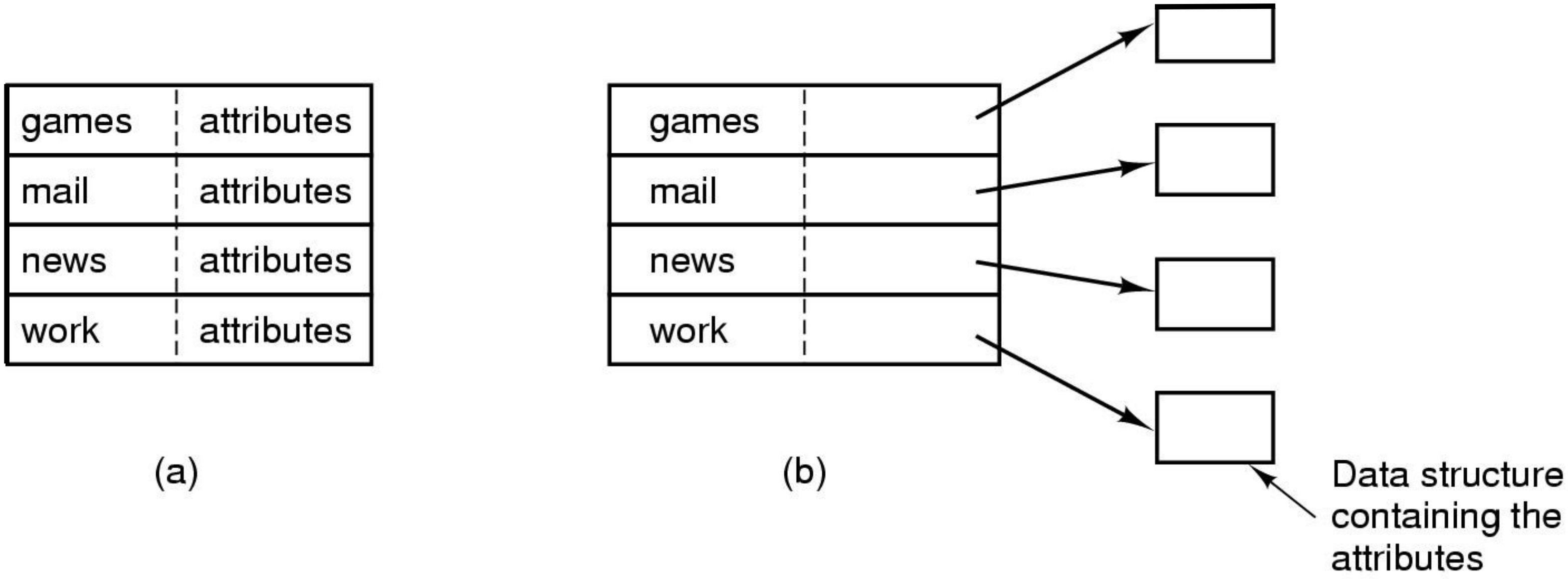
Linked list allocation using a File Allocation Table (FAT) in RAM

Implementing Files (4)



An example i-node

Implementing Directories (1)



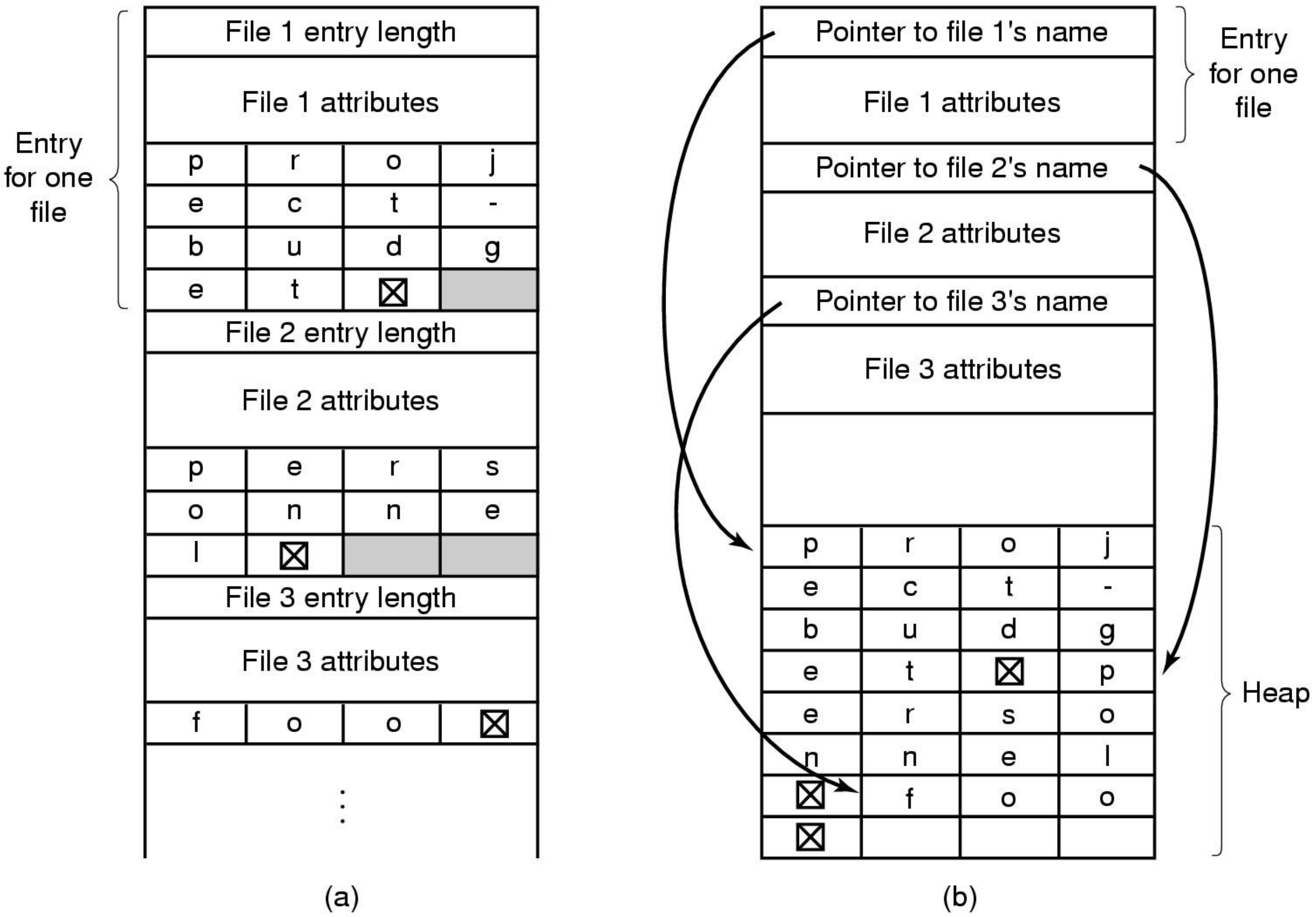
games	attributes
mail	attributes
news	attributes
work	attributes

(a) A simple directory

fixed size entries, disk addresses and attributes in directory entry

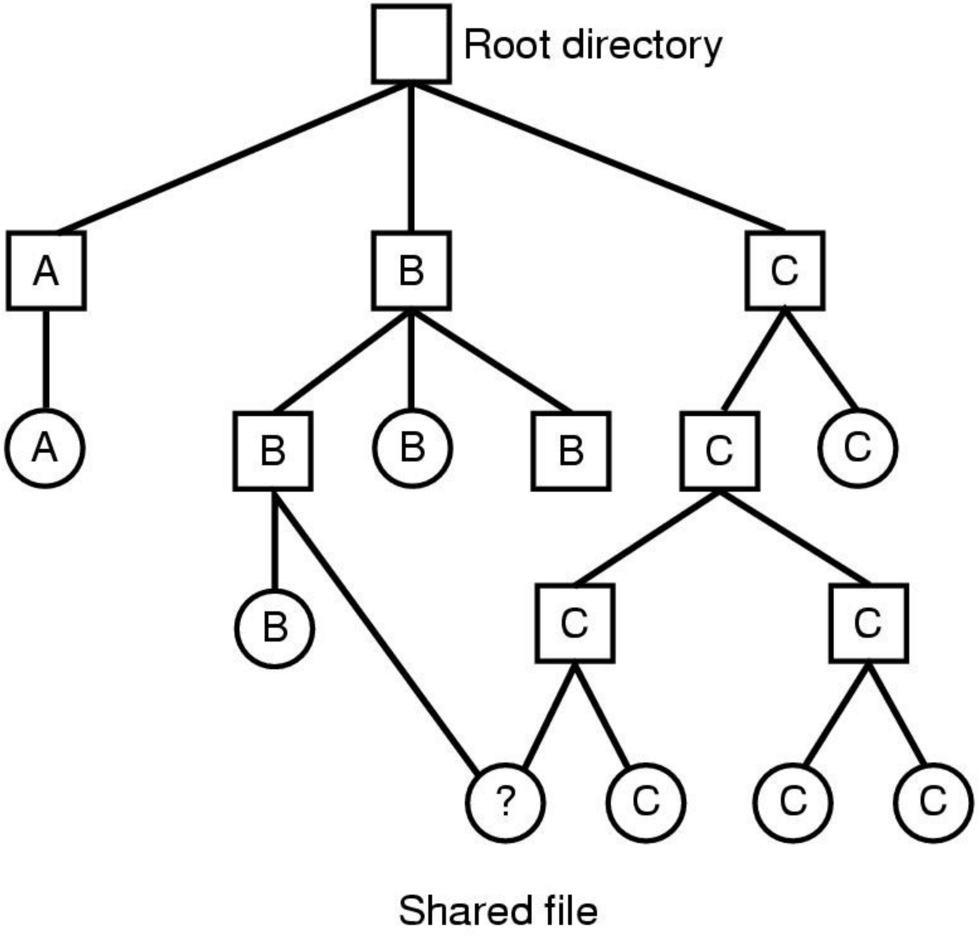
(b) Directory in which each entry just refers to an i-node

Implementing Directories (2)



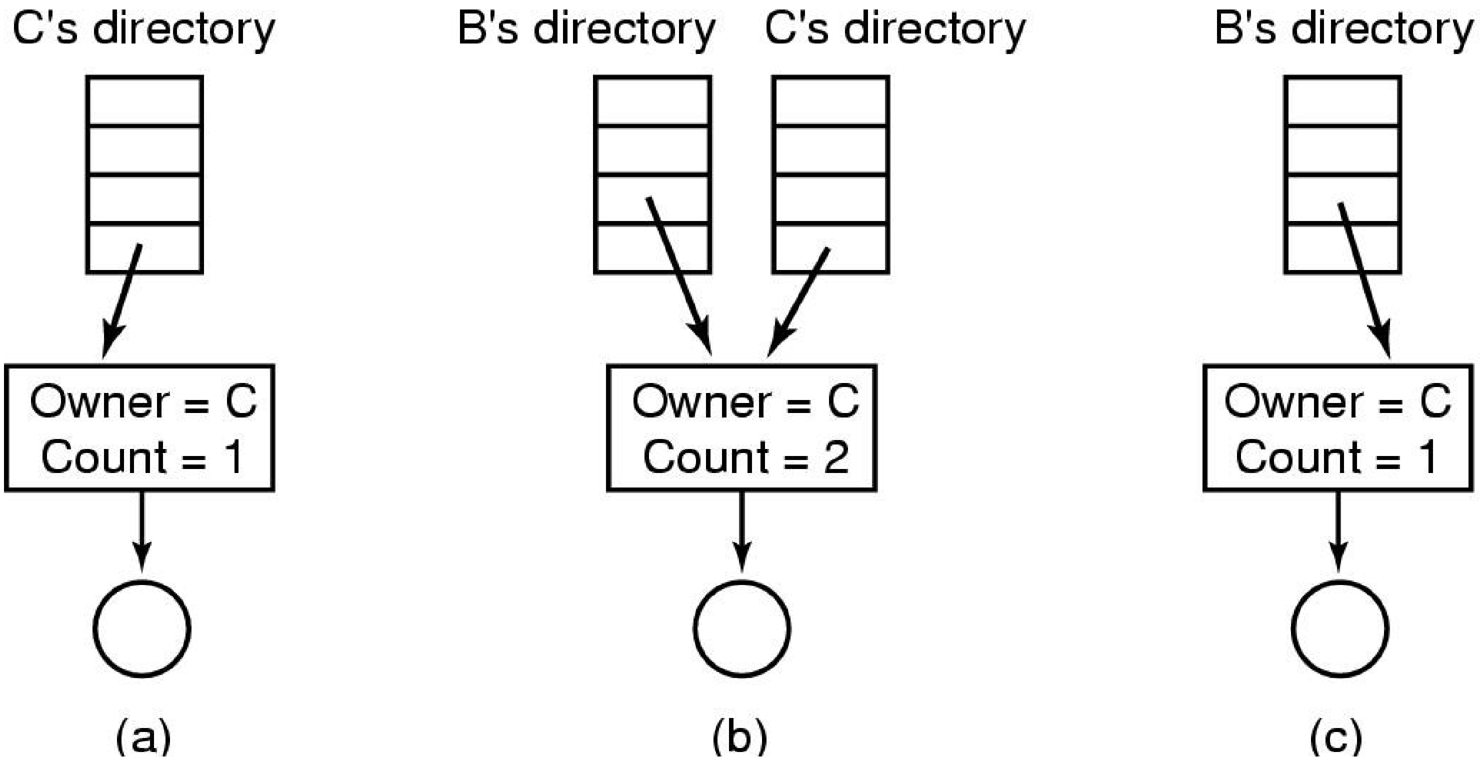
- Two ways of handling long file names in directory
  - (a) In-line
  - (b) In a heap

Shared Files (1)



File system containing a shared file

Shared Files (2)

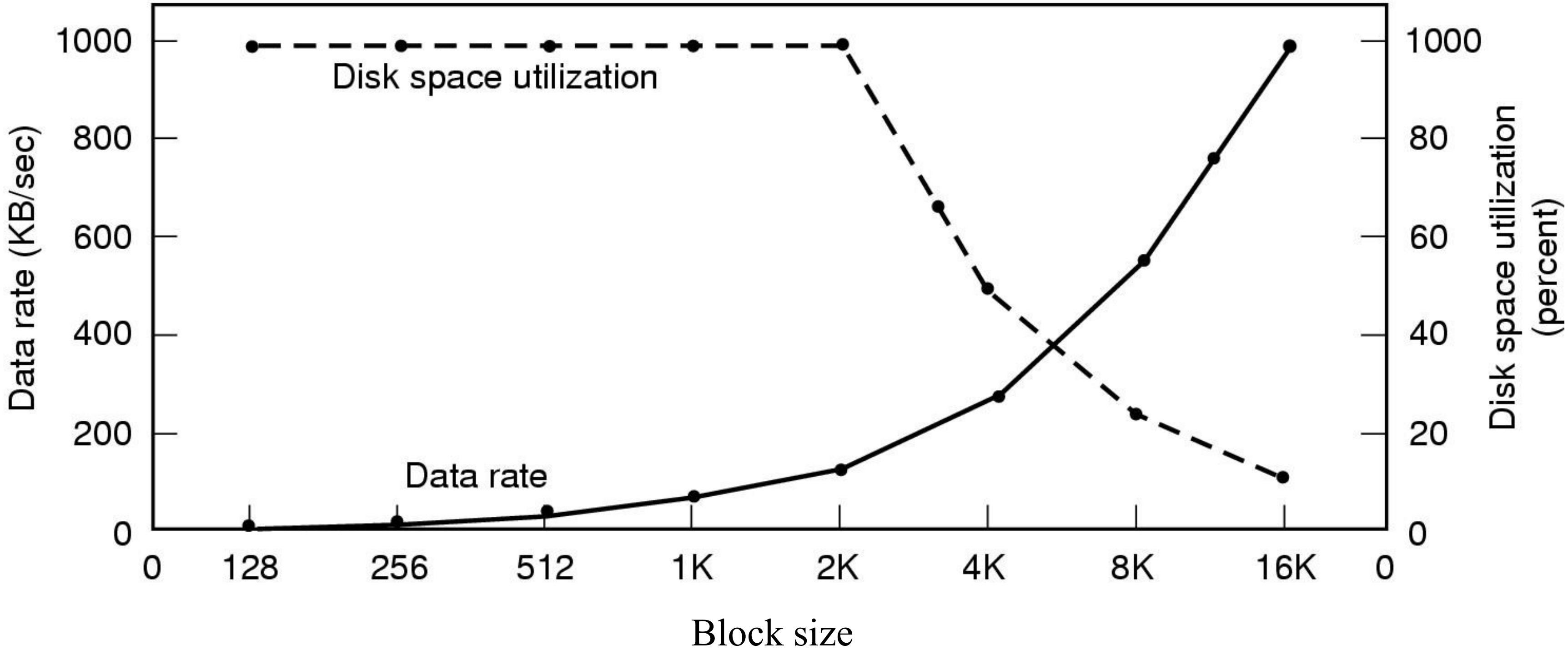


(a) Situation prior to linking

(b) After the link is created

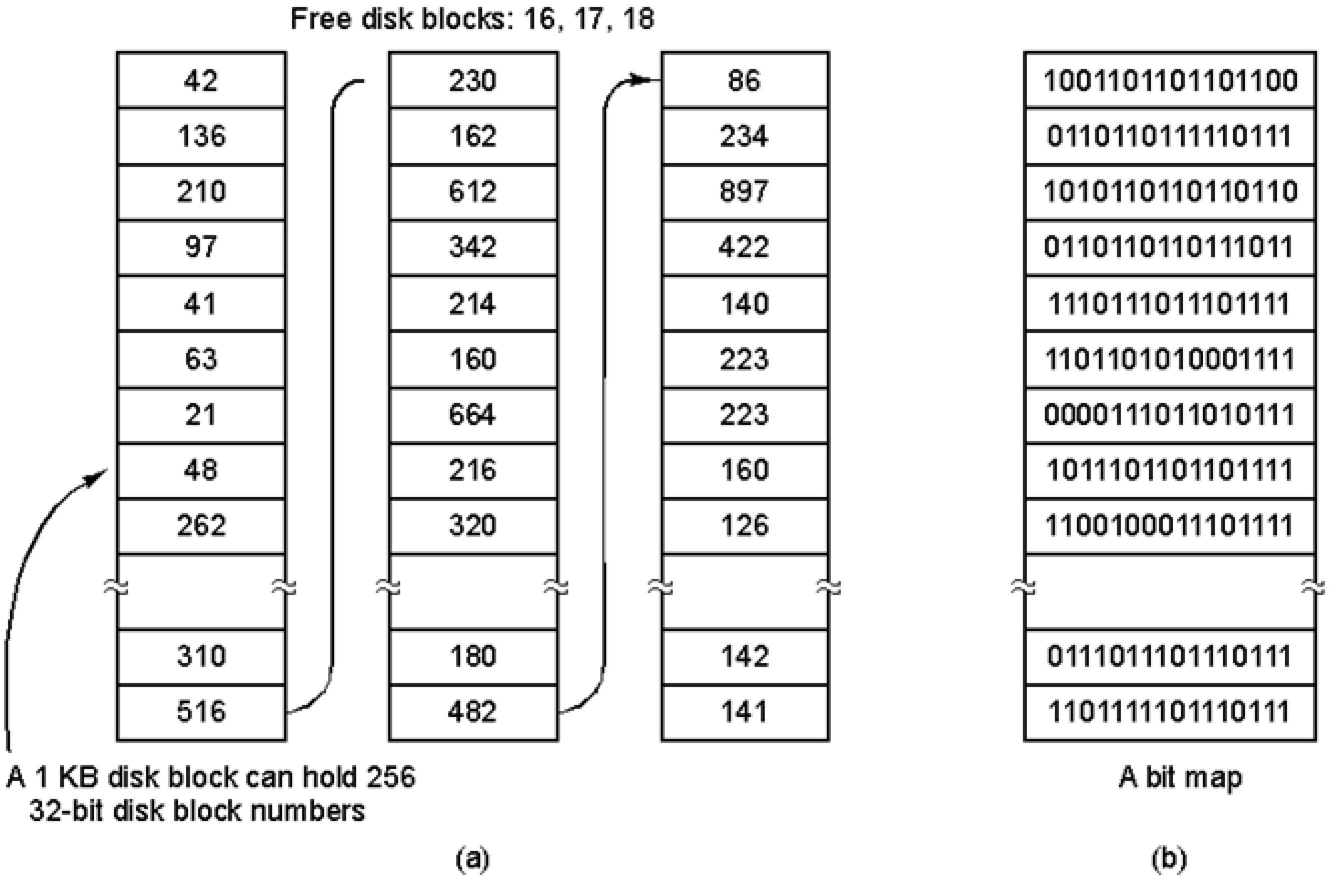
(c)After the original owner removes the file

Disk Space Management (1)



- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB

Disk Space Management (2)



Free disk blocks: 16, 17, 18

42
136
210
97
41
63
21
48
262
310
516

230
162
612
342
214
160
664
216
320
180
482

86
234
897
422
140
223
223
160
126
142
141

A 1 KB disk block can hold 256
32-bit disk block numbers

(a)

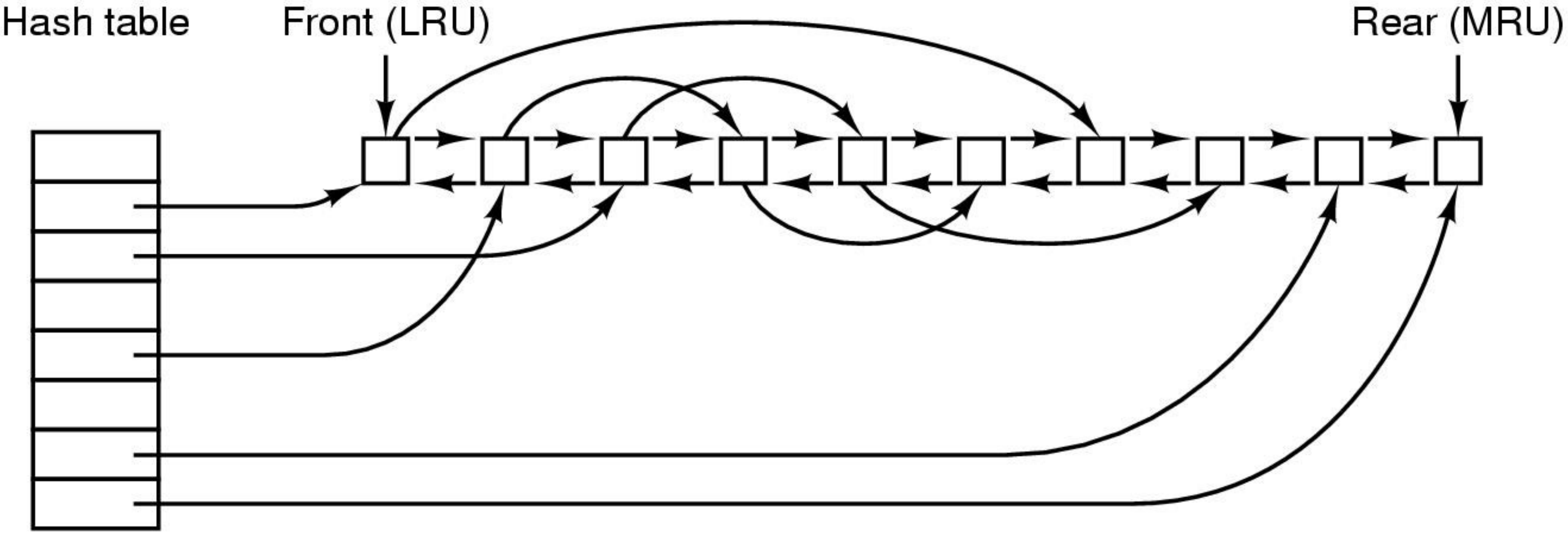
1001101101101100
0110110111110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
0111011101110111
1101111101110111

A bit map

(b)

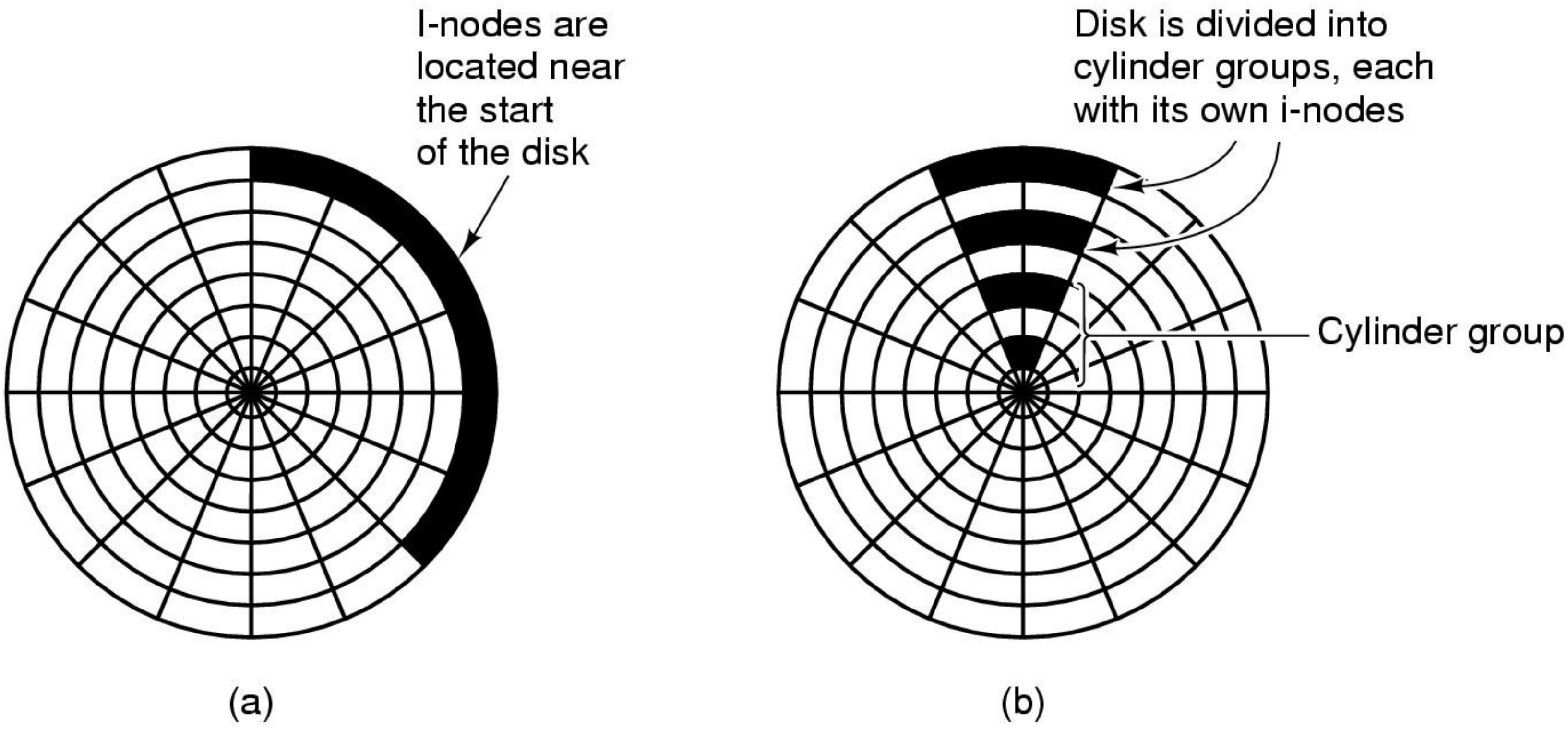
(a) Storing the free list on a linked list
(b) A bit map

File System Performance (1)



The block cache data structures

File System Performance (2)



I-nodes are located near the start of the disk

Disk is divided into cylinder groups, each with its own i-nodes

Cylinder group

(a)

(b)

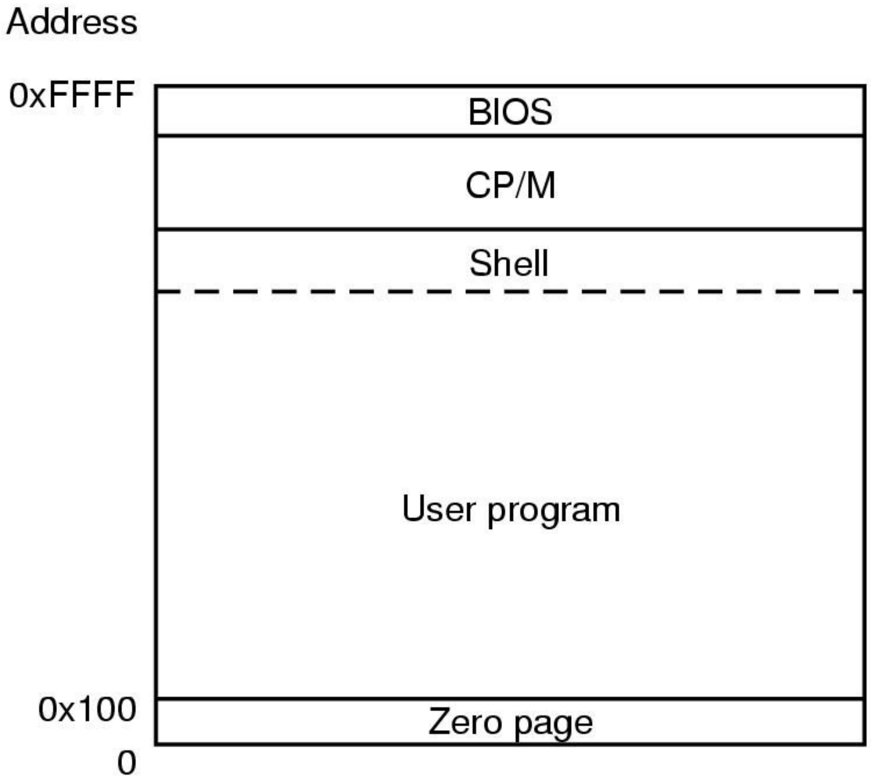
- I-nodes placed at the start of the disk
- Disk divided into cylinder groups
  - each with its own blocks and i-nodes

Log-Structured File Systems

- With CPUs faster, memory larger
  - disk caches can also be larger
  - increasing number of read requests can come from cache
  - thus, most disk accesses will be writes

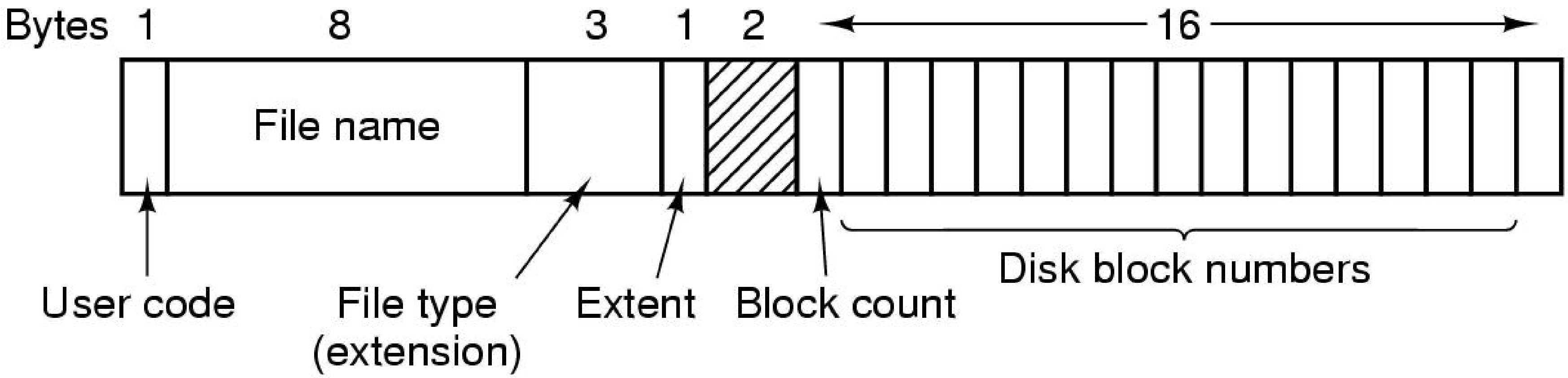
- LFS Strategy structures entire disk as a log
  - have all writes initially buffered in memory
  - periodically write these to the end of the disk log
  - when file opened, locate i-node, then find blocks

The CP/M File System (1)



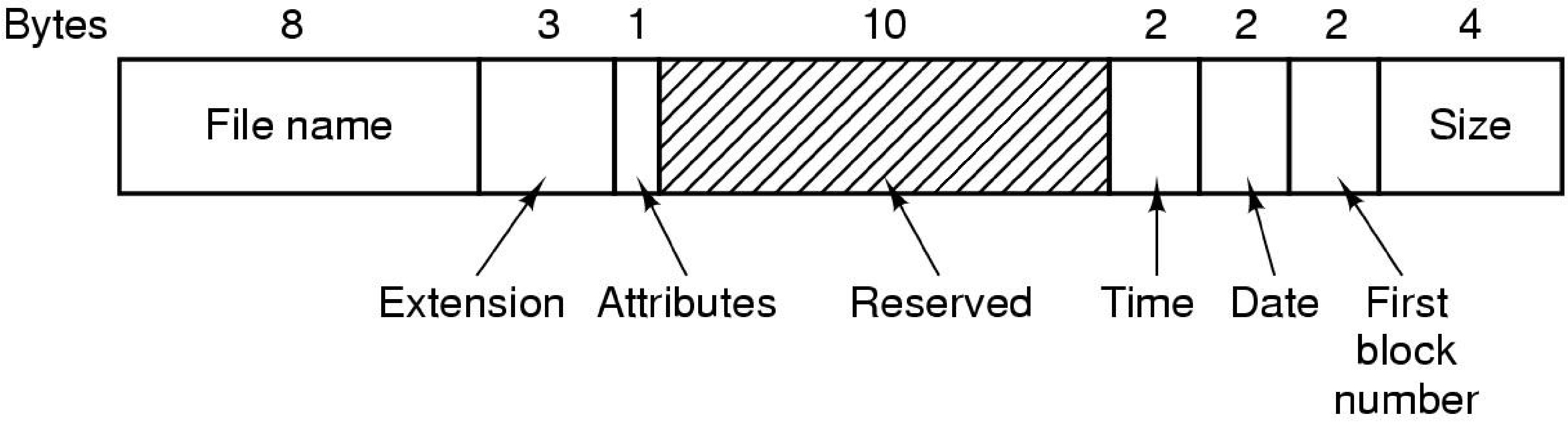
Memory layout of CP/M

The CP/M File System (2)



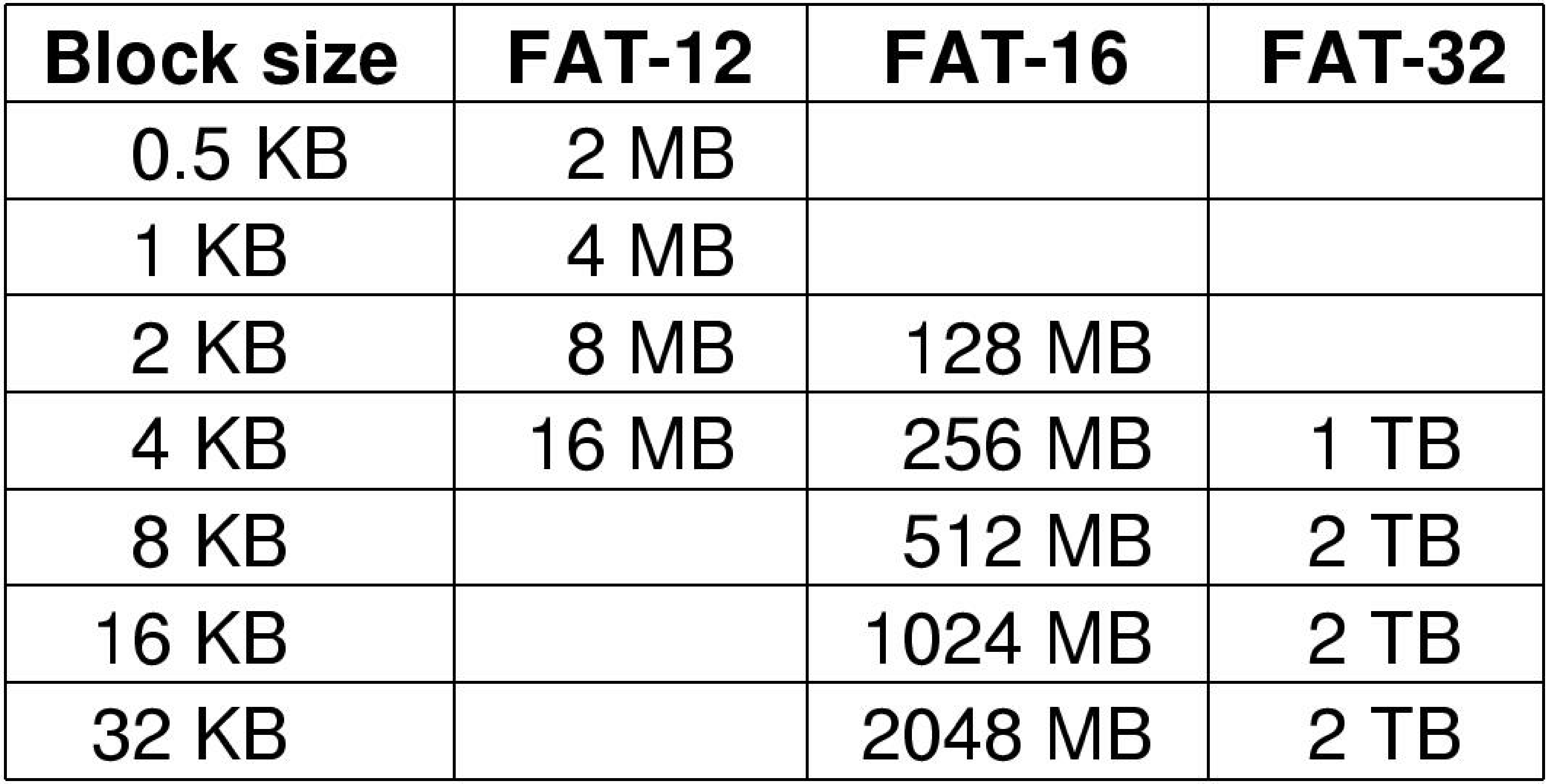
The CP/M directory entry format

The MS-DOS File System (1)



The MS-DOS directory entry

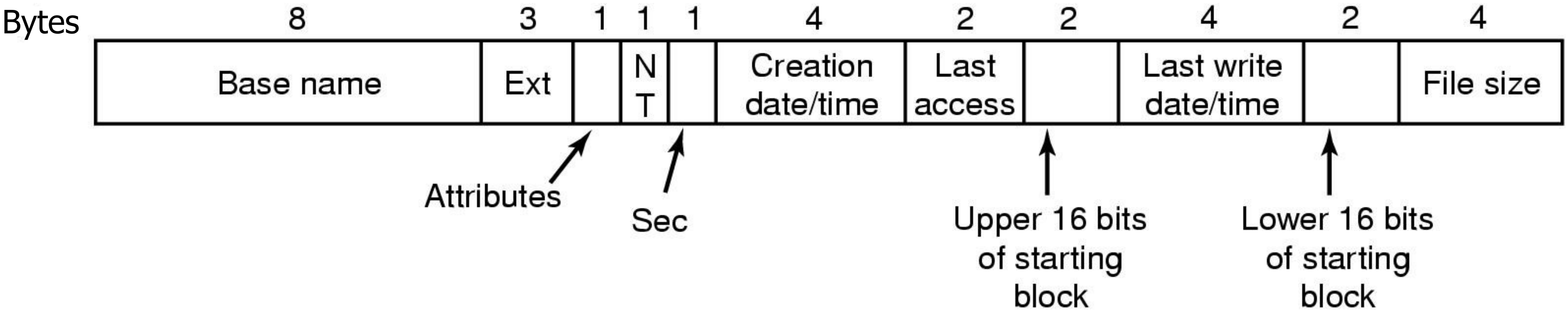
The MS-DOS File System (2)



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

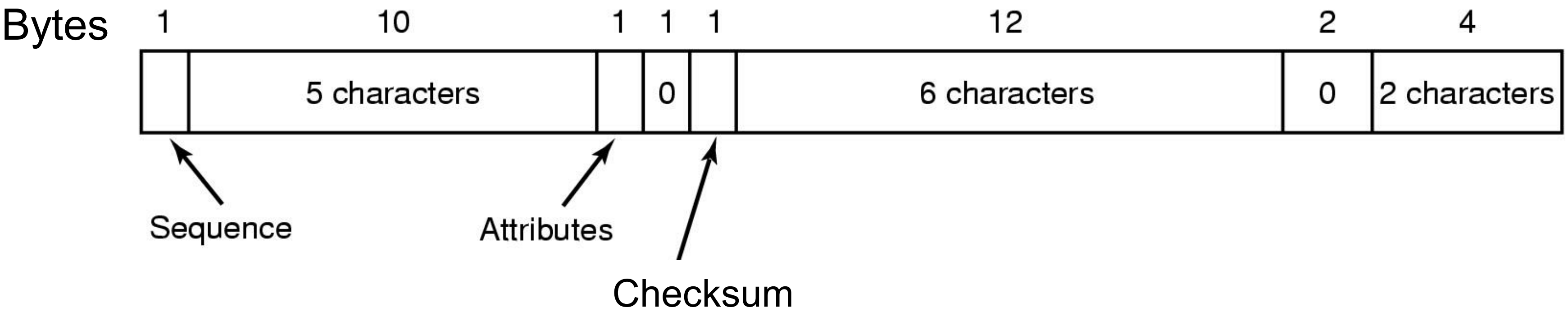
- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

The Windows 98 File System (1)



The extended MOS-DOS directory entry used in Windows 98

The Windows 98 File System (2)



Bytes	1	10	1	1	1	12	2	4
		5 characters		0		6 characters	0	2 characters

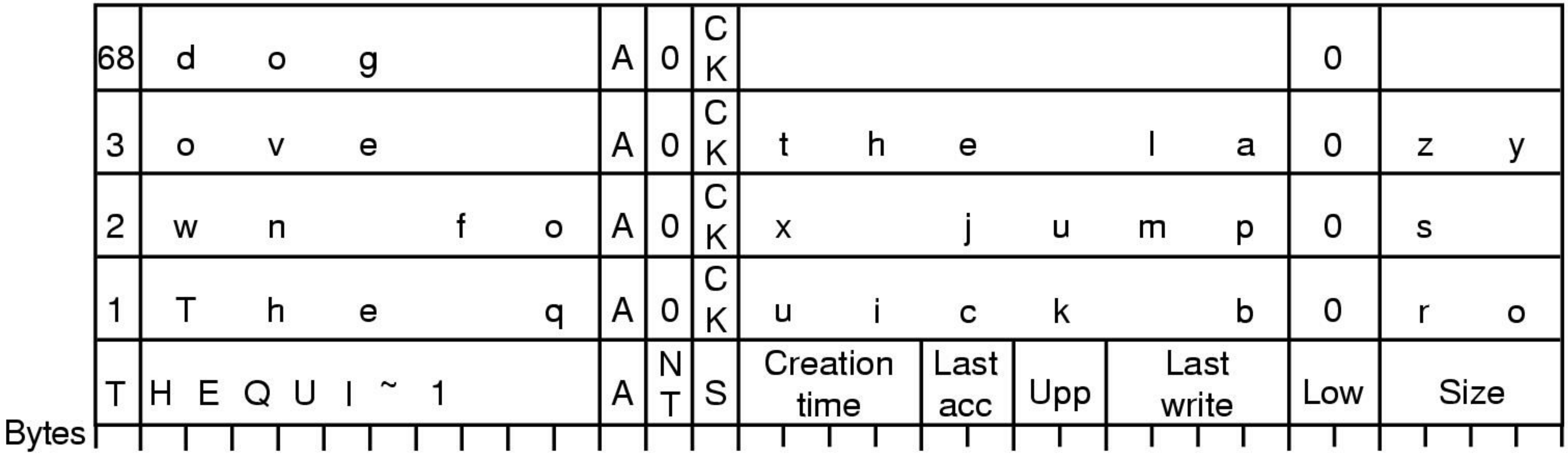
Sequence

Attributes

Checksum

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

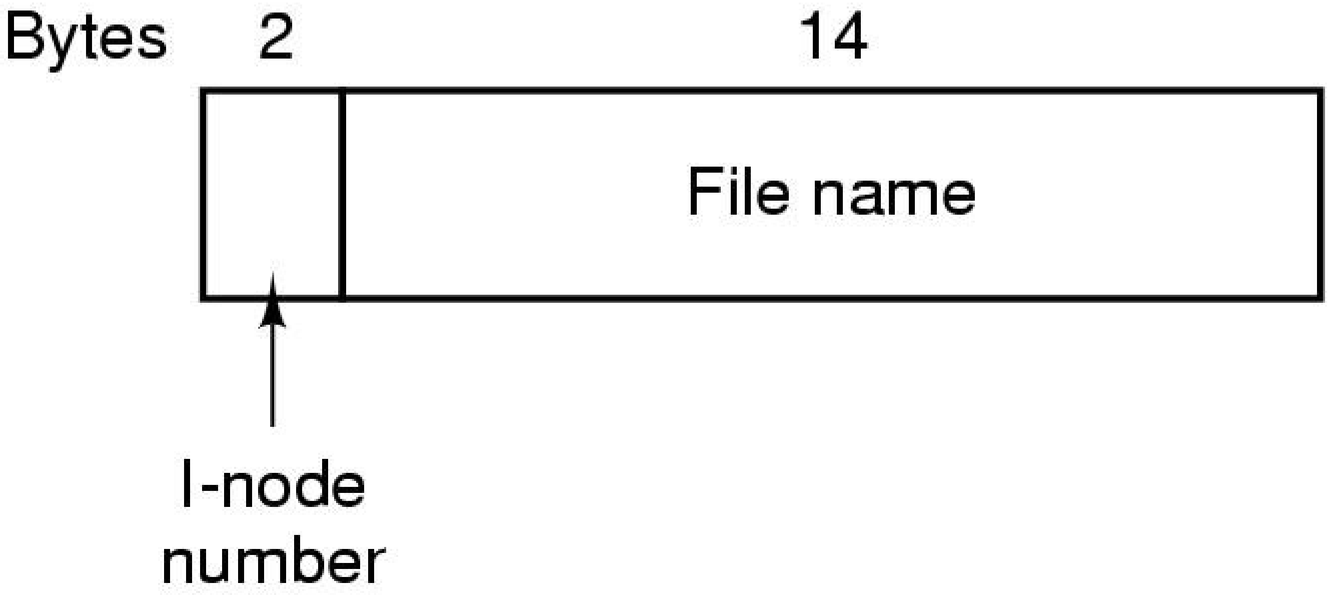
The Windows 98 File System (3)



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

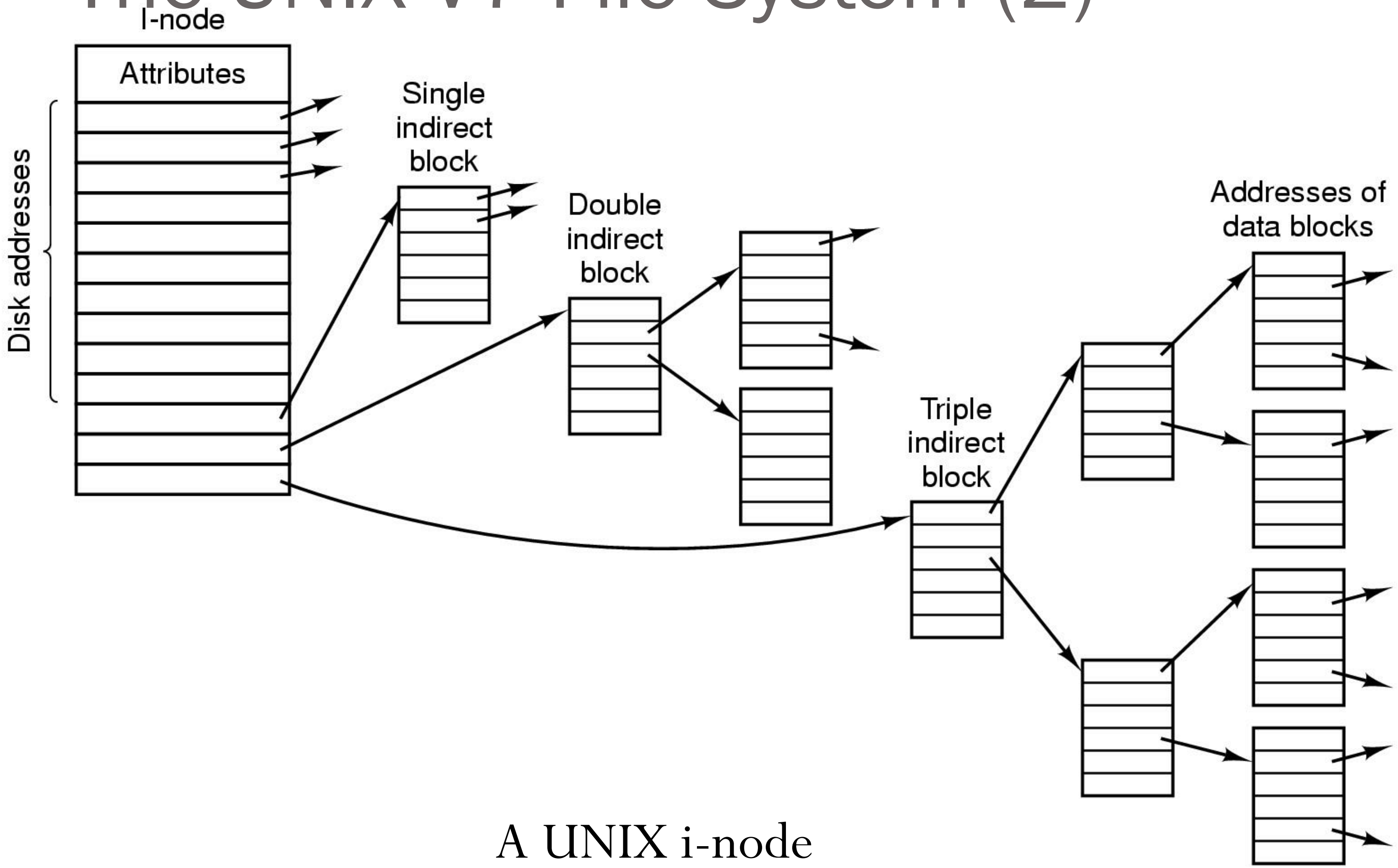
An example of how a long name is stored in Windows 98

The UNIX V7 File System (1)



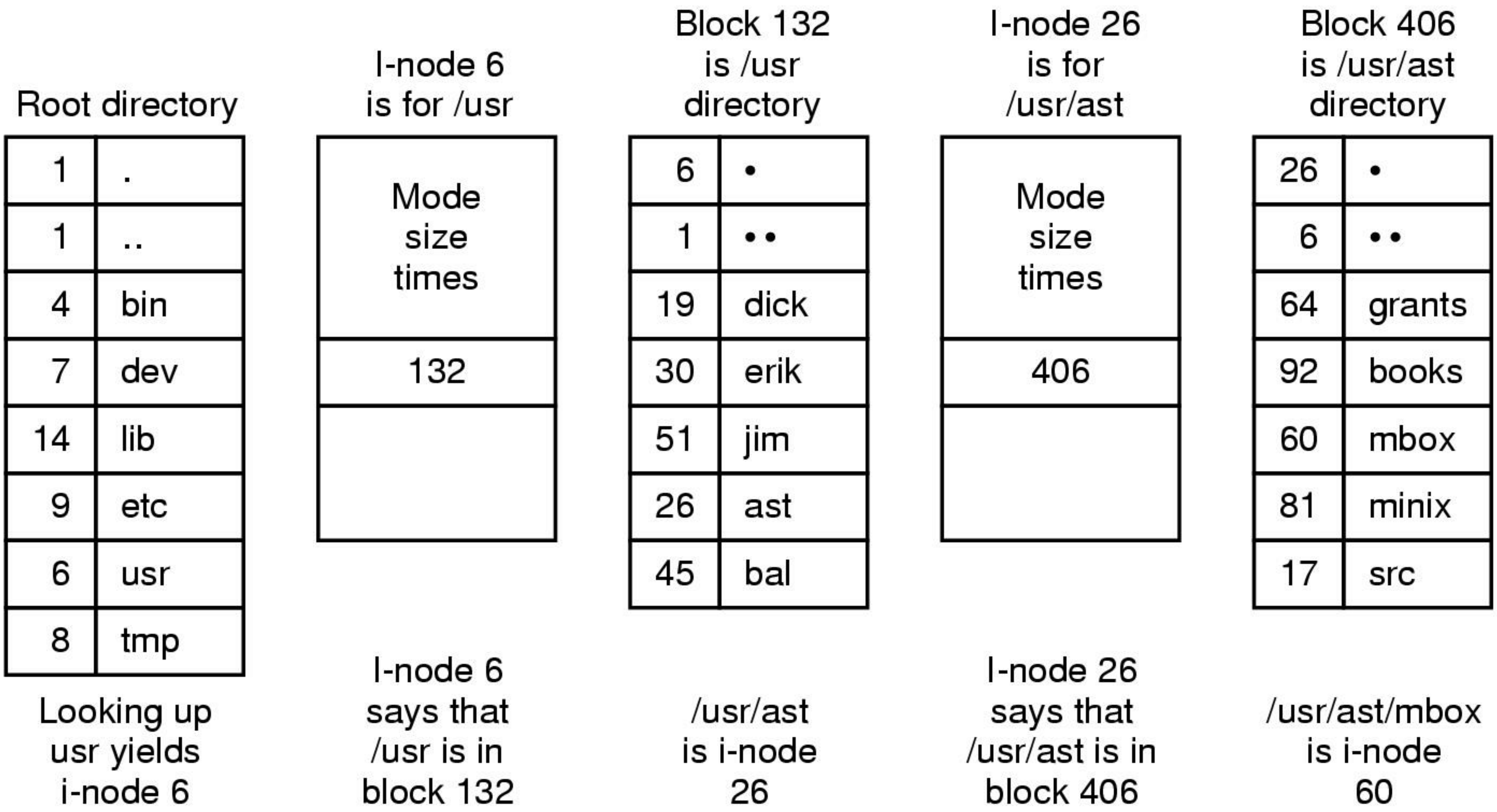
A UNIX V7 directory entry

The UNIX V7 File System (2)



A UNIX i-node

The UNIX V7 File System (3)



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

The steps in looking up */usr/ast/mbox*

Questions?