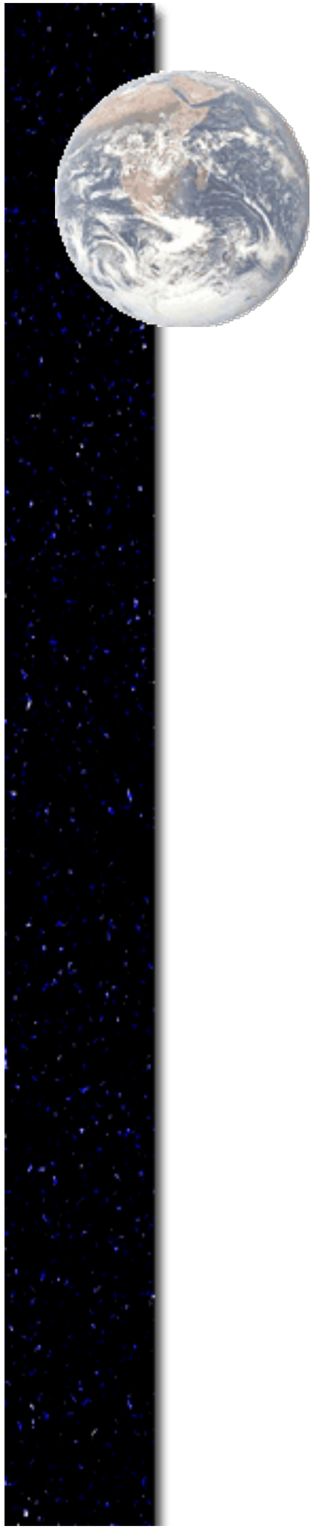
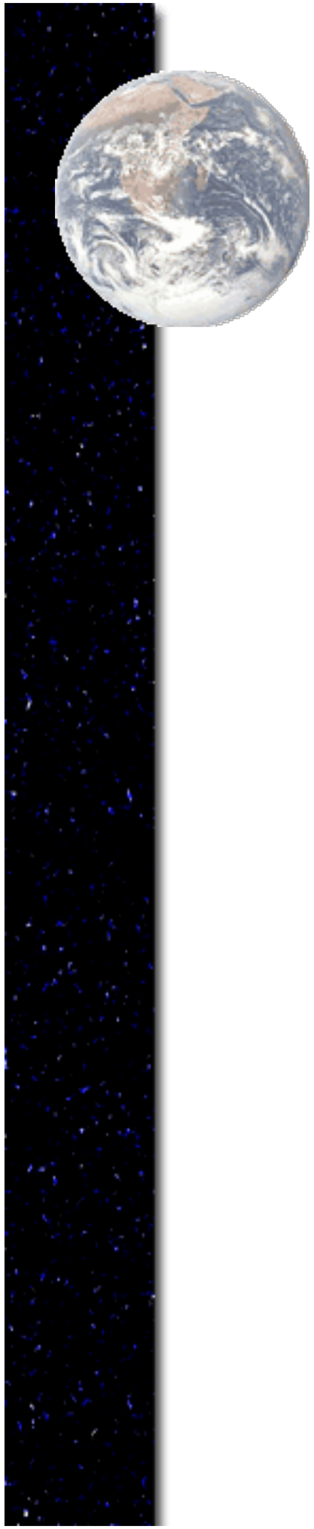
File Systems



File Systems

Essential requirements for long-term information storage:

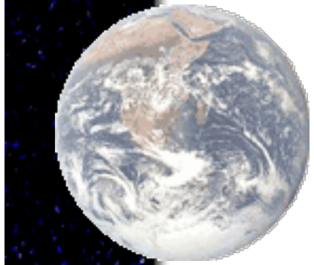
- It must be possible to store a very large amount of information.
- The information must survive the termination of the process using it.
- Multiple processes must be able to access the information concurrently.



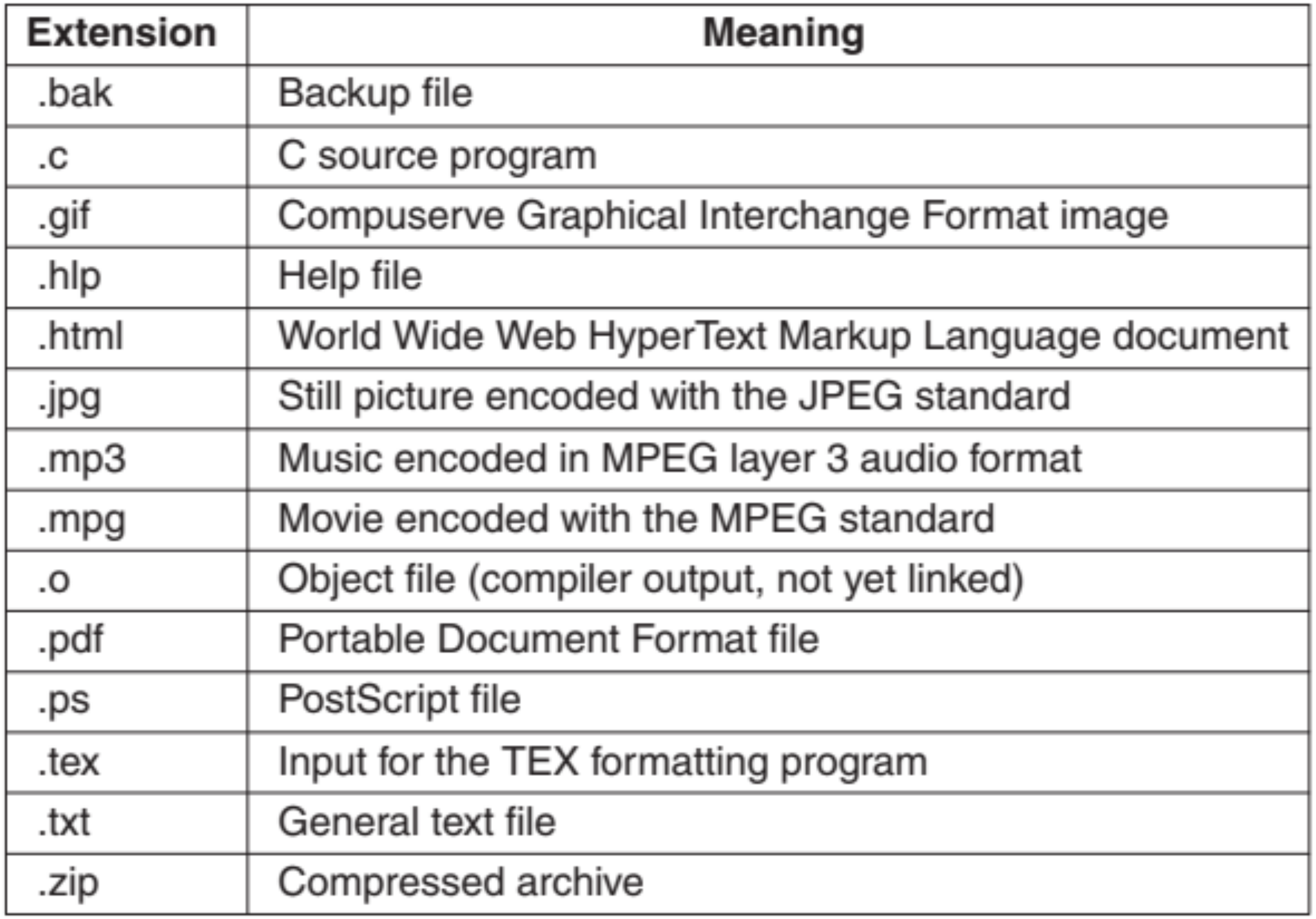
File Systems

Think of a disk as a linear sequence of fixed-size blocks and supporting reading and writing of blocks. Questions that quickly arise:

- How do you find information?
- How do you keep one user from reading another's data?
- How do you know which blocks are free?

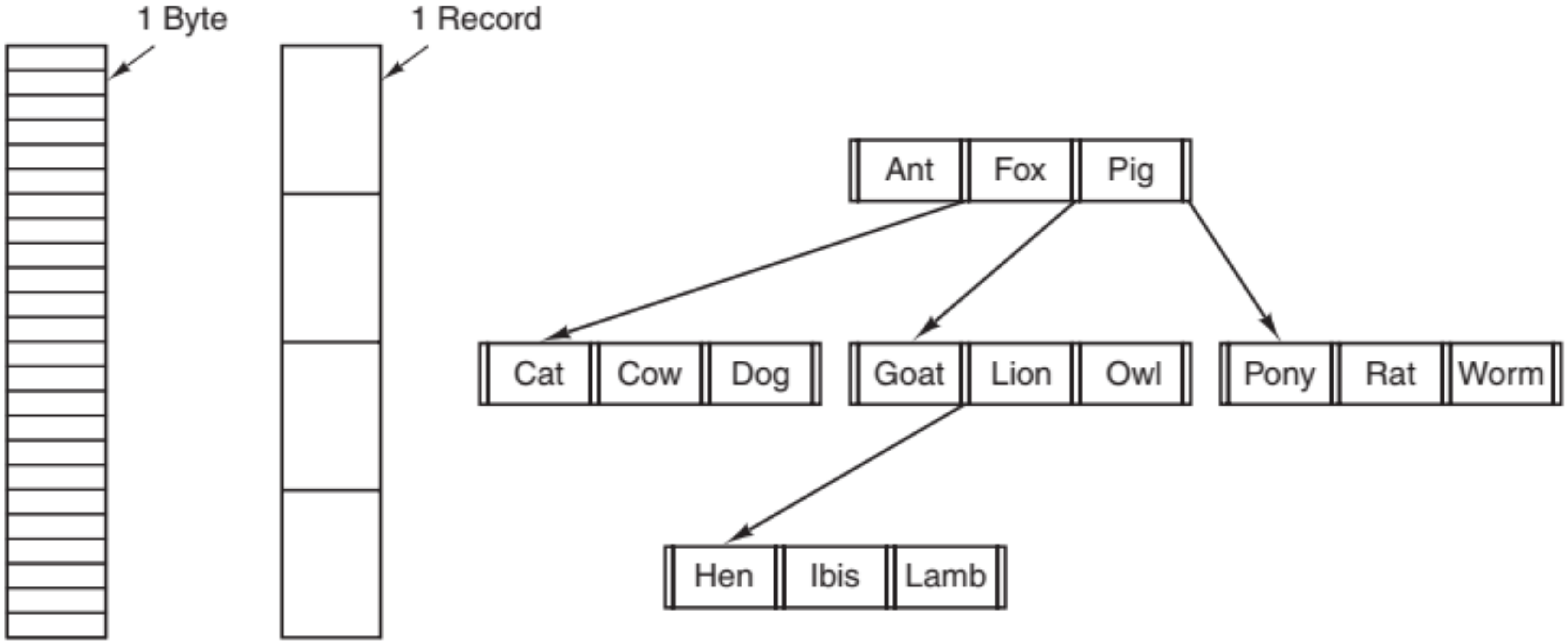


Typical File Extensions



Extension	Meaning
.bak	Backup file
.c	C source program
.gif	Compuserve Graphical Interchange Format image
.hlp	Help file
.html	World Wide Web HyperText Markup Language document
.jpg	Still picture encoded with the JPEG standard
.mp3	Music encoded in MPEG layer 3 audio format
.mpg	Movie encoded with the MPEG standard
.o	Object file (compiler output, not yet linked)
.pdf	Portable Document Format file
.ps	PostScript file
.tex	Input for the TEX formatting program
.txt	General text file
.zip	Compressed archive

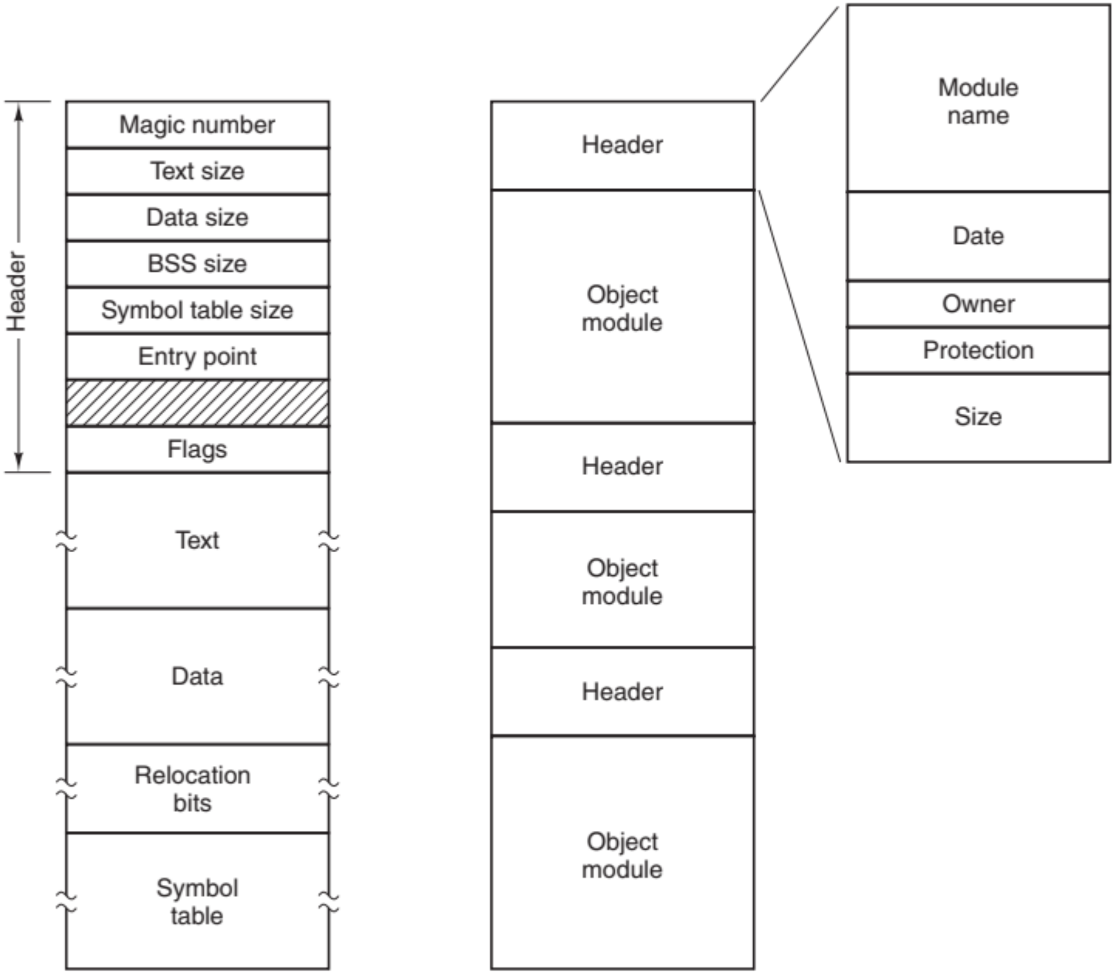
File Structure



(a) (b) (c)

Byte sequence Record sequence Tree

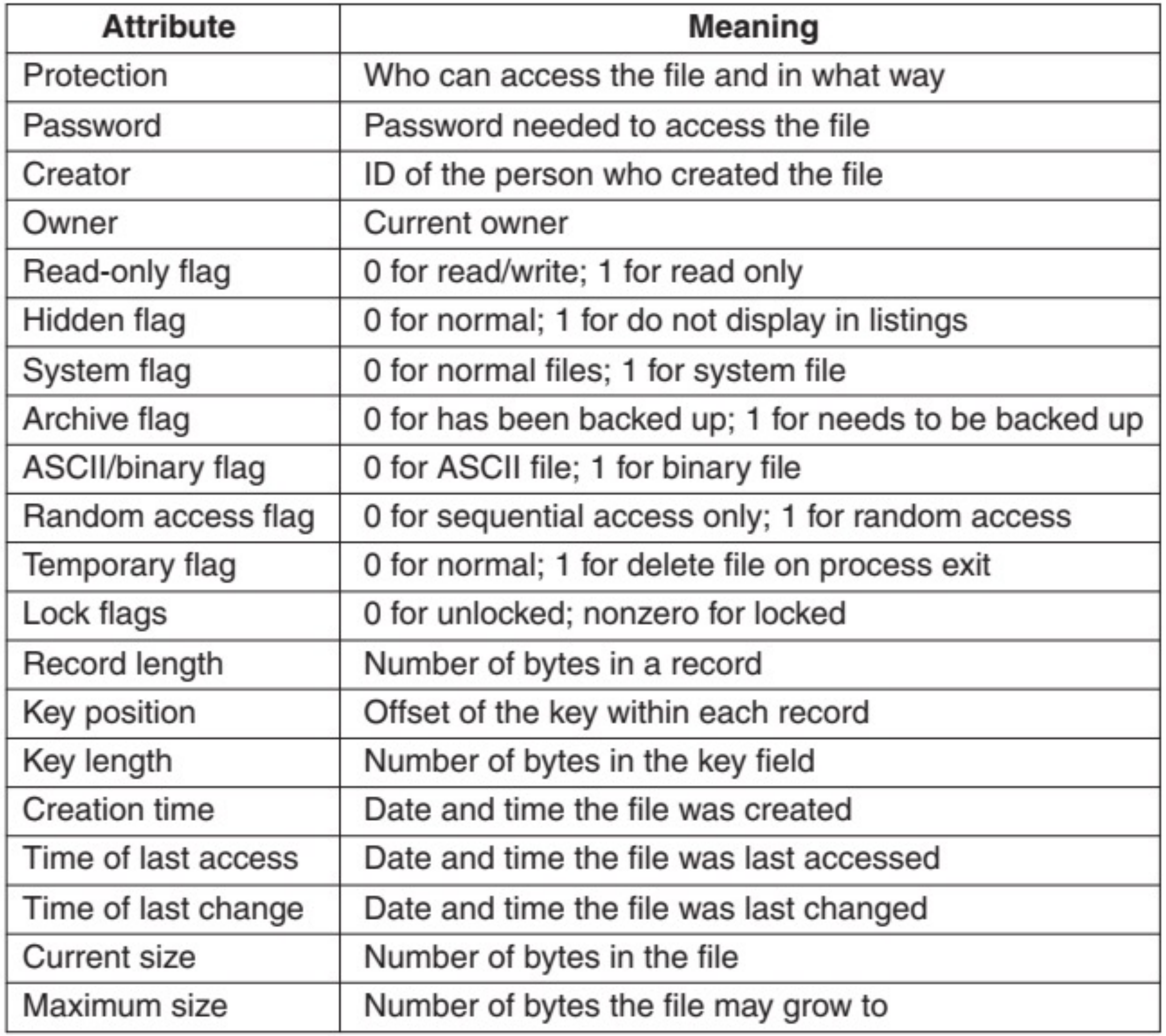
File Types



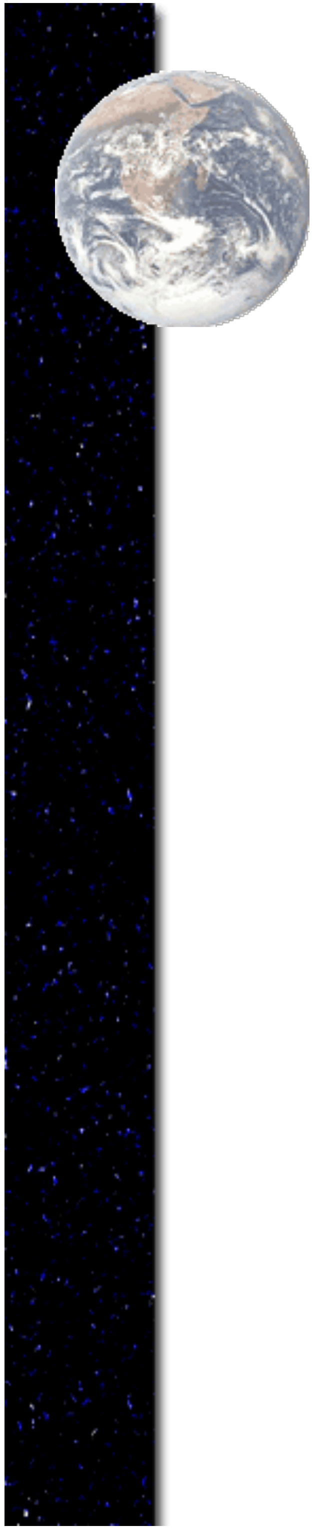
(a) An executable file

(b) An archive

Some Possible File Attributes



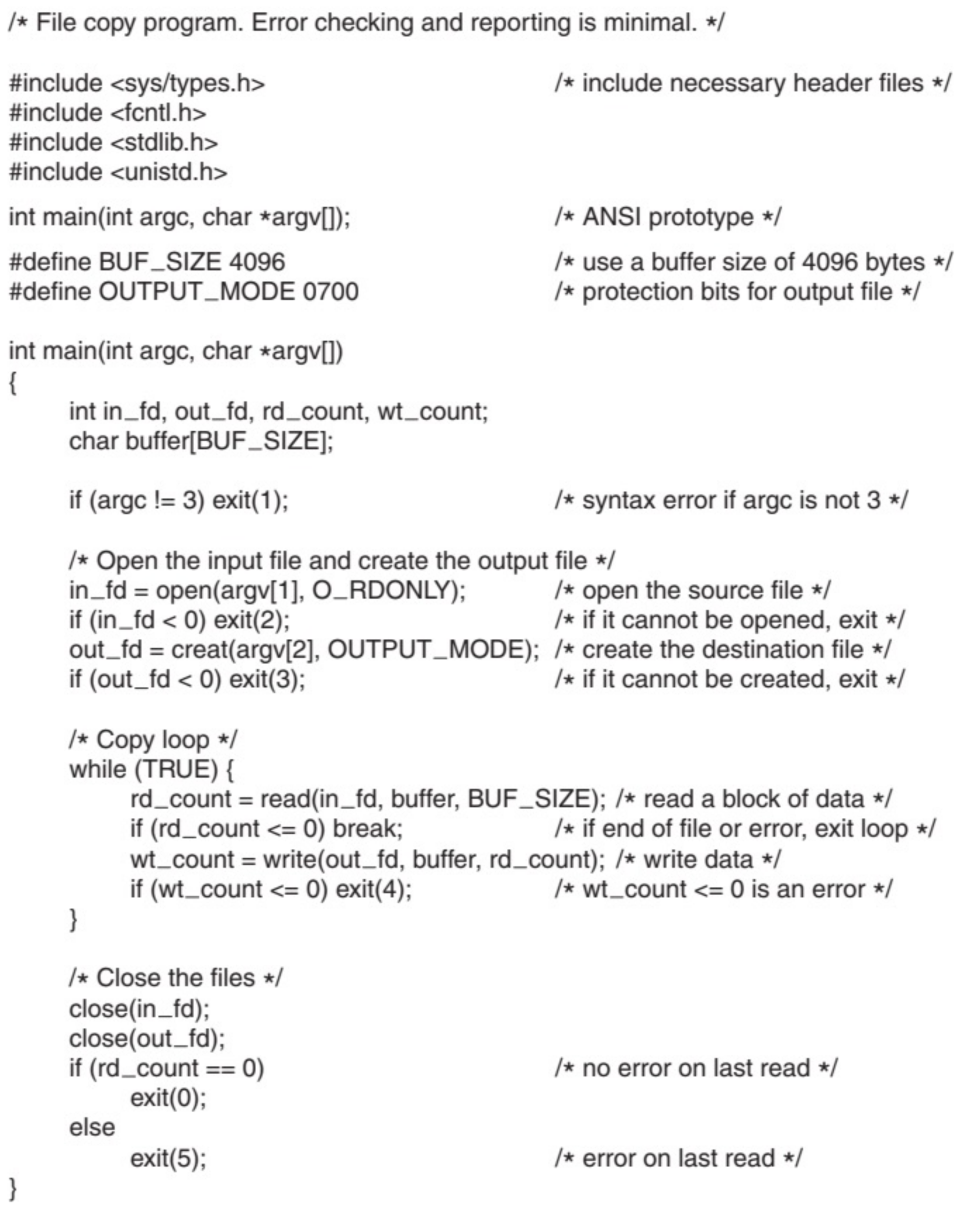
Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 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 was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to



File Operations

- The most common system calls relating to files:
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write
  - Append
  - Seek
  - Get Attributes
  - Set Attributes
  - Rename

Example Program Using File System Calls

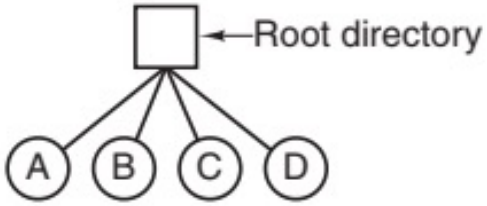


```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                      /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);           /* ANSI prototype */

#define BUF_SIZE 4096                       /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                    /* protection bits for output file */

int main(int argc, char *argv[])
{
     int in_fd, out_fd, rd_count, wt_count;
     char buffer[BUF_SIZE];

     if (argc != 3) exit(1);                /* syntax error if argc is not 3 */

     /* Open the input file and create the output file */
     in_fd = open(argv[1], O_RDONLY);       /* open the source file */
     if (in_fd < 0) exit(2);                /* if it cannot be opened, exit */
     out_fd = creat(argv[2], OUTPUT_MODE);  /* create the destination file */
     if (out_fd < 0) exit(3);               /* if it cannot be created, exit */

     /* Copy loop */
     while (TRUE) {
          rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
          if (rd_count <= 0) break;         /* if end of file or error, exit loop */
          wt_count = write(out_fd, buffer, rd_count); /* write data */
          if (wt_count <= 0) exit(4);       /* wt_count <= 0 is an error */
     }

     /* Close the files */
     close(in_fd);
     close(out_fd);
     if (rd_count == 0)                     /* no error on last read */
          exit(0);
     else
          exit(5);                          /* error on last read */
}
```

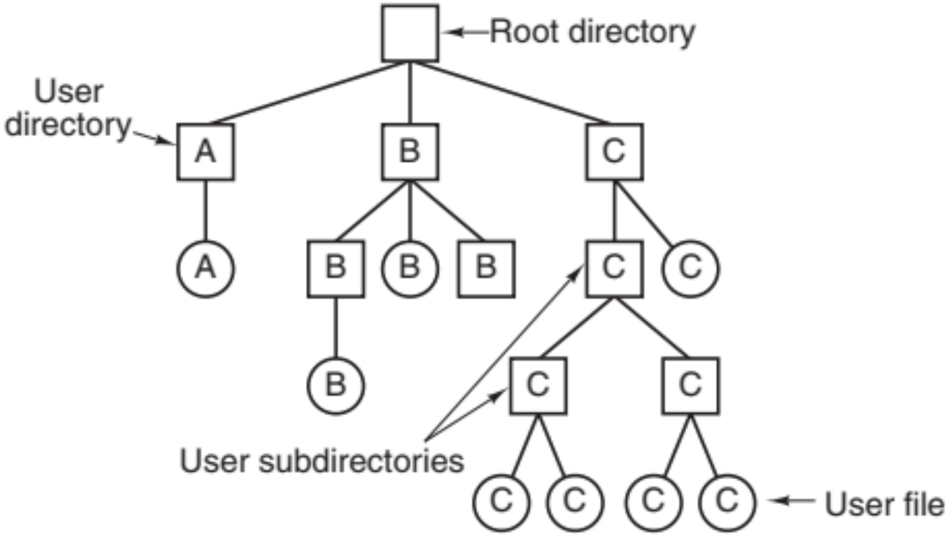
Program copyfile:

copyfile fileA fileB

Directory Systems

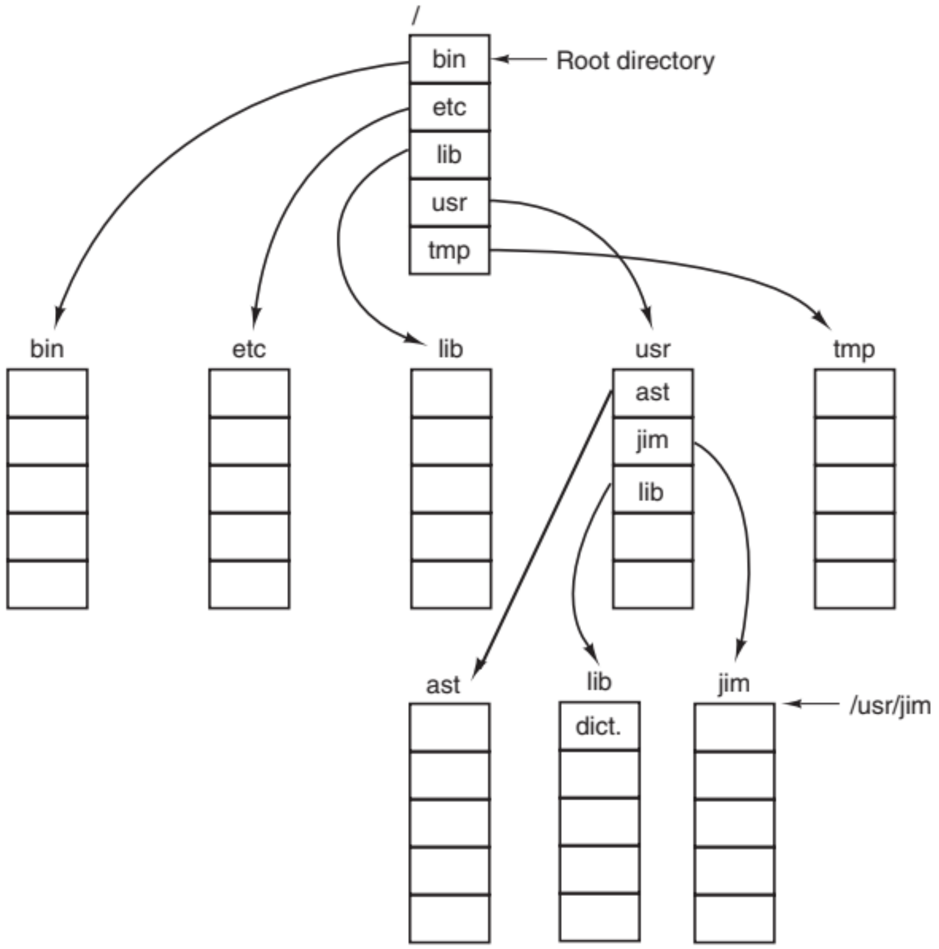


A single-level directory system

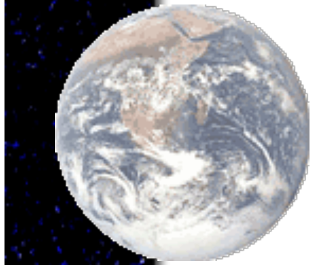


A hierarchical directory system

Path Names

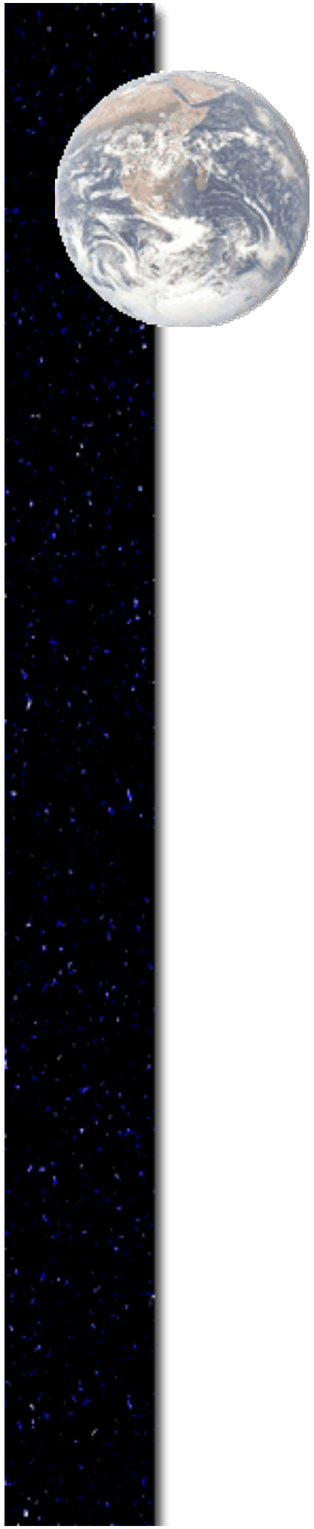


A UNIX directory tree



Directory Operations

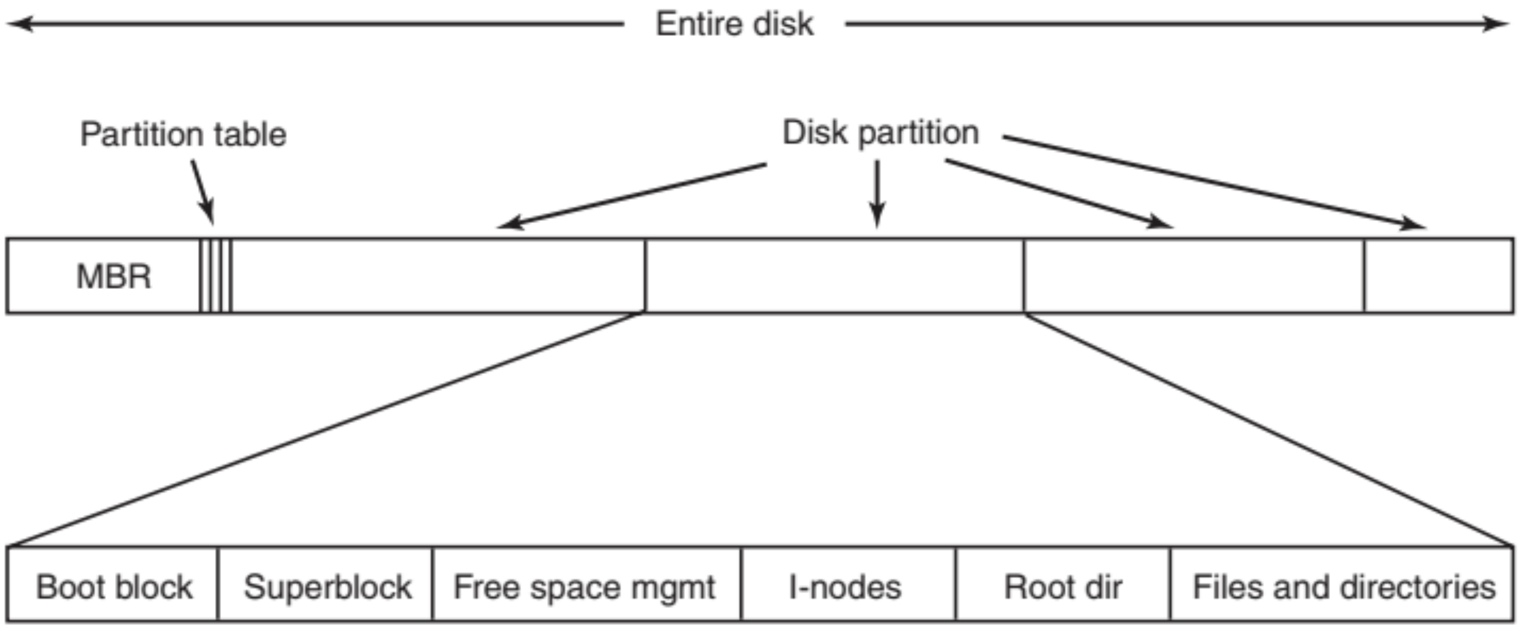
- System calls for managing directories:
  - Create
  - Delete
  - Opendir
  - Closedir
  - Readdir
  - Rename
  - Link
  - Unlink



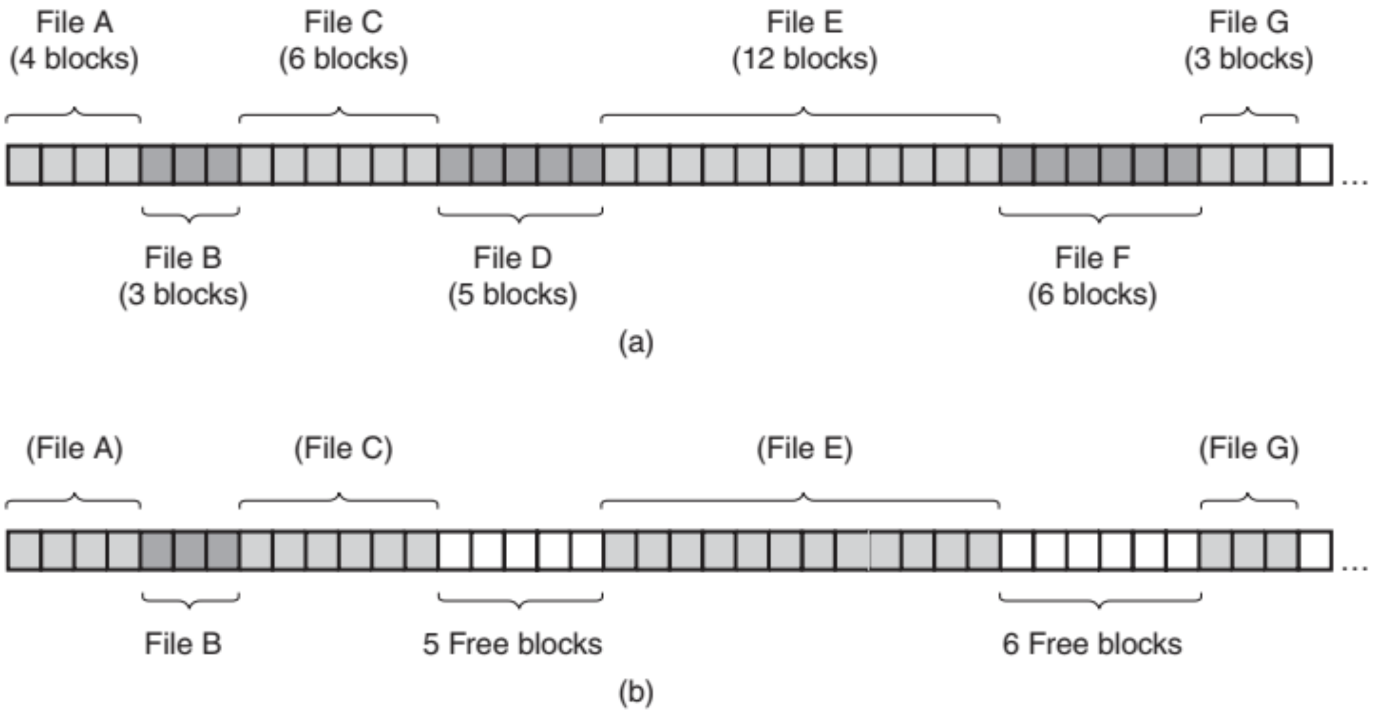
File System Implementation

- File System Layout
- Implementing File Systems
  - Contiguous allocation
  - Linked-list allocation
  - Linked-list allocation using table in memory
  - I-nodes
- Implementing Directories
- Shared Files

A Possible File System Layout



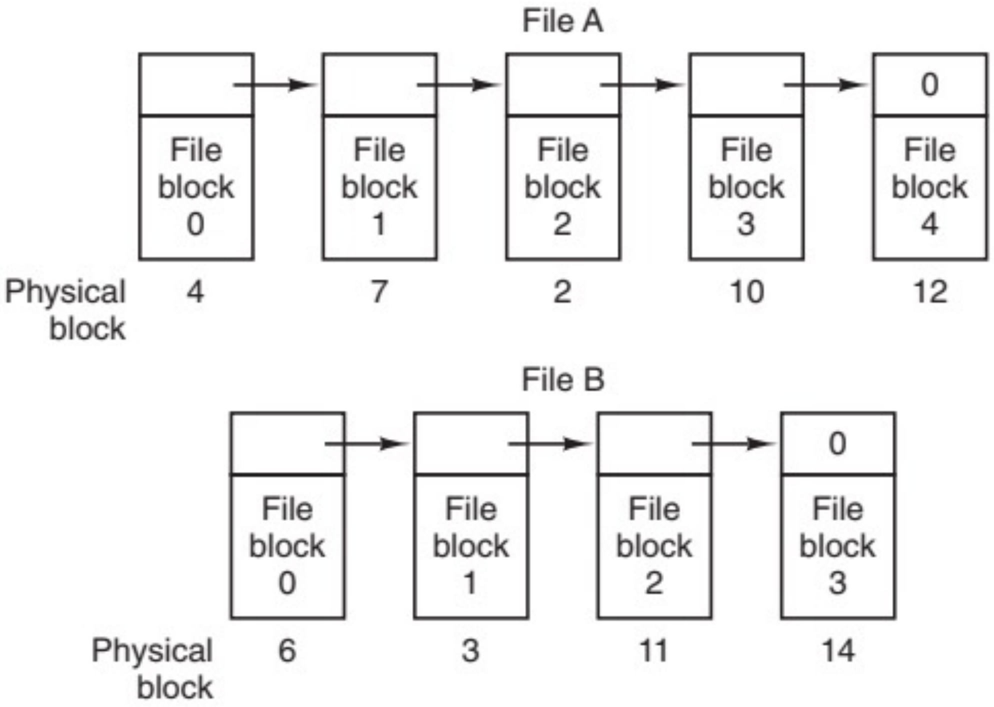
Contiguous Allcation



(a) Contiguous allocation of disk space for 7 files

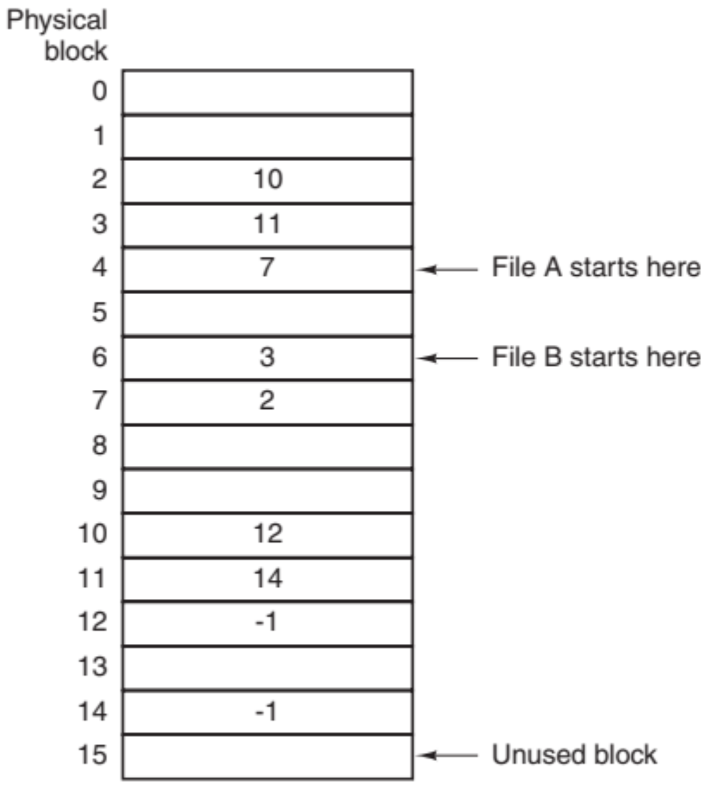
(b) The state of the disk after D and F have been removed

Linked-List Allocation



Storing a file as a linked list of disk blocks

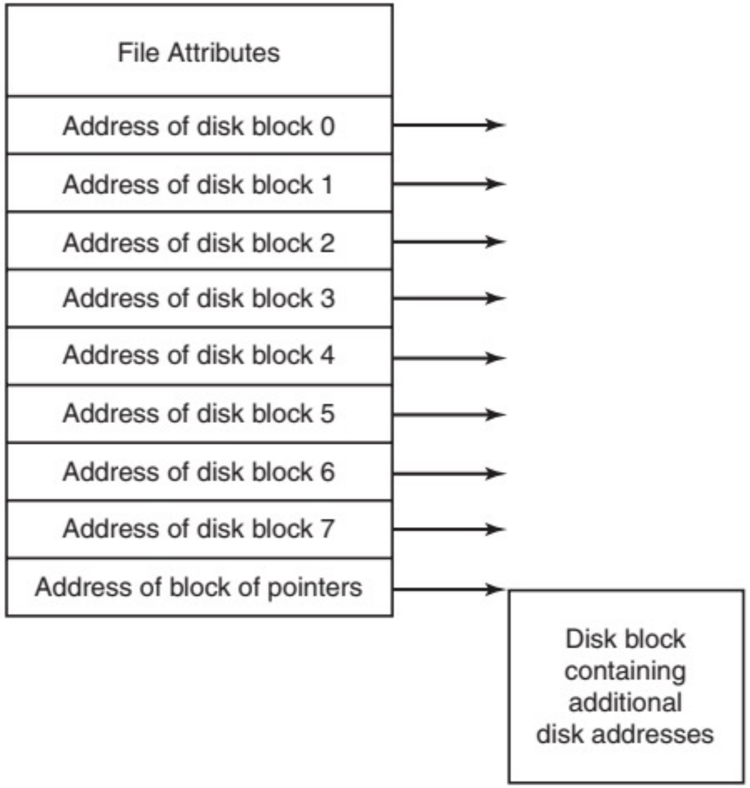
Link-List Allocation Using a Table in Memory



Physical block	
0	
1	
2	10
3	11
4	7
5	
6	3
7	2
8	
9	
10	12
11	14
12	-1
13	
14	-1
15	

Linked list allocation using a file allocation table in memory

I-node



File Attributes

Address of disk block 0

Address of disk block 1

Address of disk block 2

Address of disk block 3

Address of disk block 4

Address of disk block 5

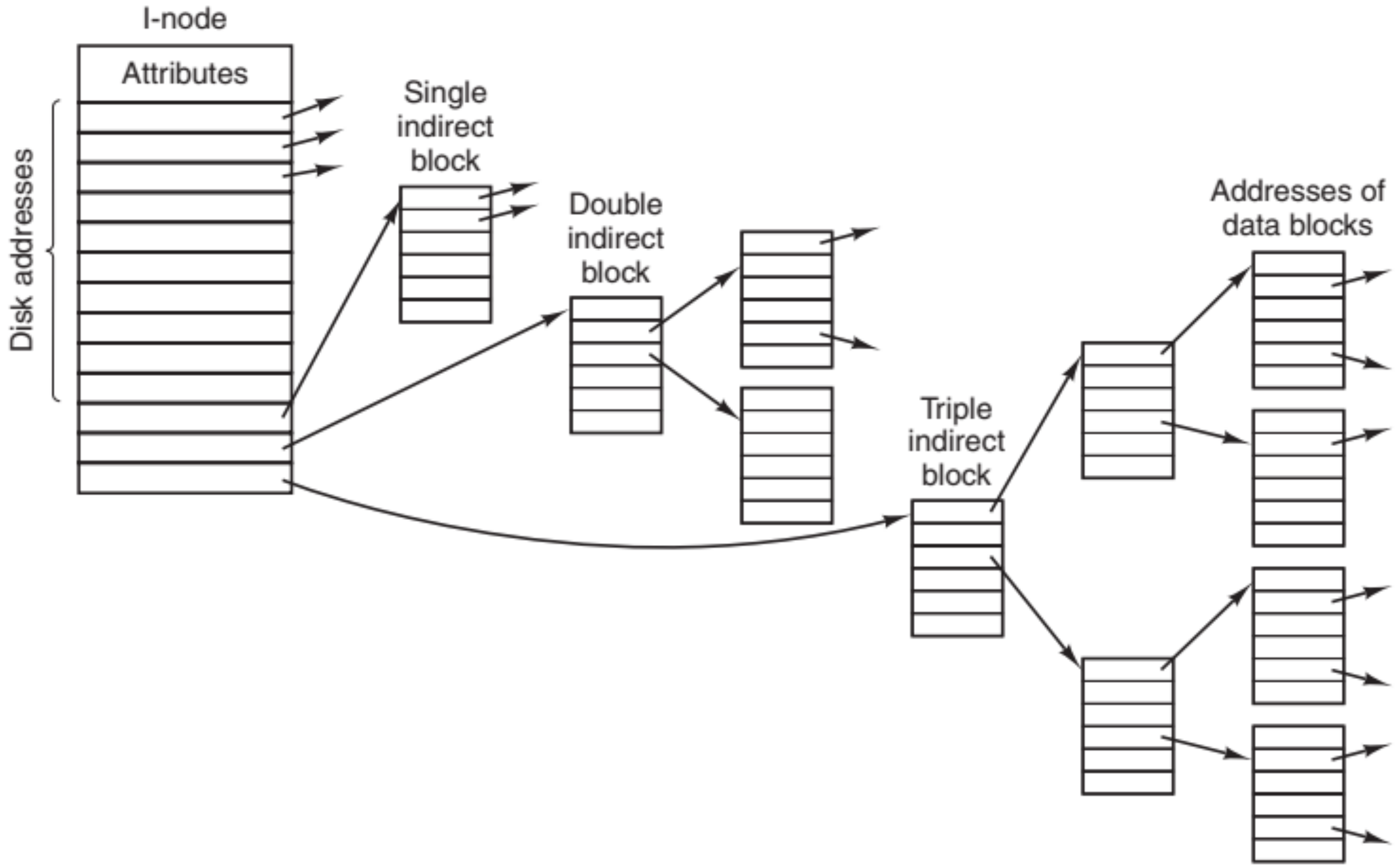
Address of disk block 6

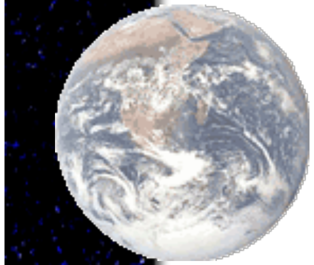
Address of disk block 7

Address of block of pointers

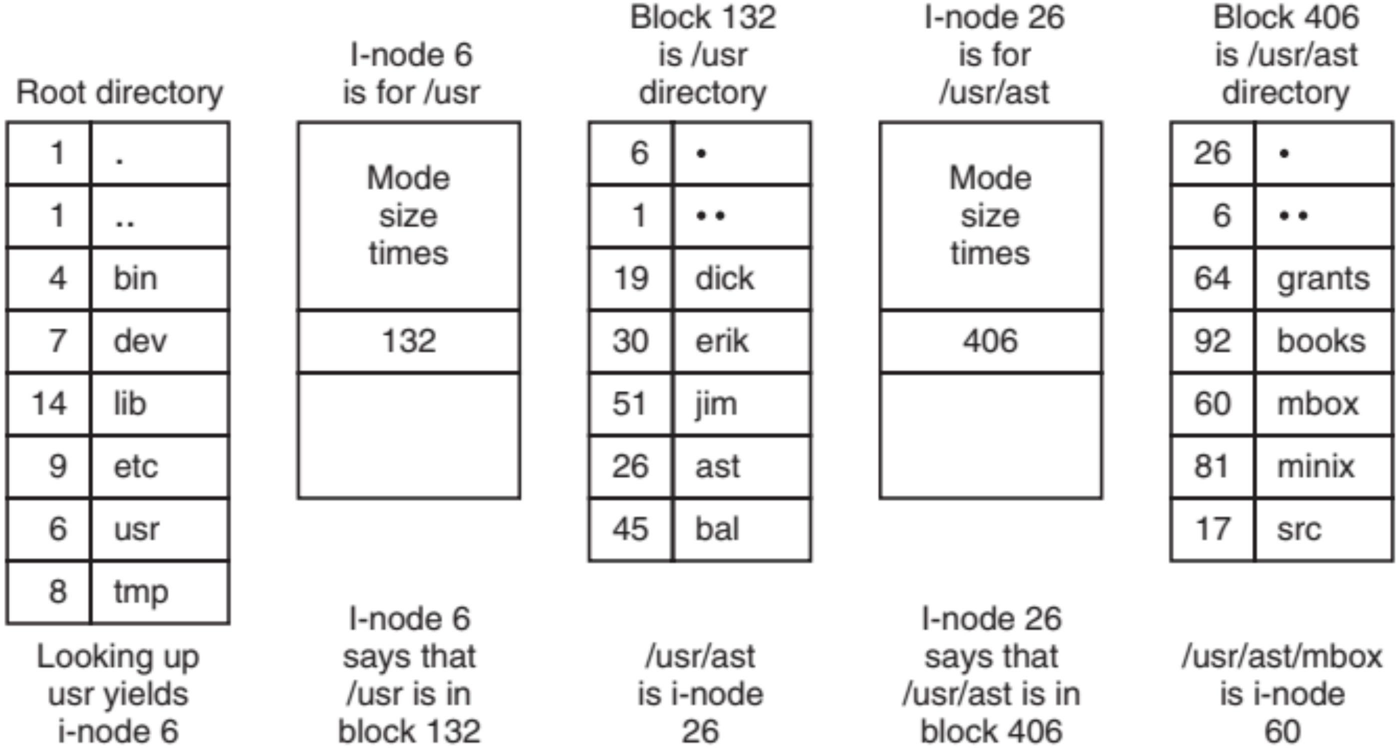
Disk block containing additional disk addresses

A UNIX I-node


I-node
Attributes
Disk addresses
Single indirect block
Double indirect block
Triple indirect block
Addresses of data blocks



The Steps in Lookin Up /usr/ast/mbox



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

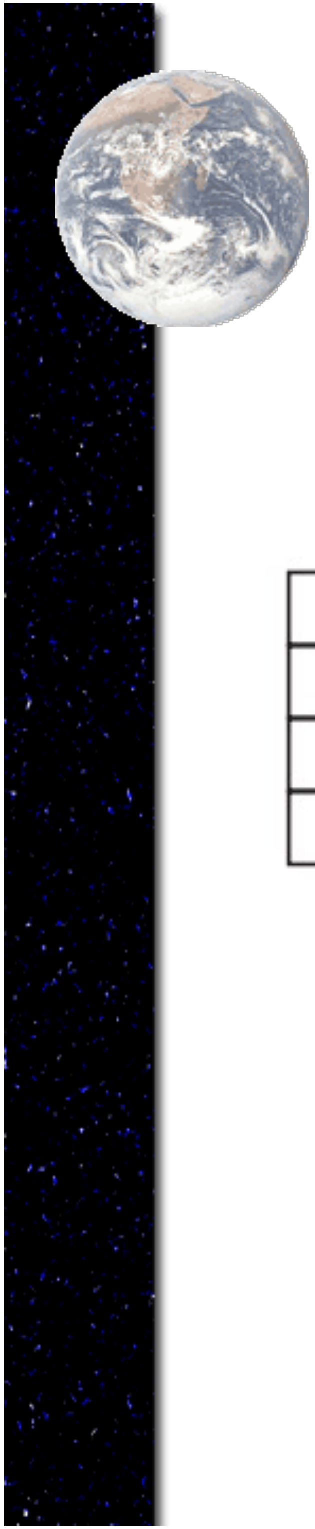
Mode size times
406

I-node 26 says that /usr/ast is in block 406

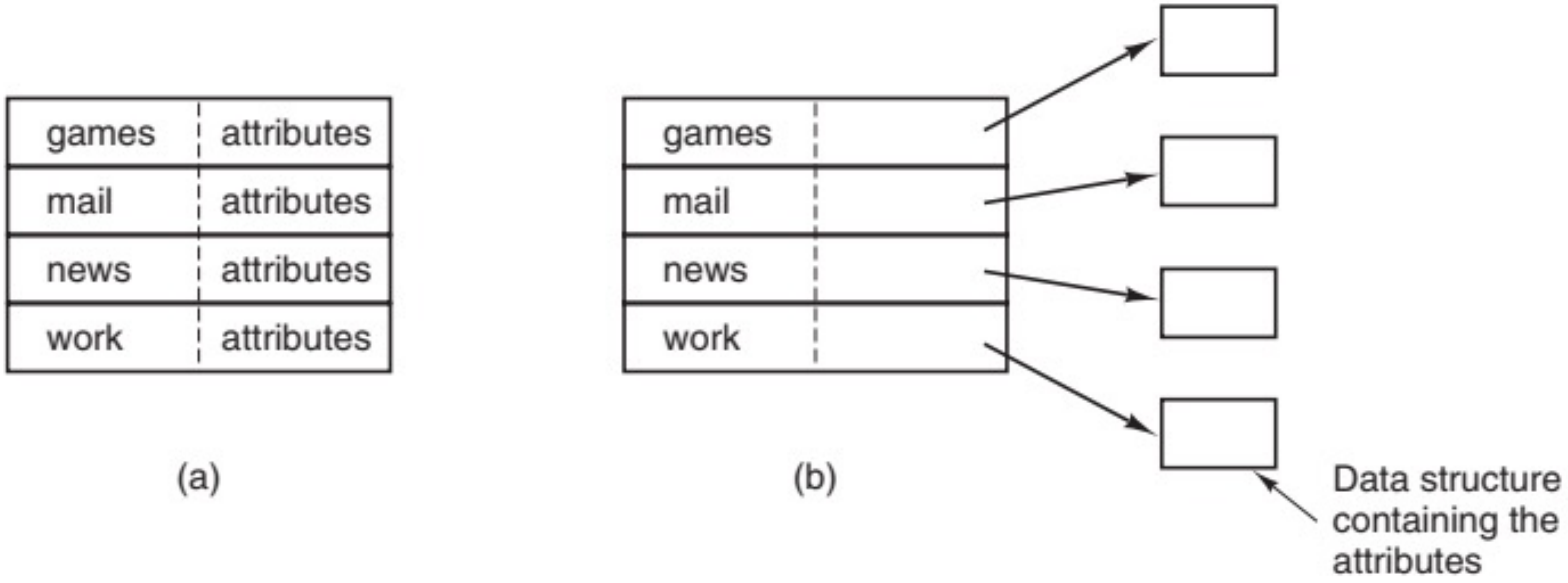
Block 406 is /usr/ast directory

26	•
6	• •
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox is i-node 60

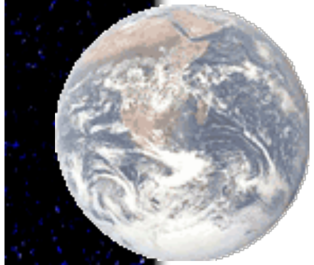


Implementing Directories

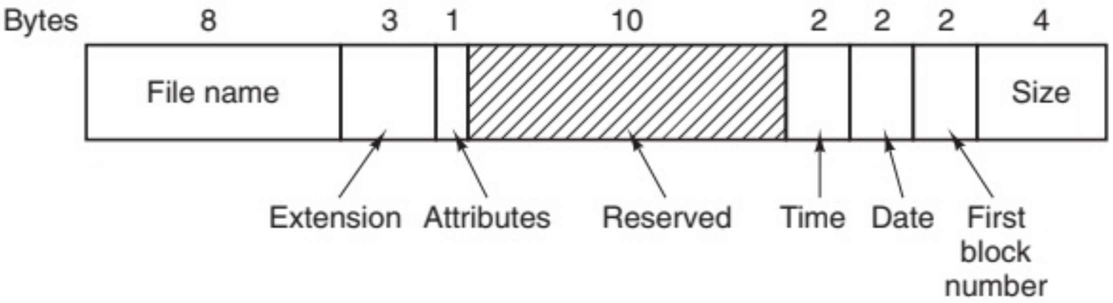


(a) A simple directory containing fixed-size entries with disk addresses and attributes in the directory entry

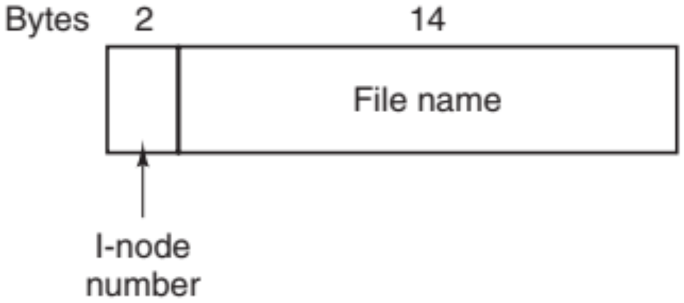
(b) A directory in which each entry just refers to an i-node



Examples of Directory Entry

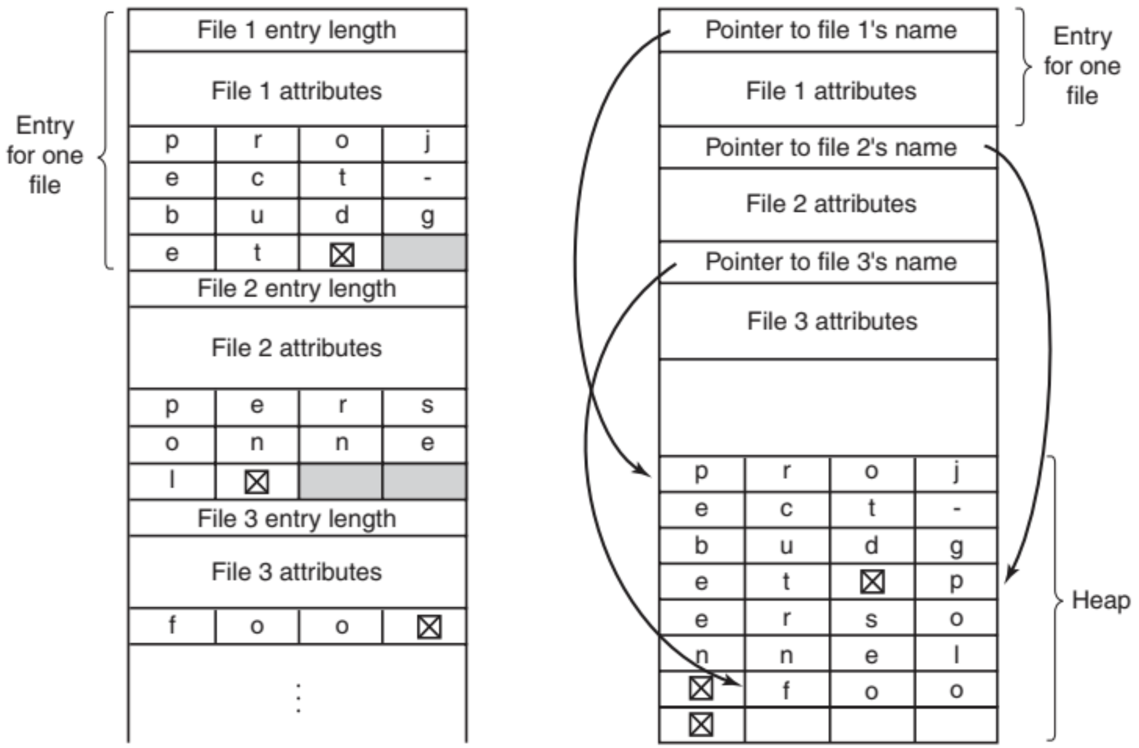


The MS-DOS directory entry



A UNIX directory entry

Handling Long File Name



Entry for one file

File 1 entry length

File 1 attributes

p	r	o	j
e	c	t	-
b	u	d	g
e	t	☒	▨

File 2 entry length

File 2 attributes

p	e	r	s
o	n	n	e
l	☒	▨	▨

File 3 entry length

File 3 attributes

f	o	o	☒

⋮

(a)

Pointer to file 1's name

File 1 attributes

Entry for one file

Pointer to file 2's name

File 2 attributes

Pointer to file 3's name

File 3 attributes

Heap

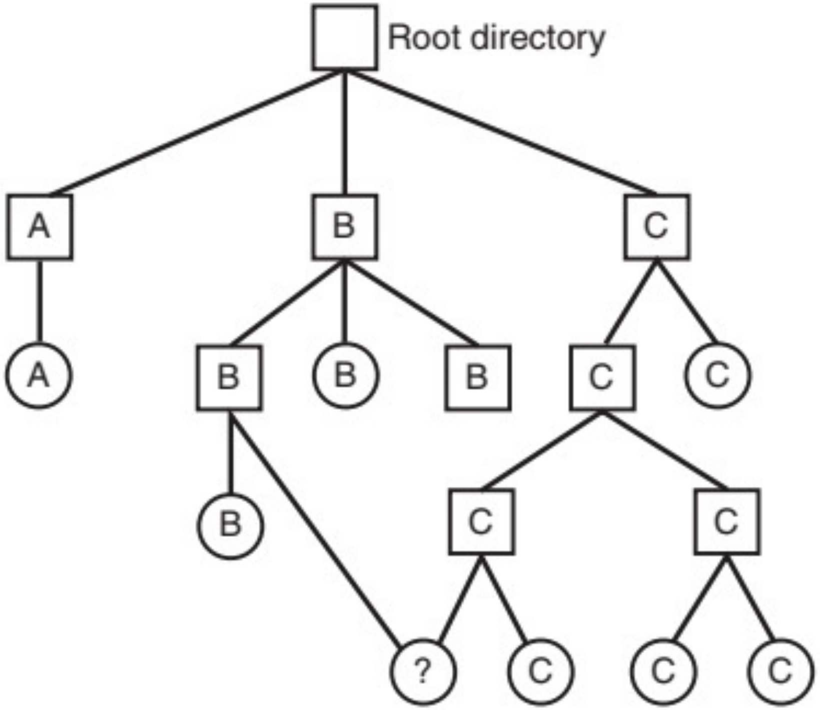
p	r	o	j
e	c	t	-
b	u	d	g
e	t	☒	p
e	r	s	o
n	n	e	l
☒	f	o	o
☒			

(b)

(a) In-line

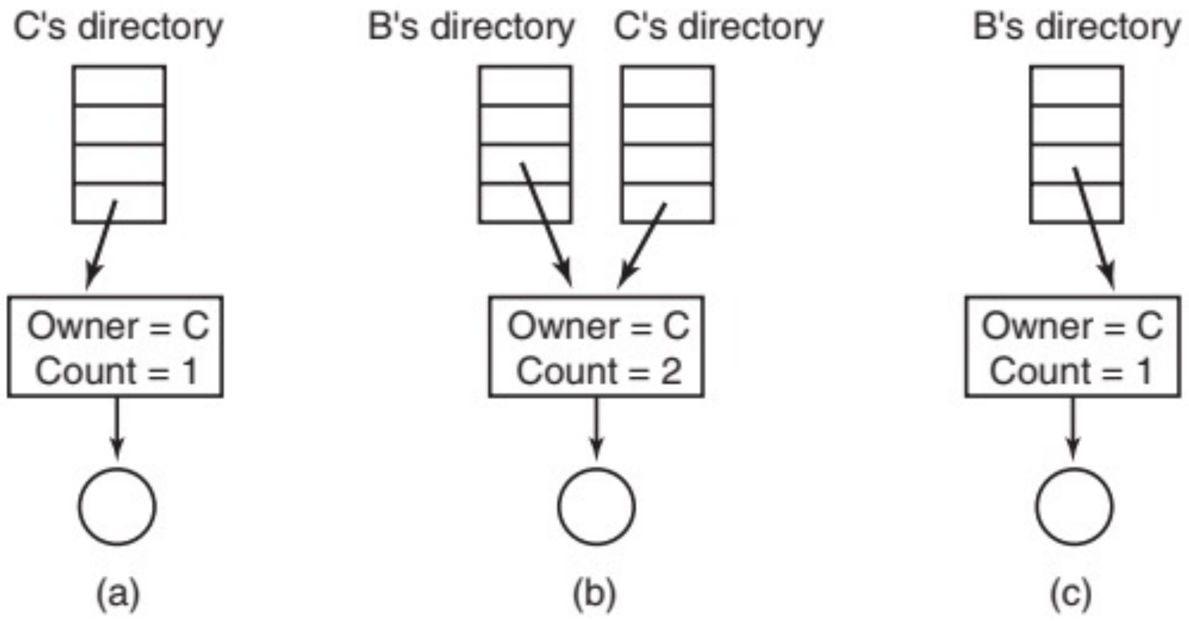
(b) In a heap

Shared Files



Shared file

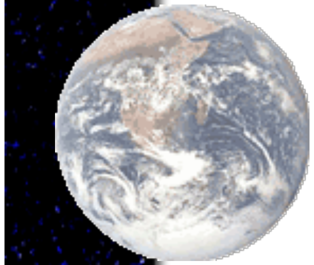
Shared Files



(a) Situation prior to linking

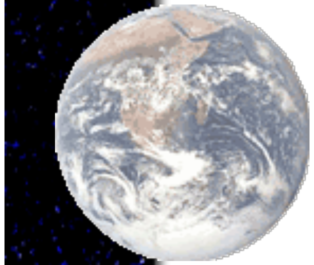
(b) After the link is created

(c) After the original owner removes the file

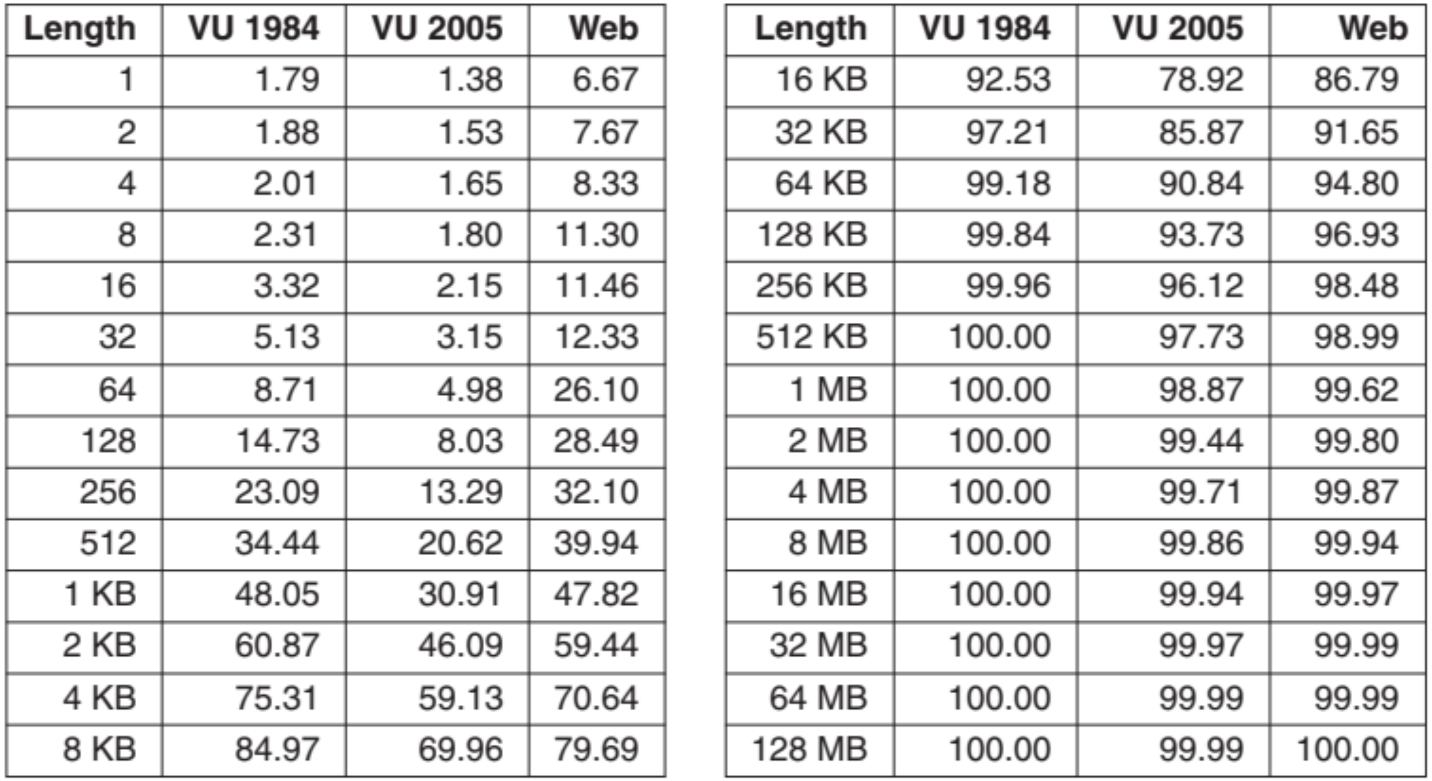


File System Management and Optimization

- Disk space management
- File-system backups
- File-system consistency
- File-system performance
  - Caching
  - Block read ahead
  - Reducing disk arm motion



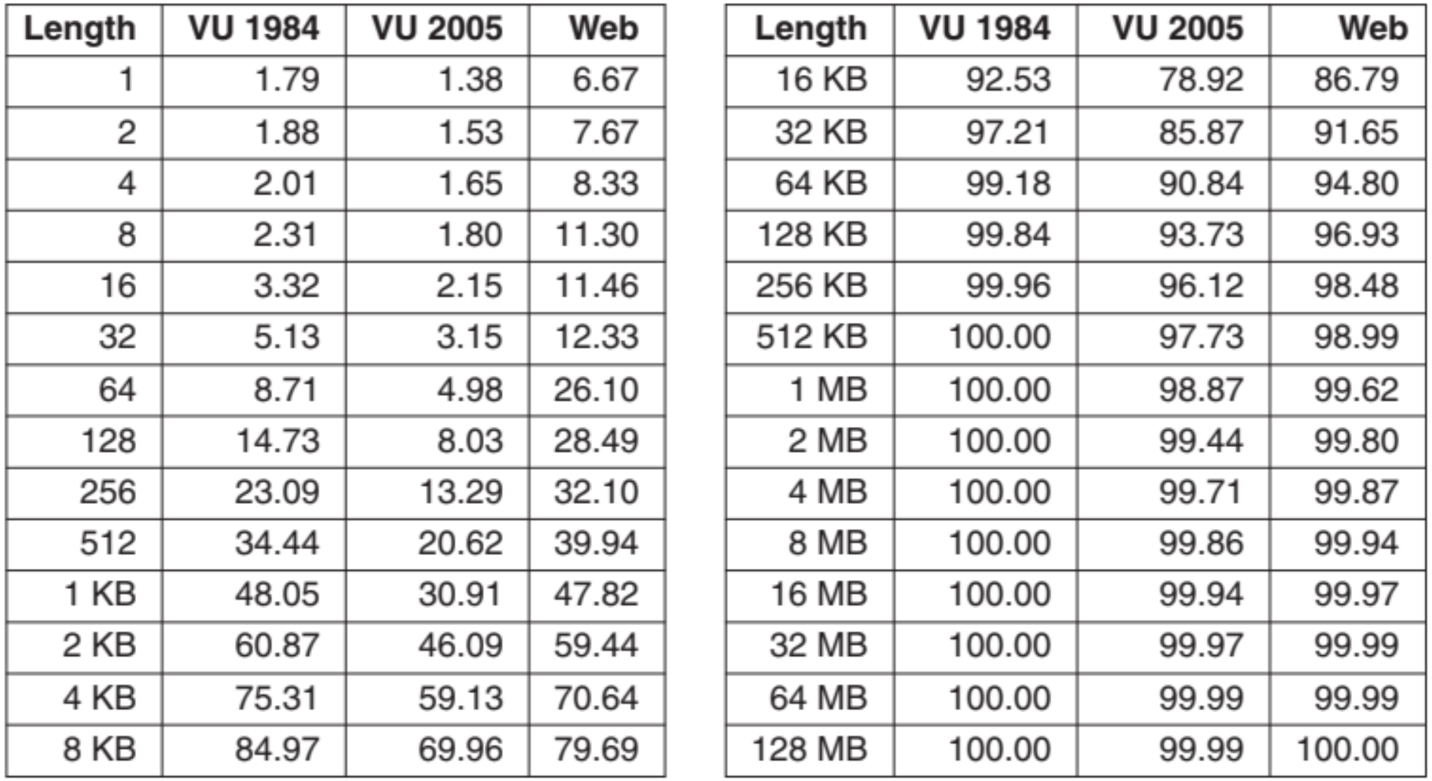
Disk Space Management – Block Size



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

Percentage of files smaller than a given size (in bytes)

Disk Space Management – Block Size

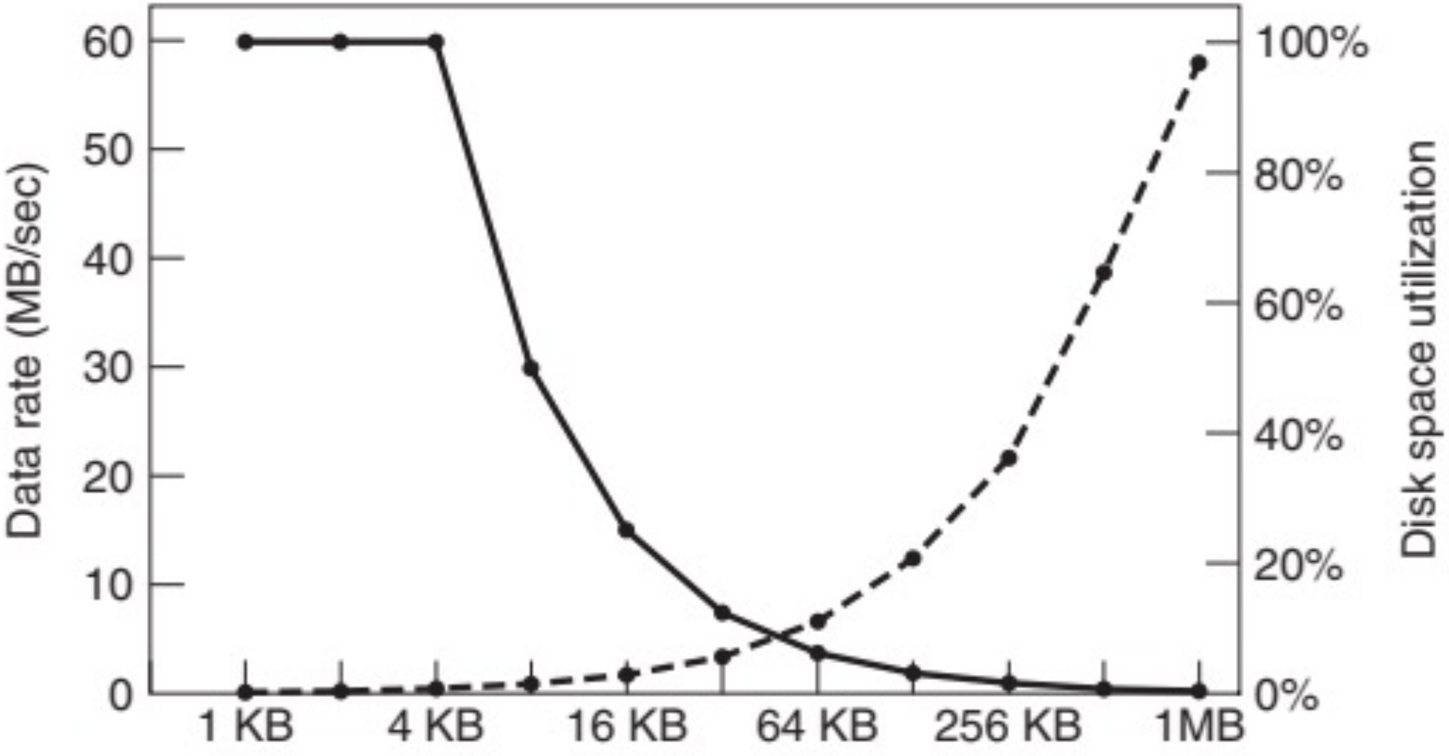


Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69

Length	VU 1984	VU 2005	Web
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

Percentage of files smaller than a given size (in bytes)

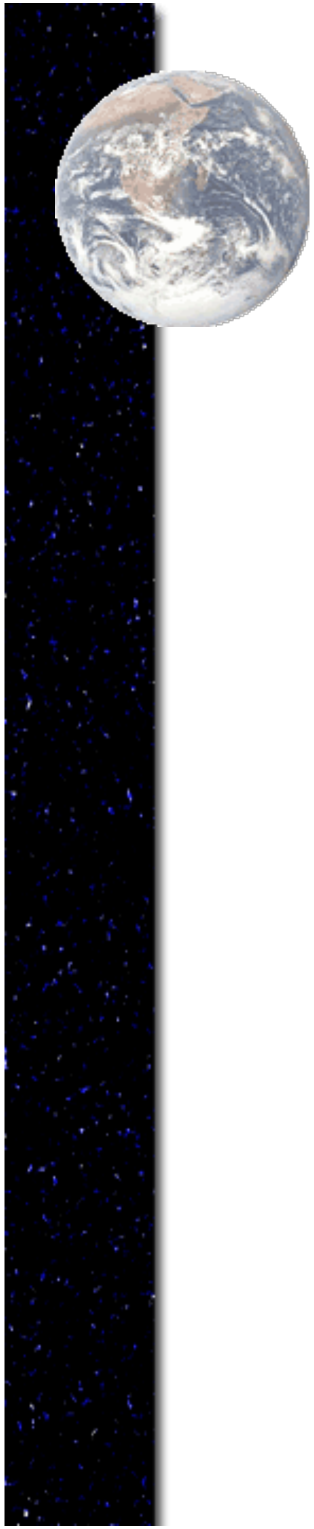
File System Backups



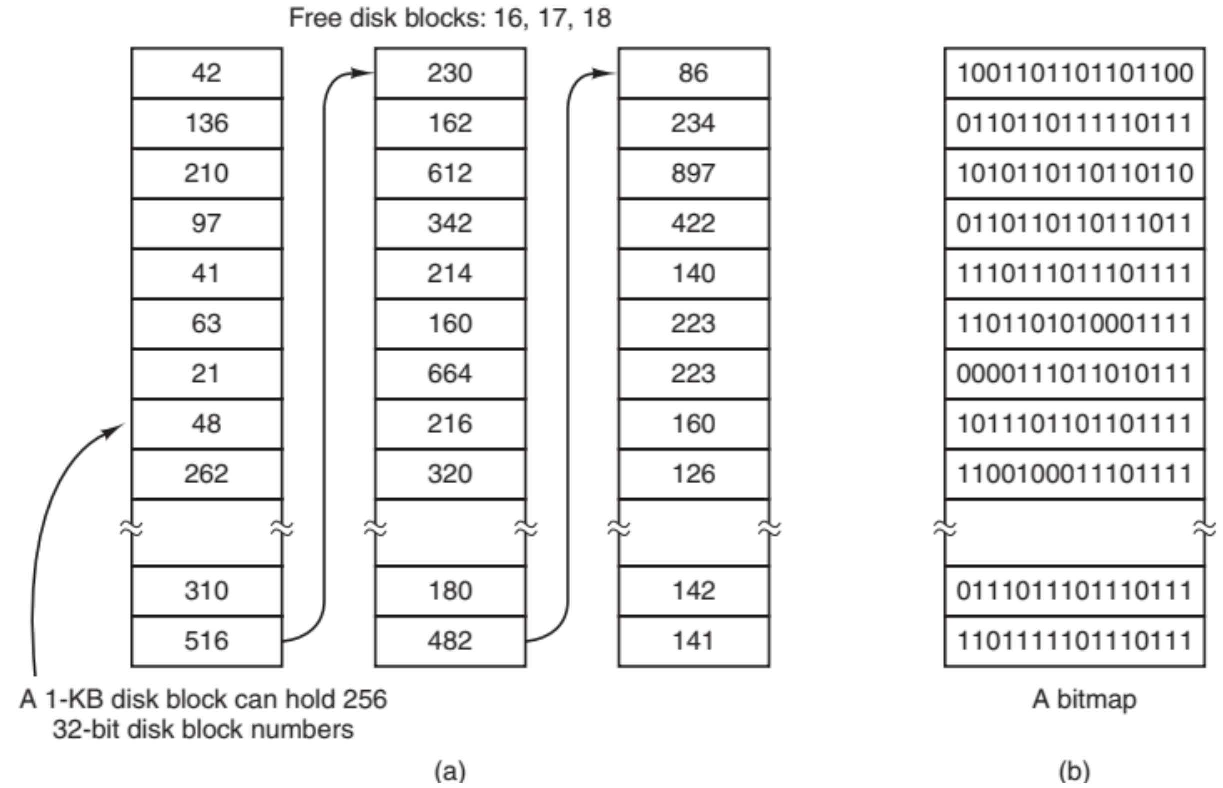
The dashed curve (left-hand scale) gives the data rate of a disk.

The solid curve (right-hand scale) gives the disk-space efficiency.

All files are 4 KB.



Keeping Track of Free Blocks



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

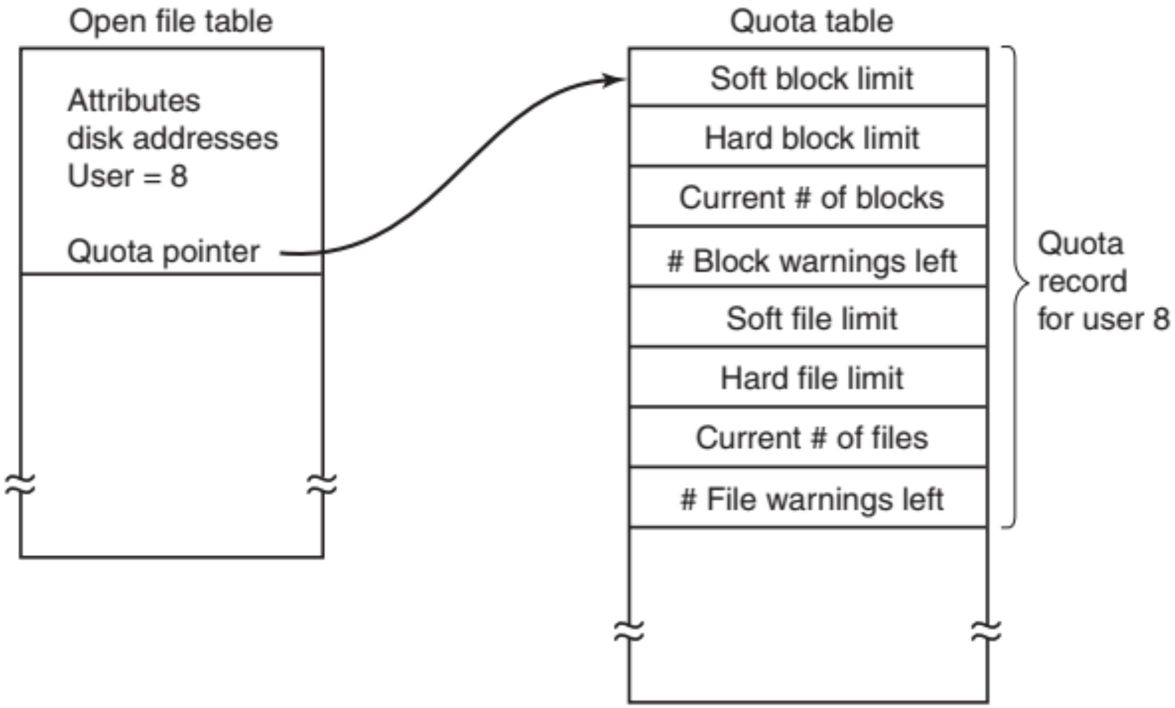
A bitmap

(b)

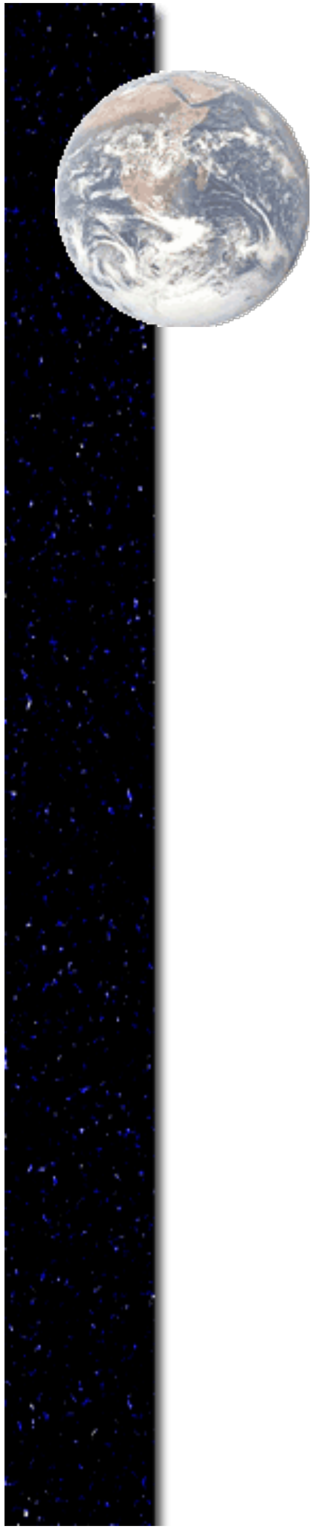
(a) Storing the free list on a linked list.

(b) A bitmap.

Disk Quotas



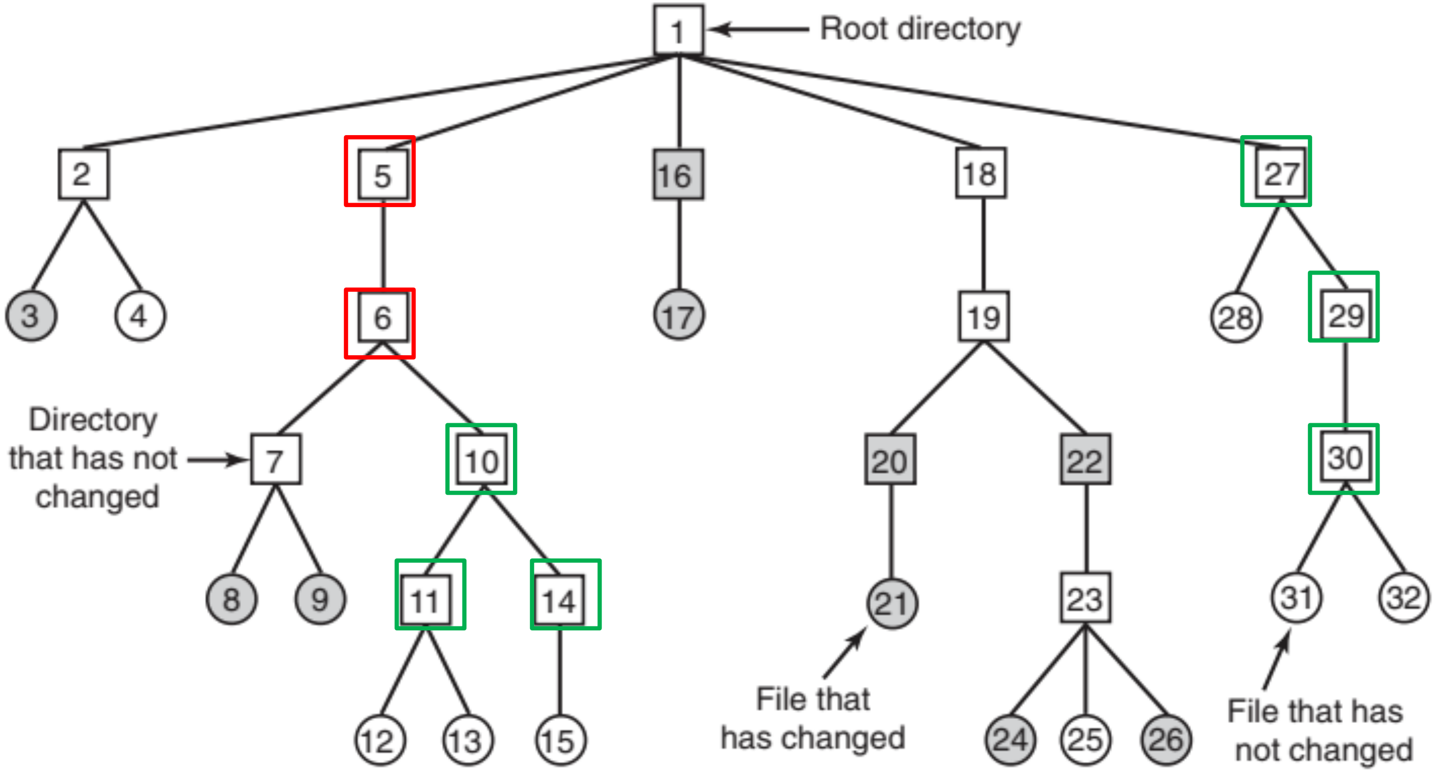
Quotas are kept track of on a per-user basis in a quota table.



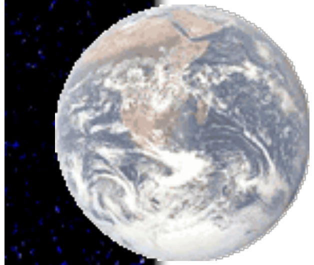
File-System Backups

A file system to be dumped:

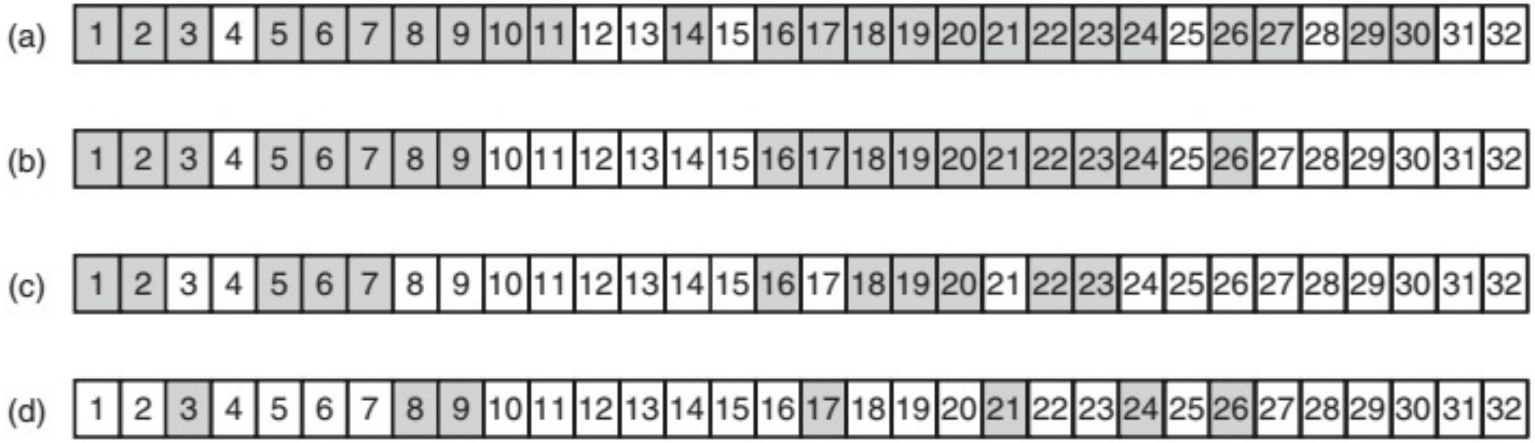
- The squares are directories and the circles are files.
- Each directory and file is labeled by its i-node number.



The shaded items have been modified since the last dump.



Bitmaps Used by the Logical Dumping Algorithm


(a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
(b) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
(c) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
(d) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

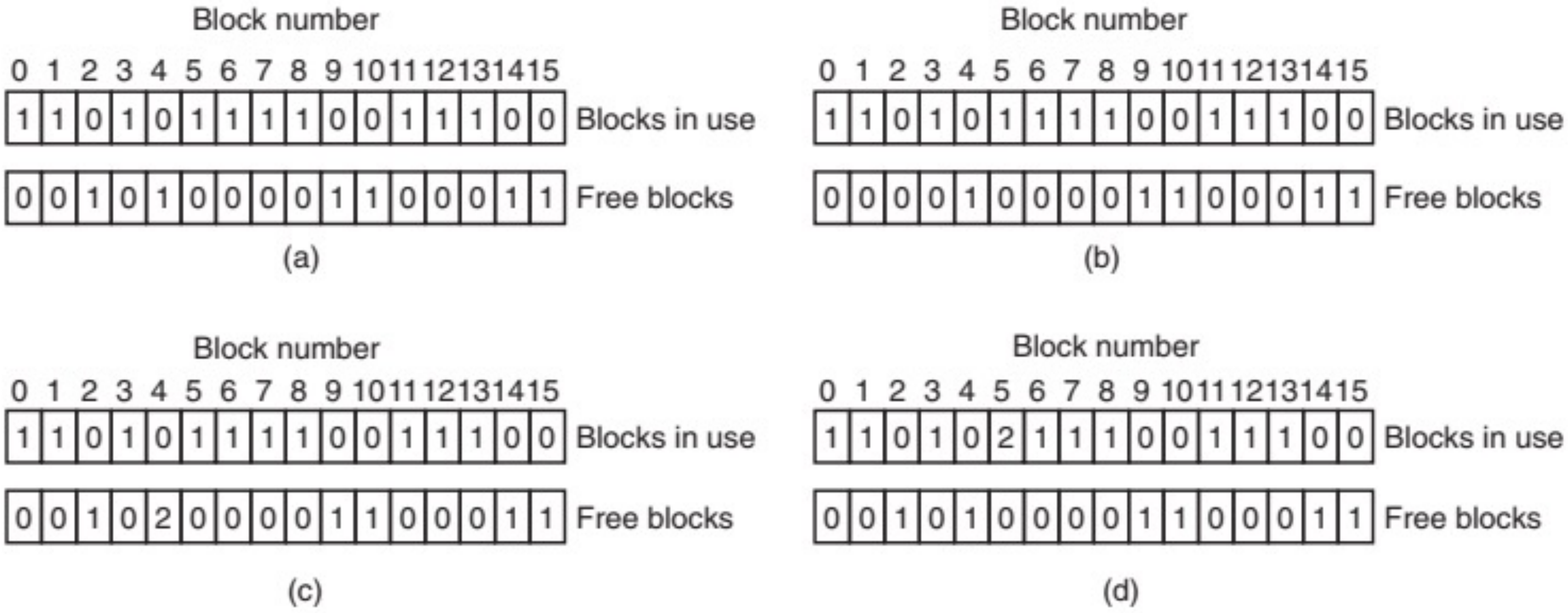
a) All modified files and all directories have been marked in the bitmap, as shown (by shading)

(b) Unmarking any directories that have no modified files or directories in them or under them.

(c) Dumping all the directories that are marked for dumping

(d) Dumping all the files marked

File System Consistency



Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

(a)

Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1

(b)

Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1

(c)

Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

(d)

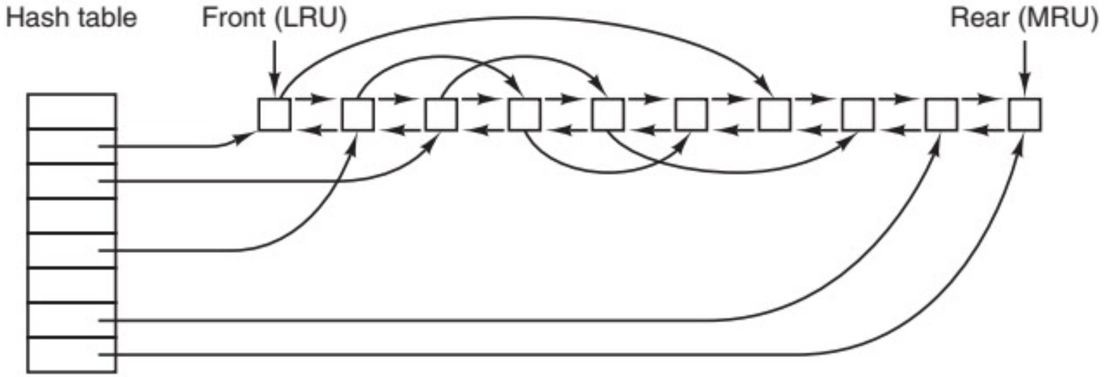
(a) Consistent

(b) Missing block

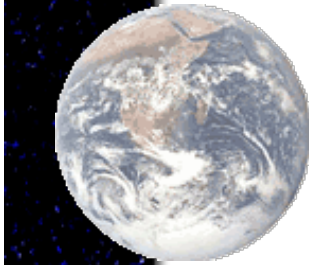
(c) Duplicate block in free list

(d) Duplicate data block

Caching



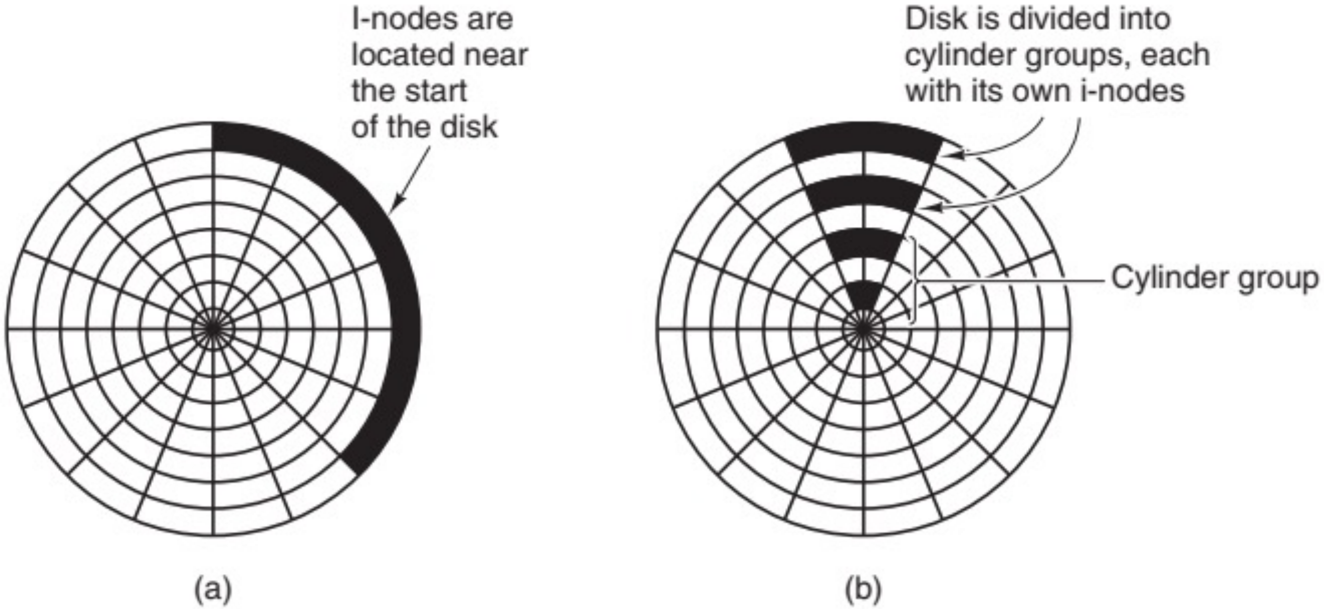
The buffer cache data structures



Block Read Ahead

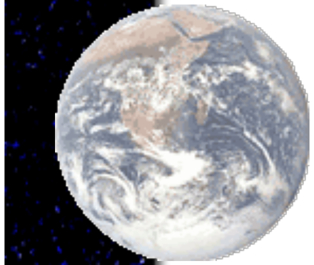
- Get blocks into the cache before they are needed to increase the hit rate.
- For sequentially read file
  - Read block k+1 along with a read request of block k

Reducing Disk Arm Motion


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)

(a) I-nodes placed at the start of the disk.

(b) Disk divided into cylinder groups, each with its own blocks and i-nodes.



References

1. Modern Operating Systems, 4th edition, Andrew S. Tanenbaum, Herbert Bos
2. Operating Systems, 3rd edition, H.M.Deitel, Pearson Education Limited: Longman.
3. Operating System Concepts, 10th edition, Abraham Silberschatz, Yale University, Peter Baer Galvin, Pluribus Networks, Greg Gagne, Westminster College, 10th edition. Wiley.
4. Operating Systems: Internals and Design Principles, 7th edition, William Stallings.