Betriebssysteme und Systemnahe Programmierung

Kapitel 13 • File Systems

Winter 2016/17

Marcel Waldvogel

Storing/Retrieving Information

Essential requirements for long-term information storage:

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

File Naming

Alternatives/Options
- MIME Content-Types (1, 2)
- 4CC Type/Creator (2, 3)

Function
1. Classification
2. Format
3. Default Application

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

Figure 5-1. Some typical file extensions.

File Structure (1)

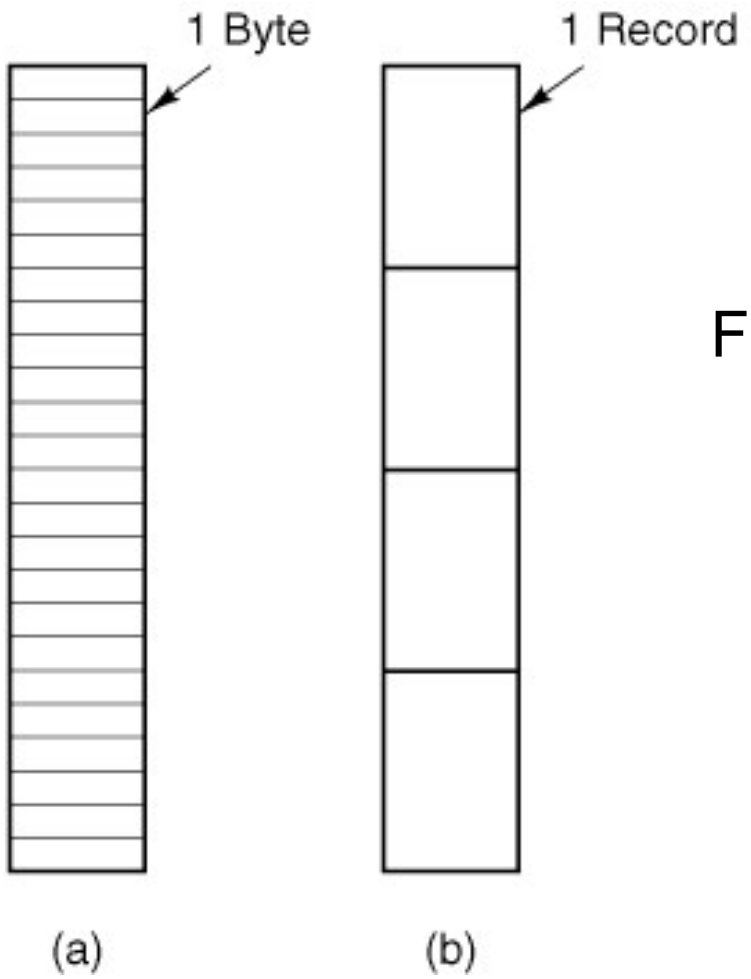


Figure 5-2. Three kinds of files.

(a) Byte sequence.

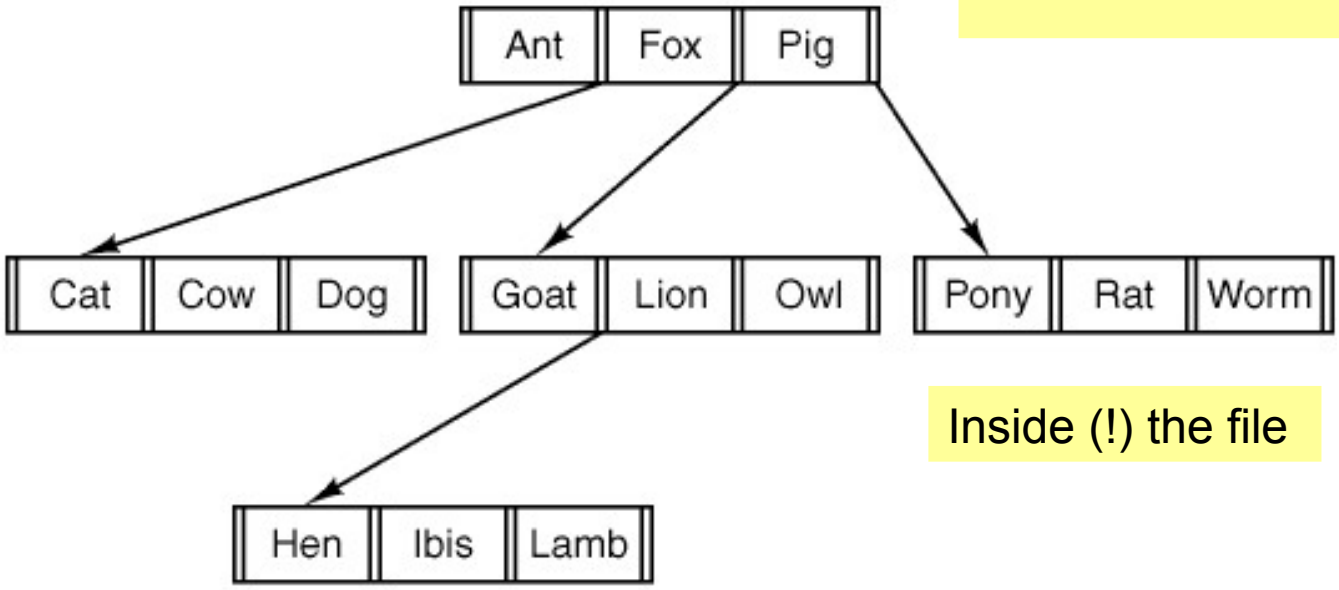
(b) Record sequence.

. . .

- Fixed length
- Variable length
- Line

File Structure (2)

Alternatives/Options
- "App Wrapper"



Inside (!) the file

Figure 5-2. Three kinds of files. … (c) Tree.

File Types (1)

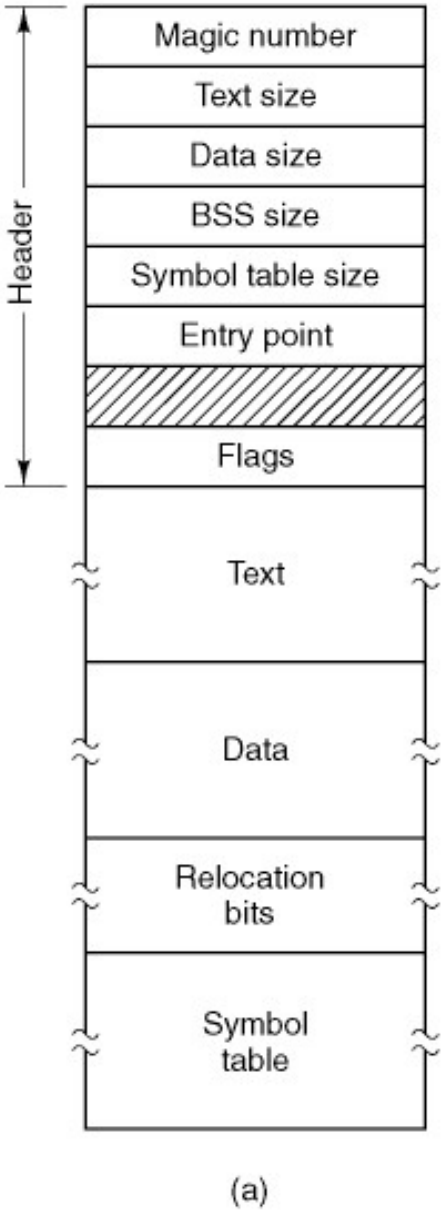


Figure 5-3. (a) An executable file.

…

No special meaning to the FS!

File Types (2)

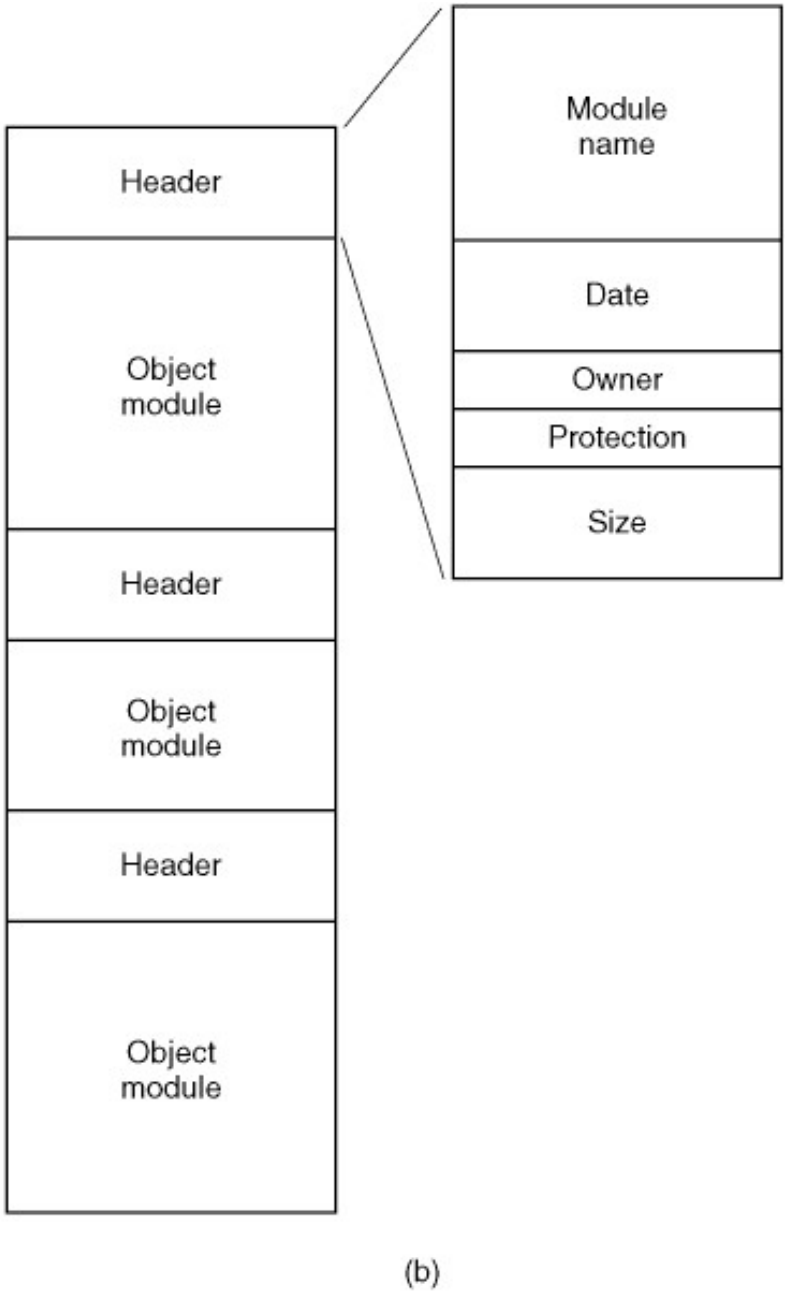


Figure 5-3. … (b) An archive.

File Attributes (1)

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

. . .

Figure 5-4. Some possible file attributes.

File Attributes (2)

. . .

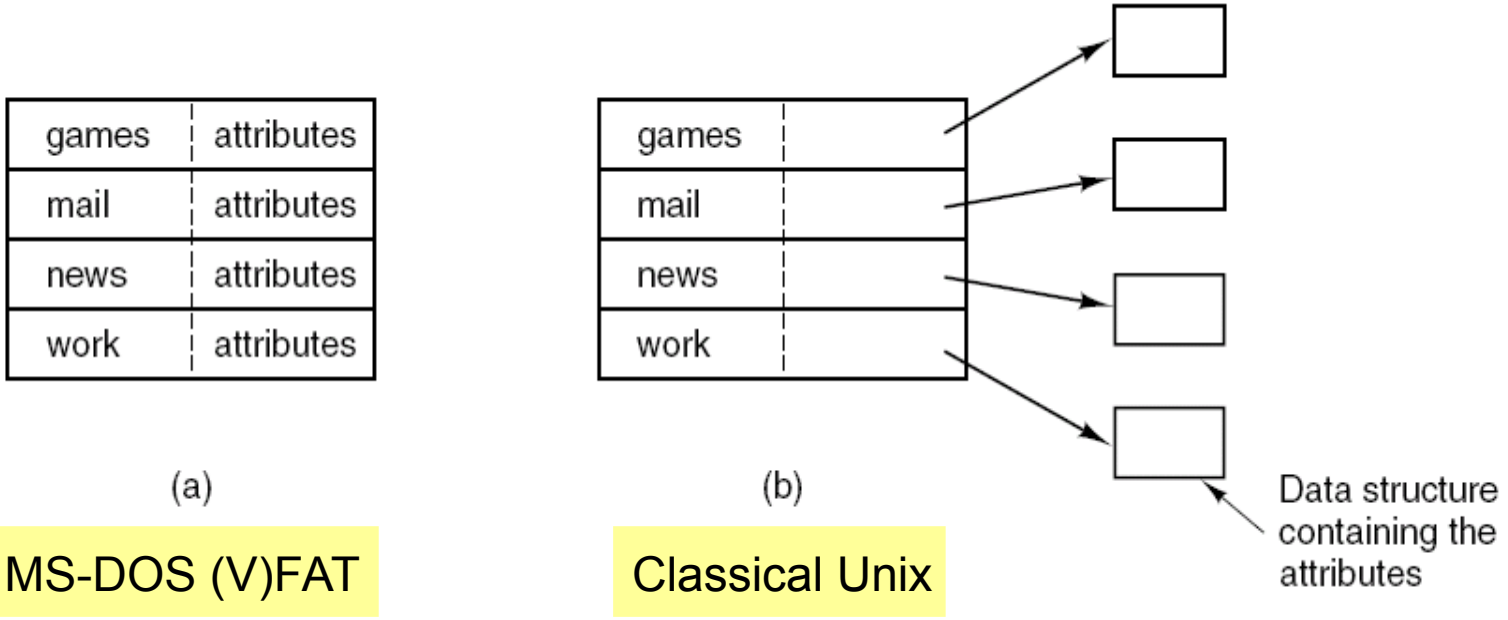
Attribute	Meaning
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was 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

Figure 5-4. Some 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
12. Lock

Directories



MS-DOS (V)FAT

Classical Unix

Figure 5-5. (a) Attributes in the directory entry. (b) Attributes elsewhere.

Hierarchical Directory Systems

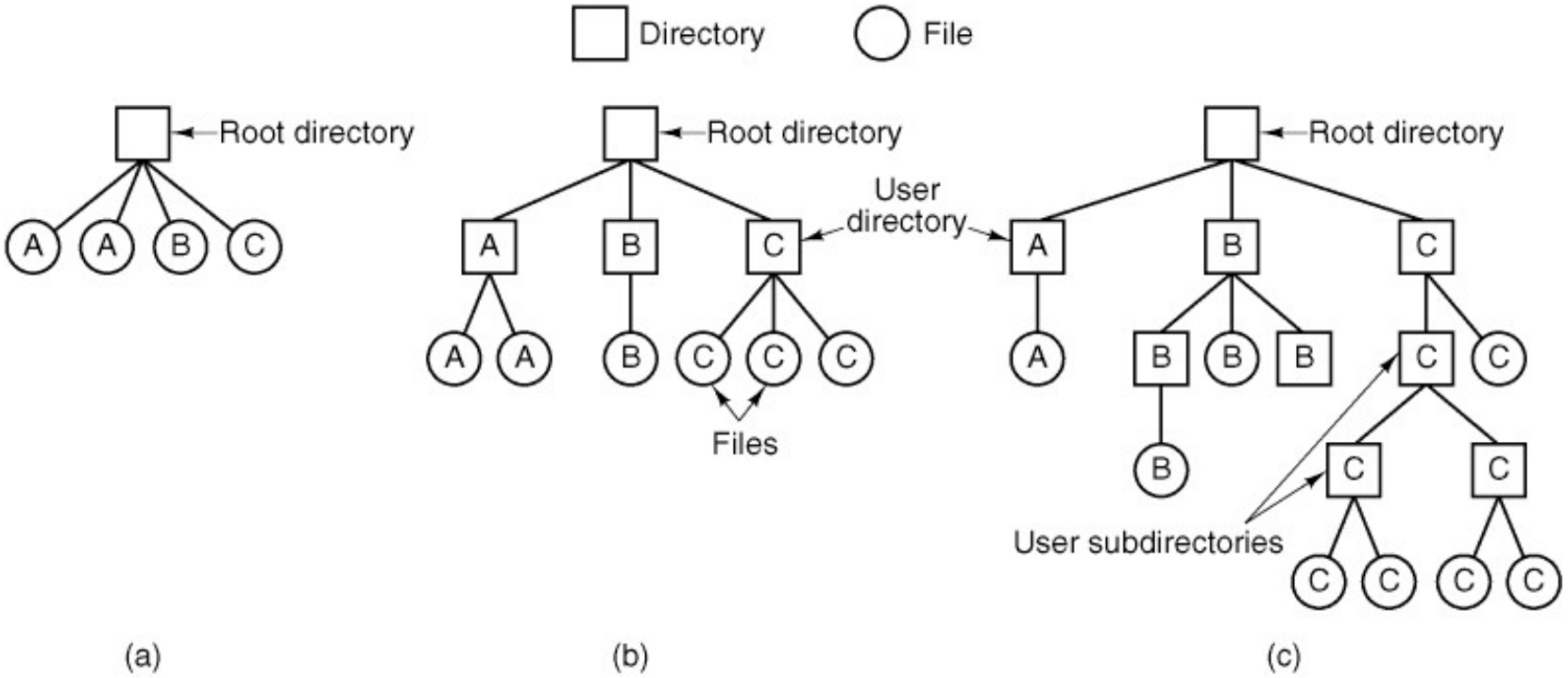


Figure 5-6. Three file system designs. (a) Single directory shared by all users. (b) One directory per user. (c) Arbitrary tree per user. The letters indicate the directory or file's owner.

Path Names

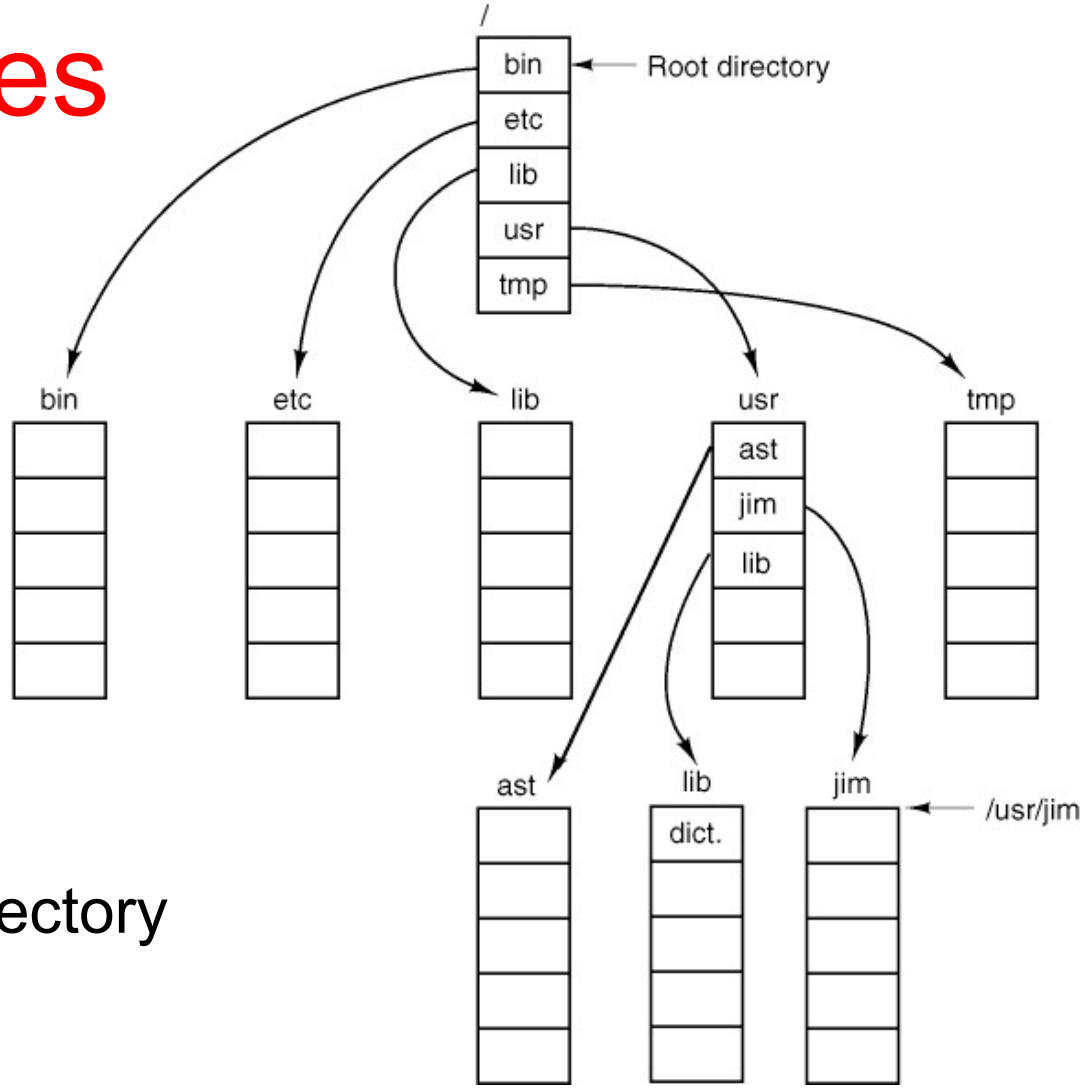


Figure 5-7. A UNIX directory tree.

Directory Operations

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

File System Layout

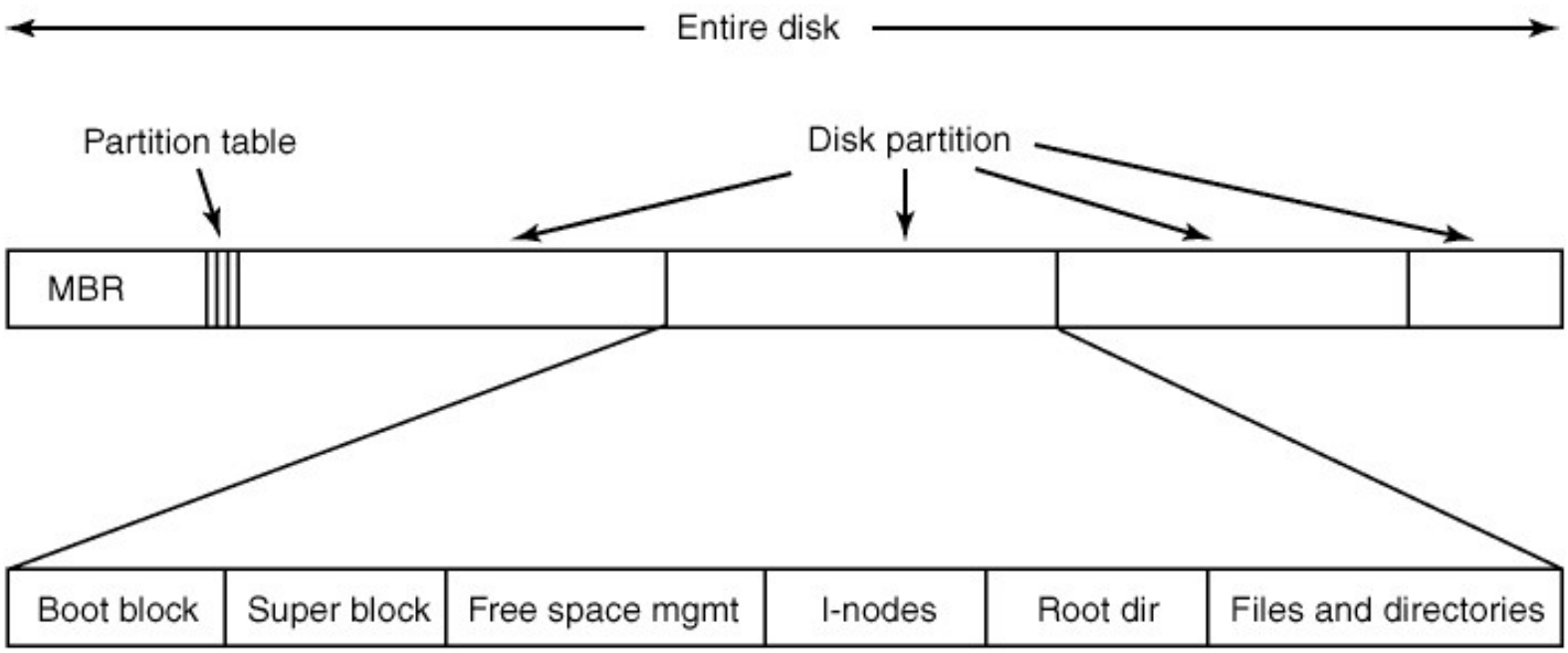
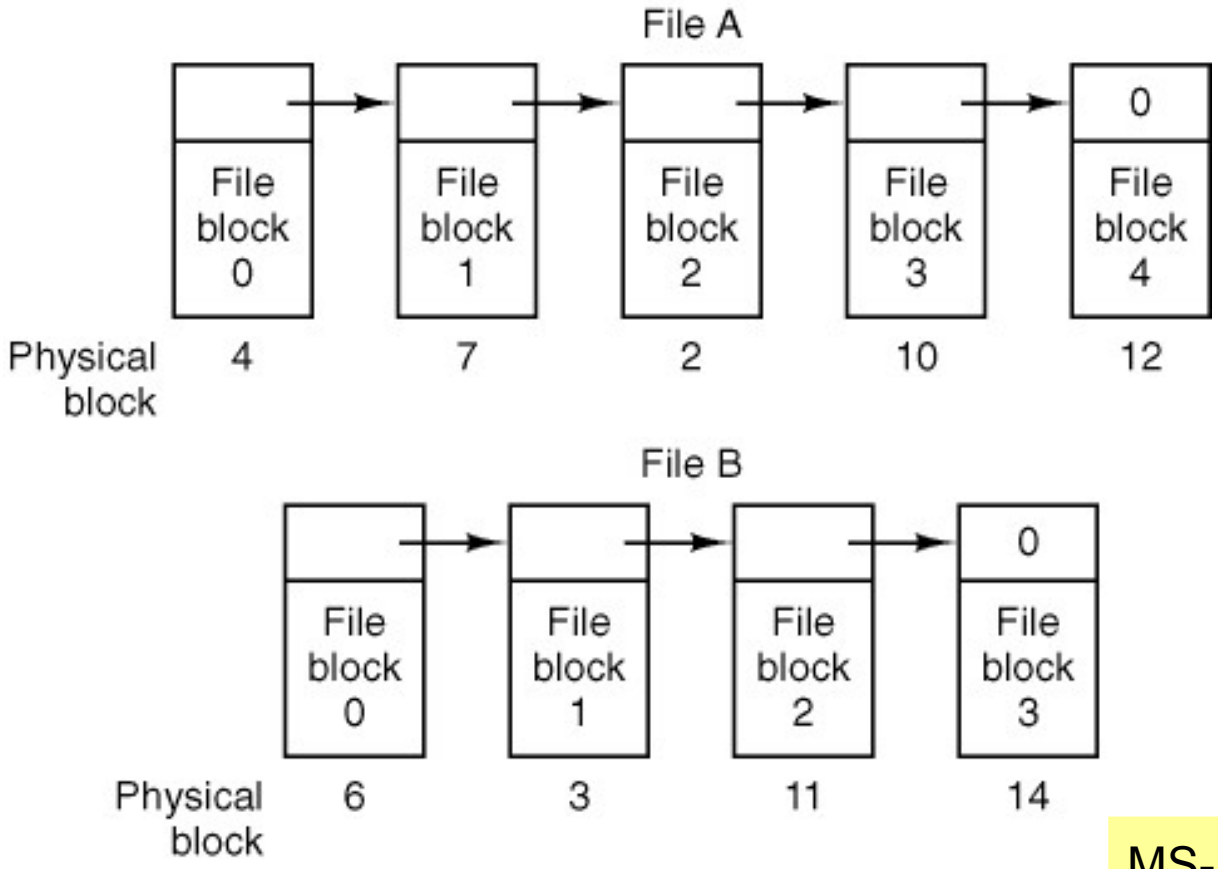


Figure 5-8. A possible file system layout.

Linked List Allocation

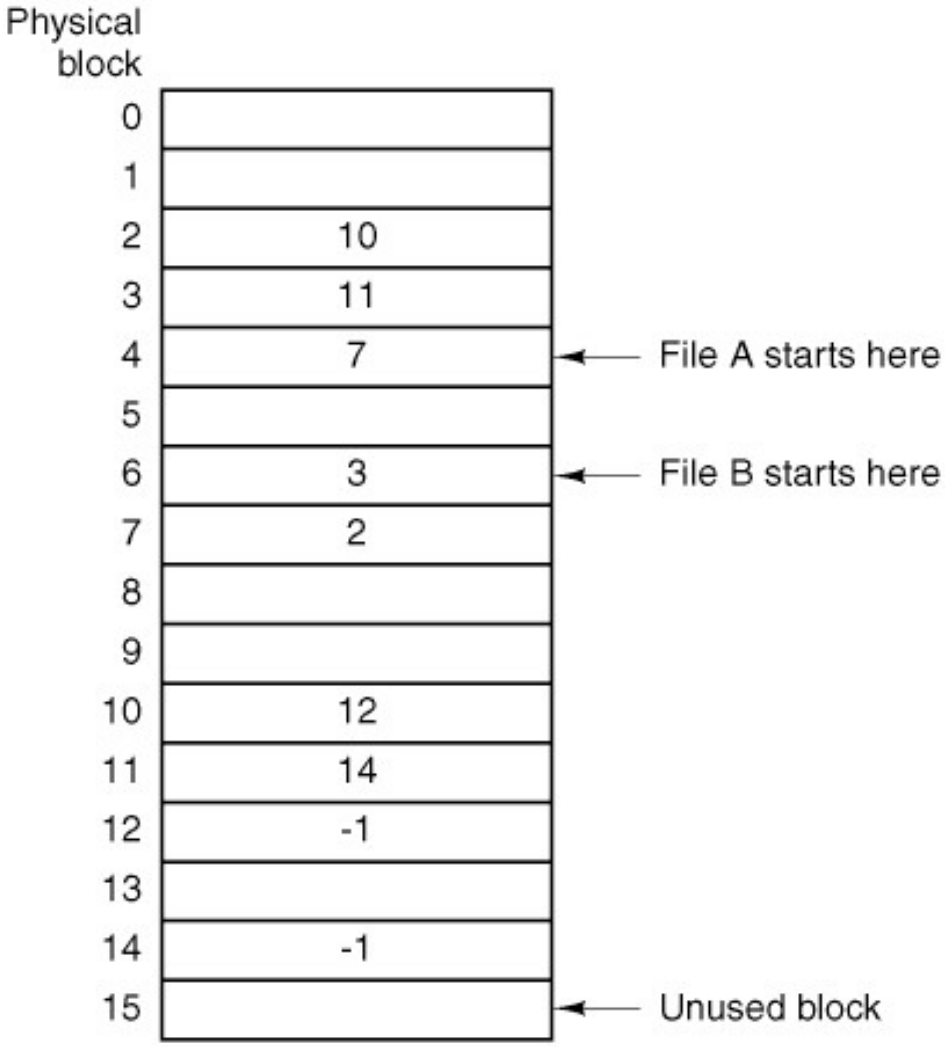


MS-DOS (V)FAT

Figure 5-9. Storing a file as a linked list of disk blocks.

Linked List Allocation Using a Table in Memory

Figure 5-10. Linked list allocation using a file allocation table in main memory.



I-nodes

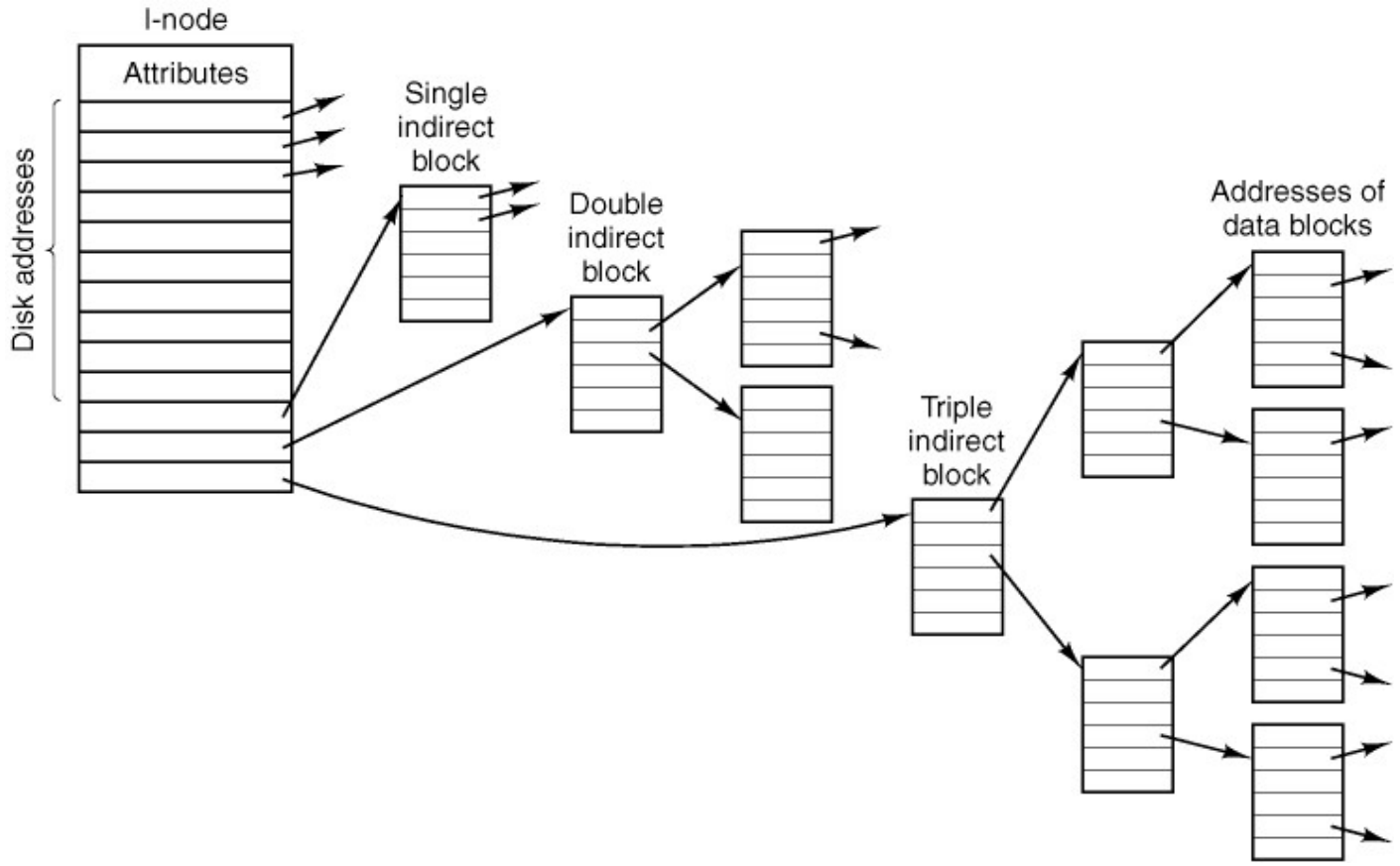
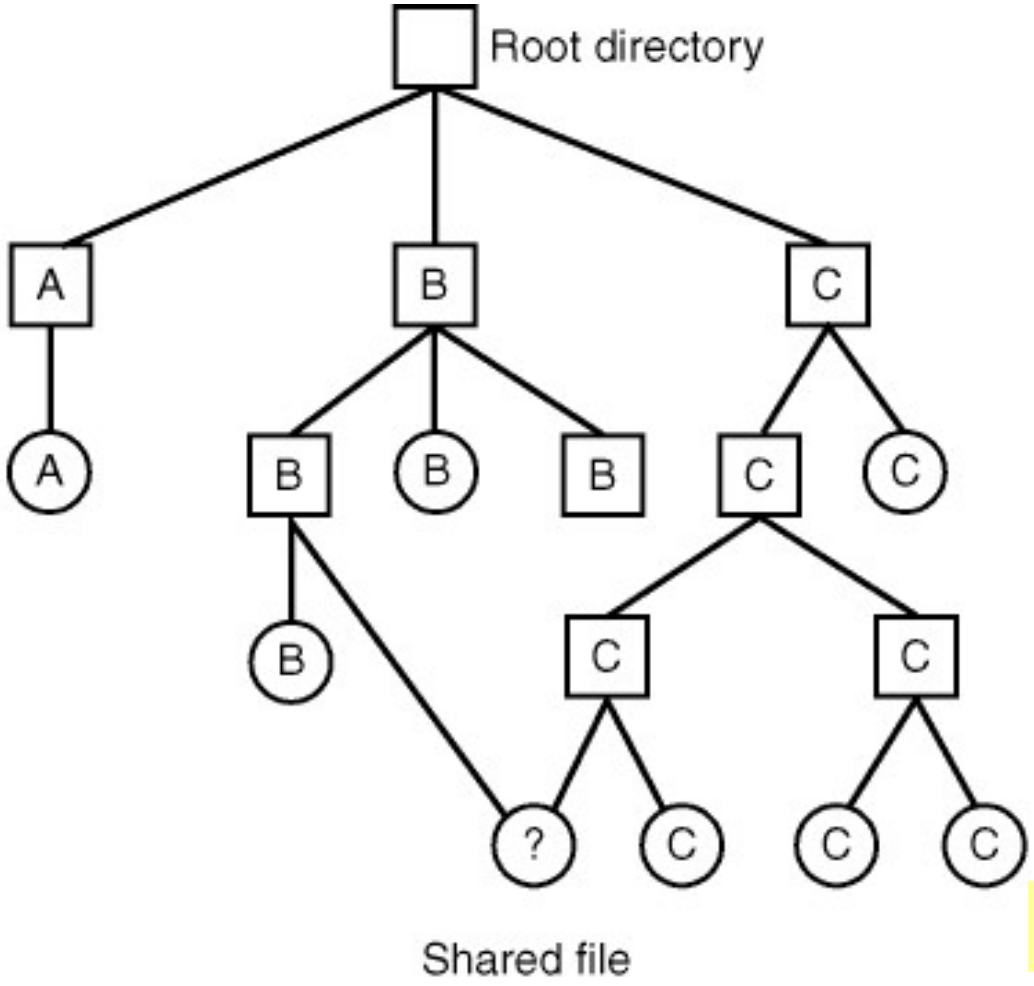


Figure 5-11. An i-node with three levels of indirect blocks.

Shared Files



Snapshots, COW

Figure 5-12. File system containing a shared file.

Directories in Windows 98 (1)

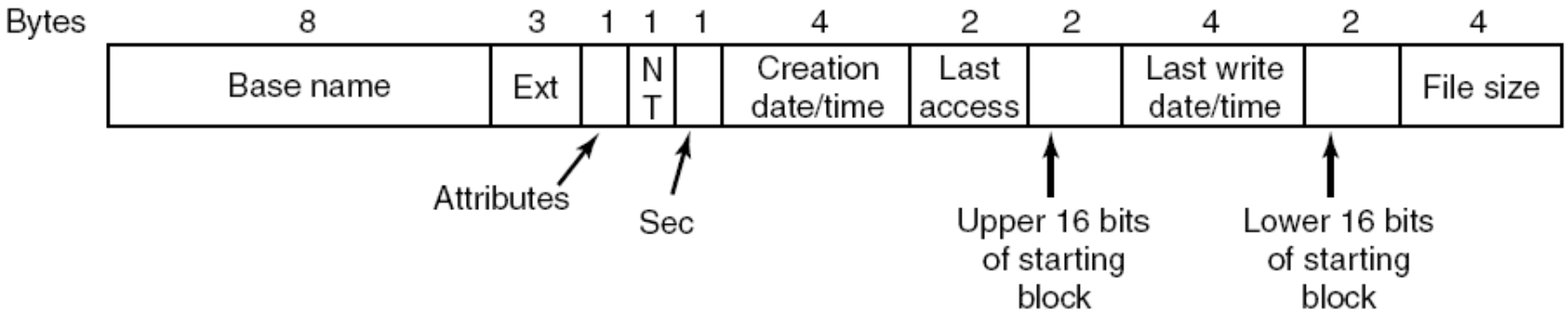


Figure 5-13. A Windows 98 base directory entry.

Directories in Windows 98 (2)

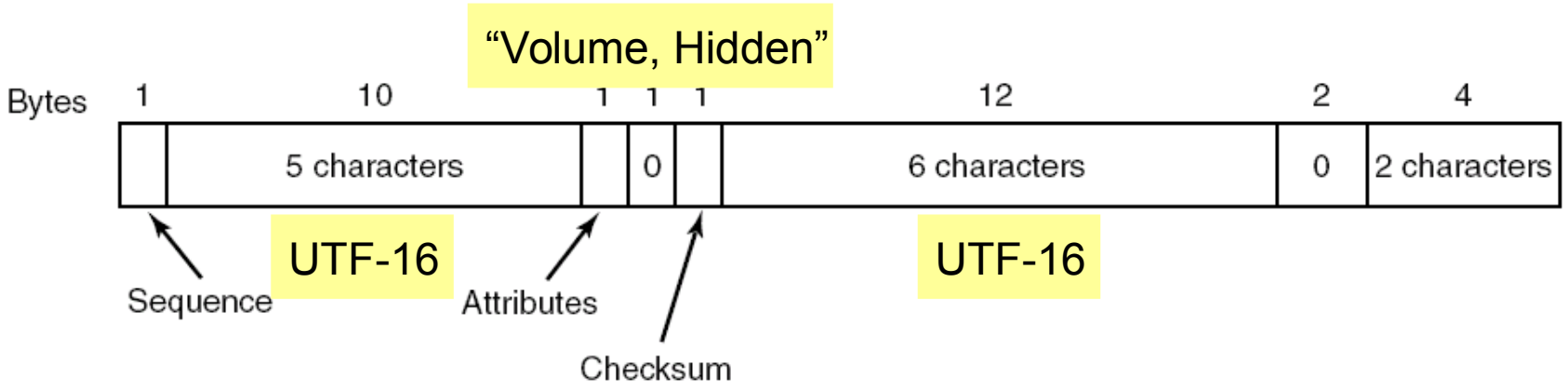


Figure 5-14. An entry for (part of) a long file name in Windows 98.

Directories in UNIX (1)

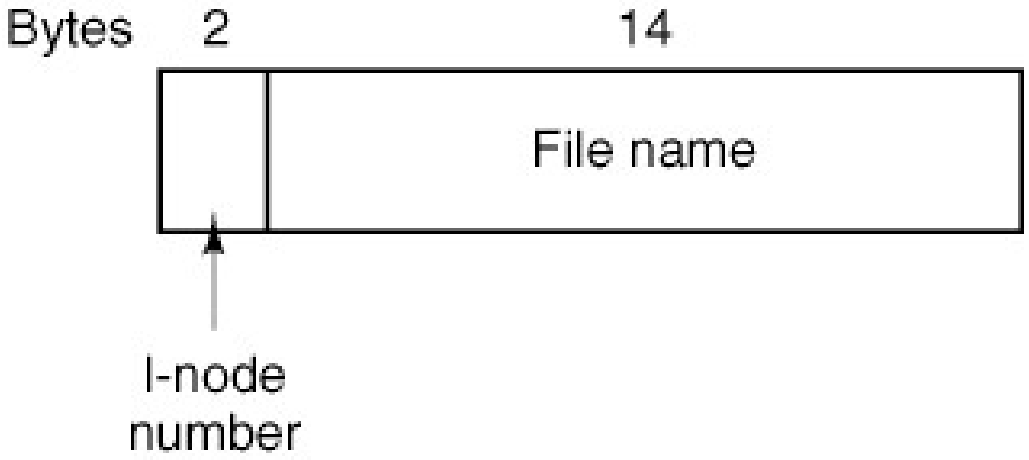


Figure 5-15. A Version 7 UNIX directory entry.

Directories in UNIX (2)

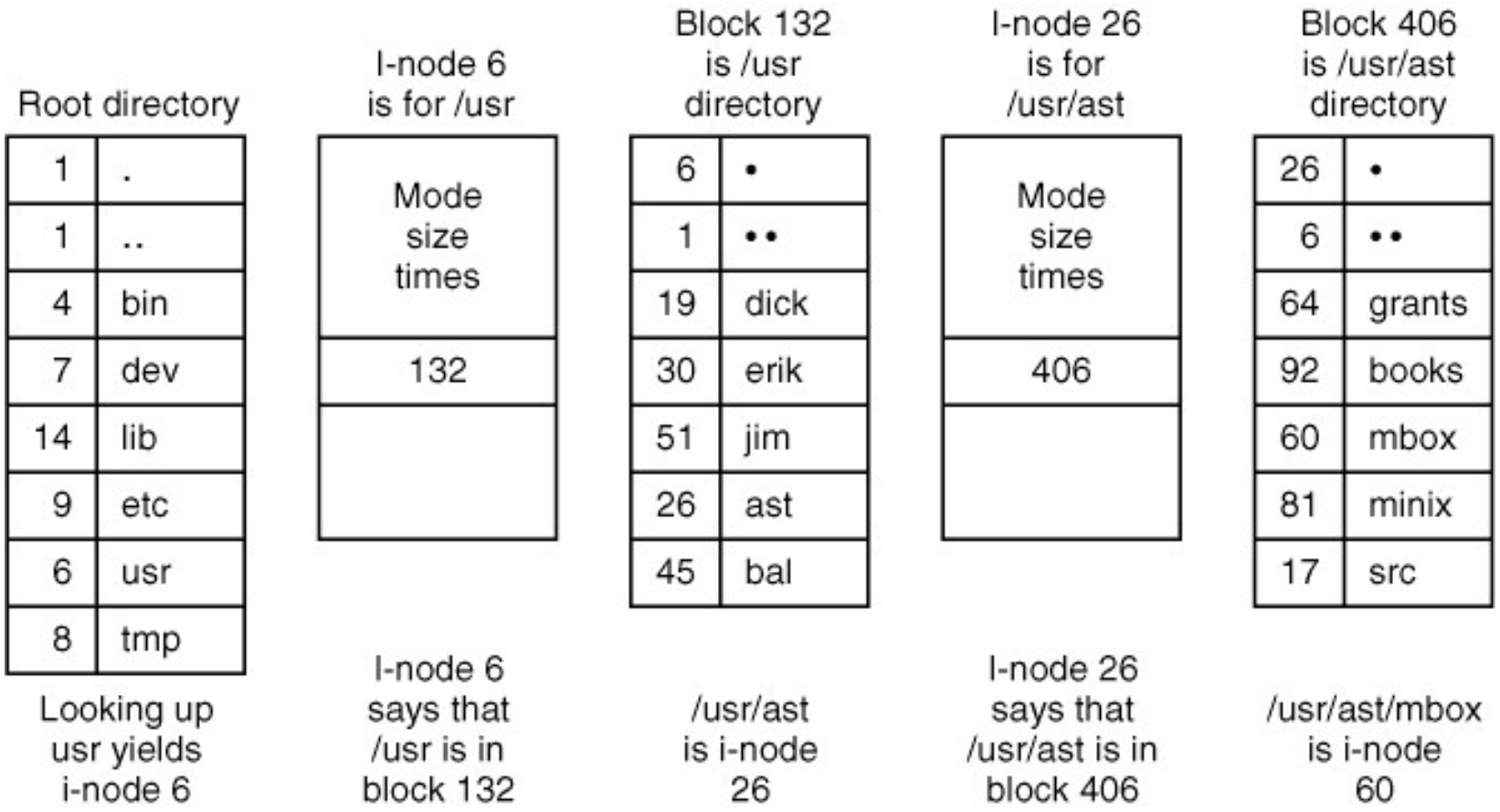


Figure 5-16. The steps in looking up */usr/ast/mbox*.

Block Size

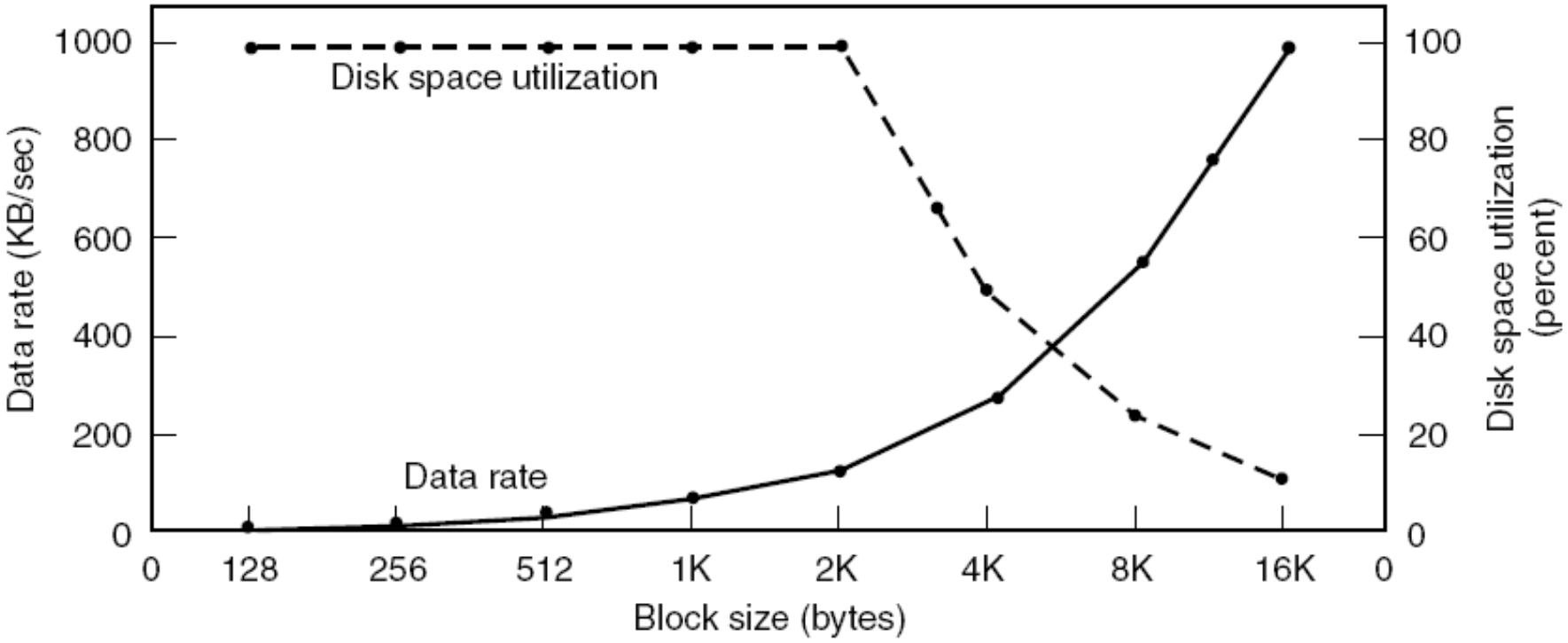
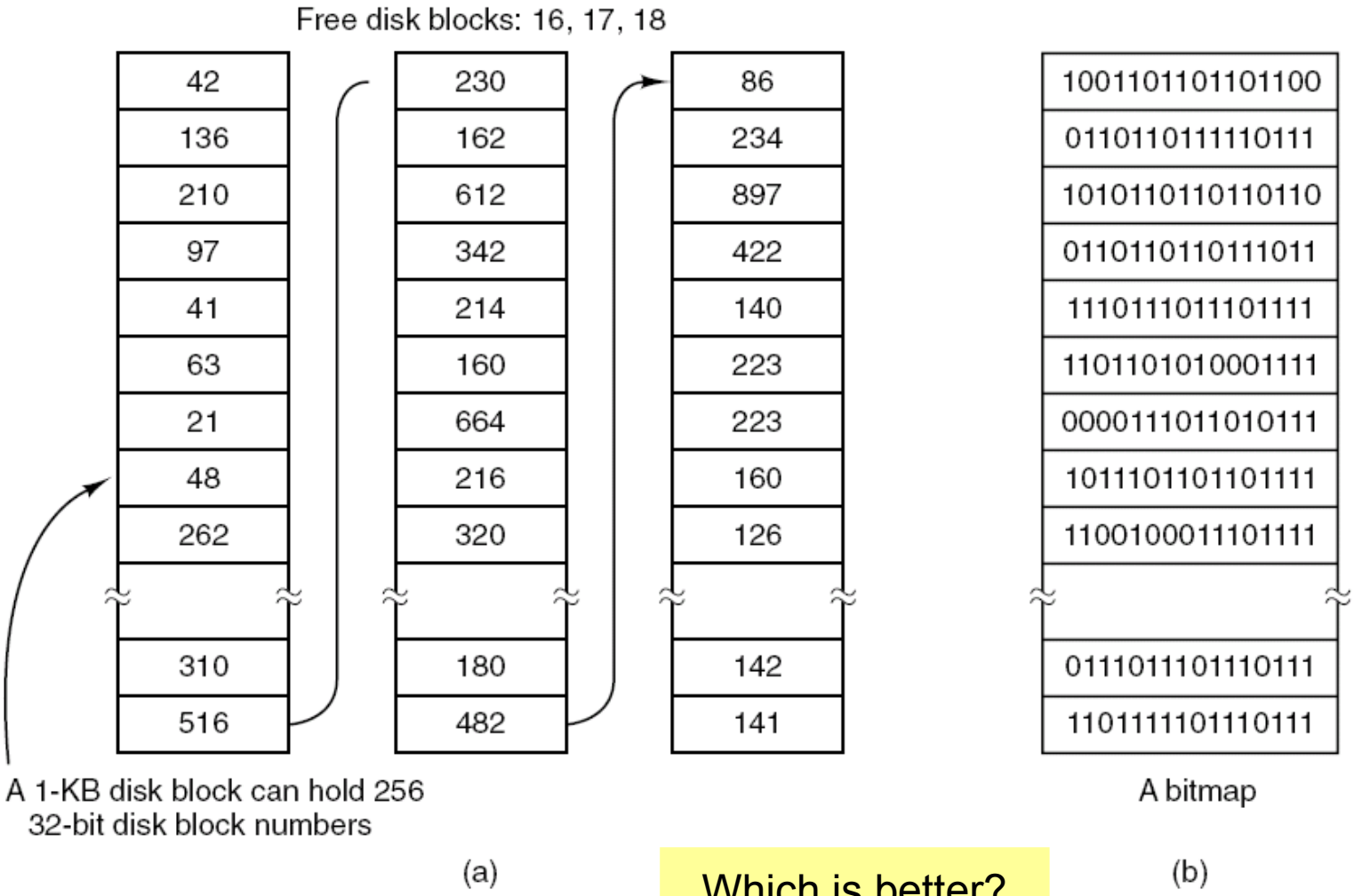


Figure 5-17. The solid curve (left-hand scale) gives the data rate of a disk. The dashed curve (right-hand scale) gives the disk space efficiency. All files are 2 KB.

Keeping Track of Free Blocks



Which is better?

Figure 5-18. (a) Storing the free list on a linked list. (b) A bitmap.

File System Reliability

Potential problems solved by backups:

1. Recover from disaster.
2. Recover from stupidity.

When which?

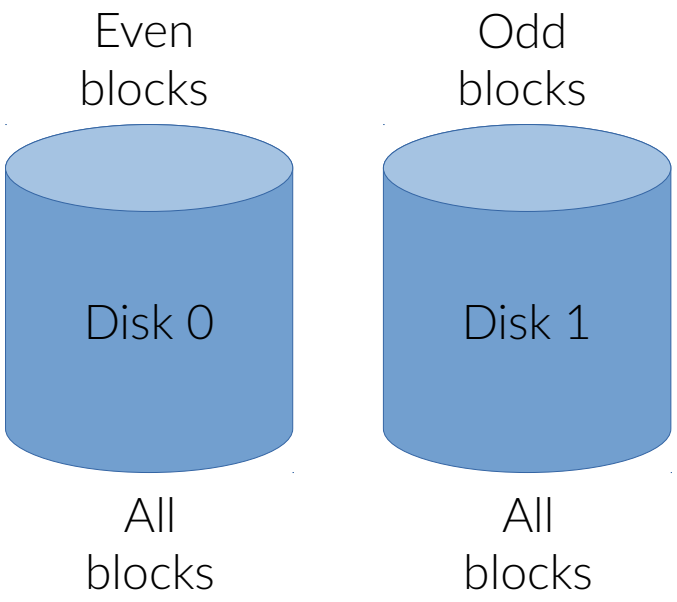
Backup Issues

1. Backup all or part of the system?
2. Don’t backup file if not changed
3. Compression of backup or not?
4. Difficulty of backup while file system active
5. Physical security of backup media
6. File-based or image-based?
7. Incremental or full?

RAID: The Idea

- Reliable, fast disks are expensive
- Cheap disks are unreliable, slow
- “Redundant Array of [Inexpensive| Independent] Disks”
- Use multiple cheap disks to get/exceed the reliability/speed of expensive disks

Simple RAID

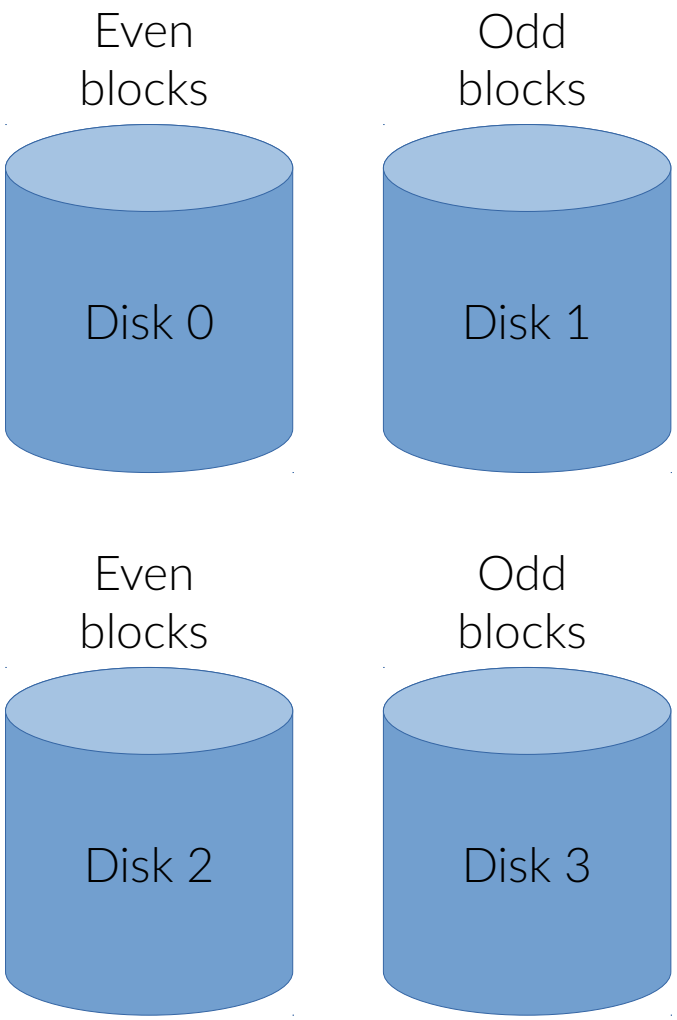


RAID 0

RAID 1

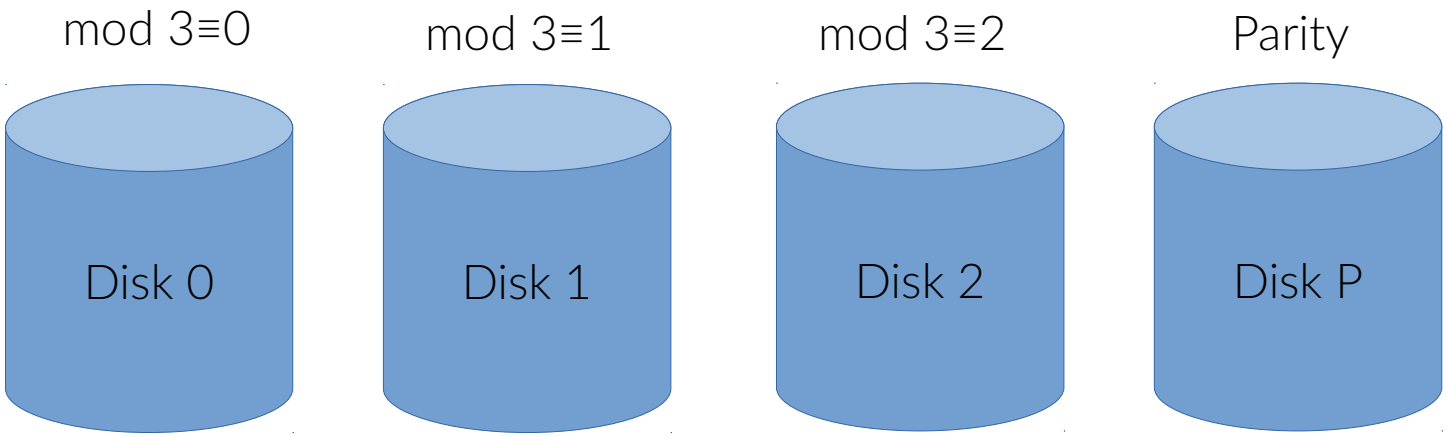
- Capacity
- Read effort/speed
- Write effort/speed
- Outages
- Generalizations

Simple Combination: RAID 10



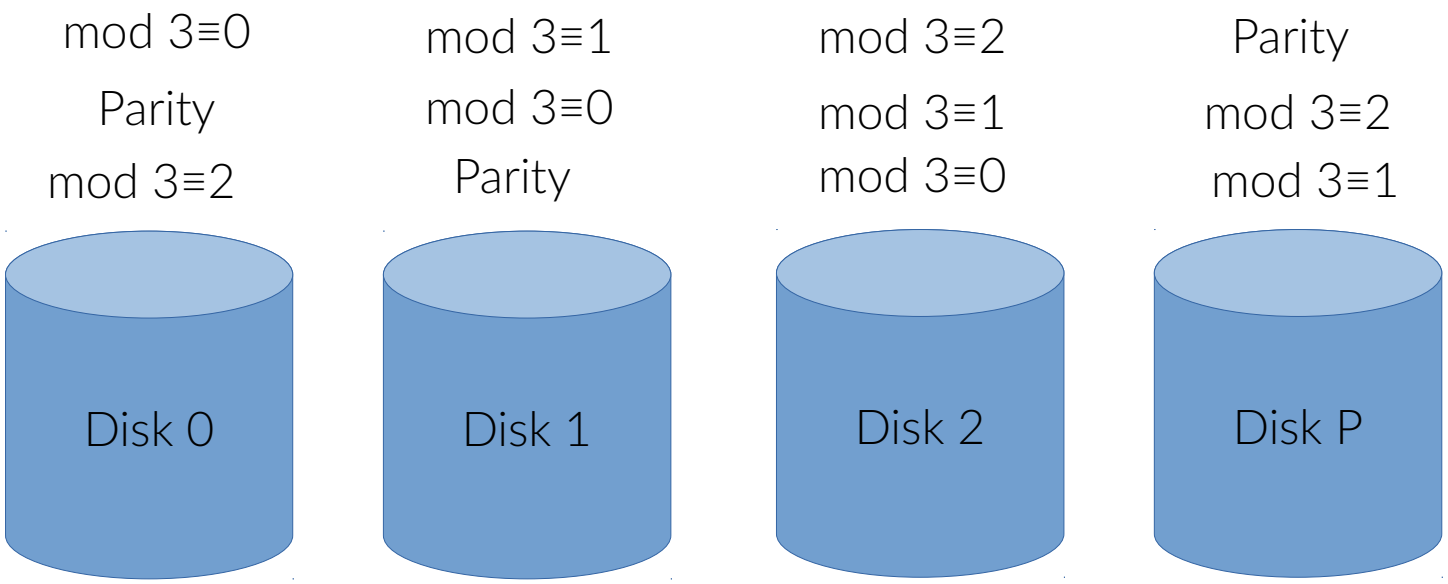
- Capacity
- Read effort/speed
- Write effort/speed
- Outages
- Generalizations
- RAID 1 over RAID 0 or RAID 0 over RAID 1?

Cheaper Redundancy



- Capacity
- Read effort/speed
- Write effort/speed
- Outages
- Generalizations

Faster Redundancy

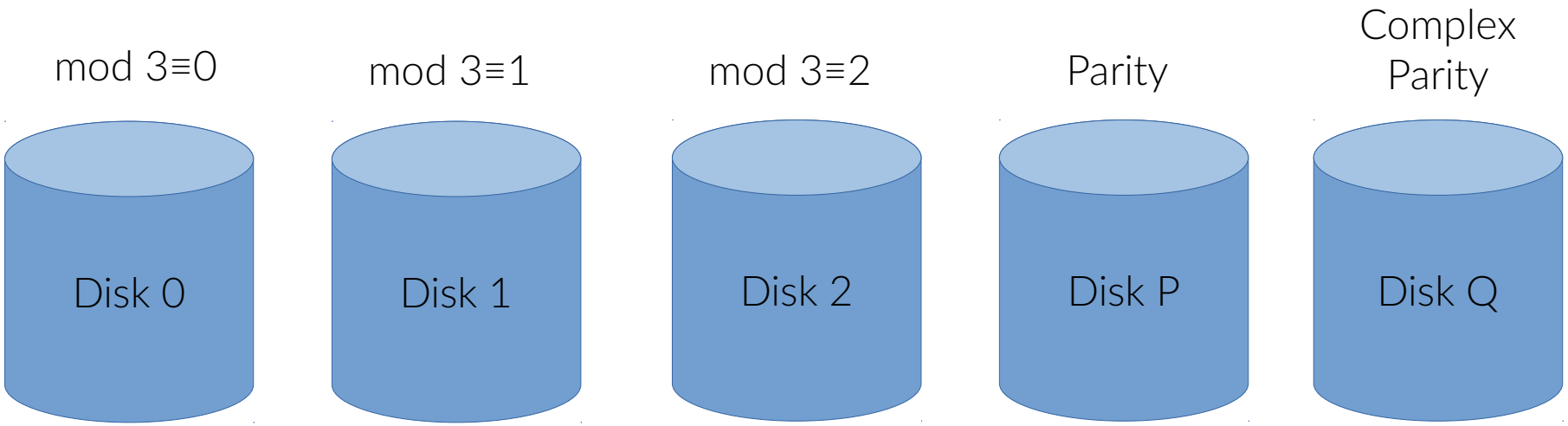


RAID 5

- Capacity
- Read effort/speed
- Write effort/speed
- Outages
- Generalizations

Higher Redundancy

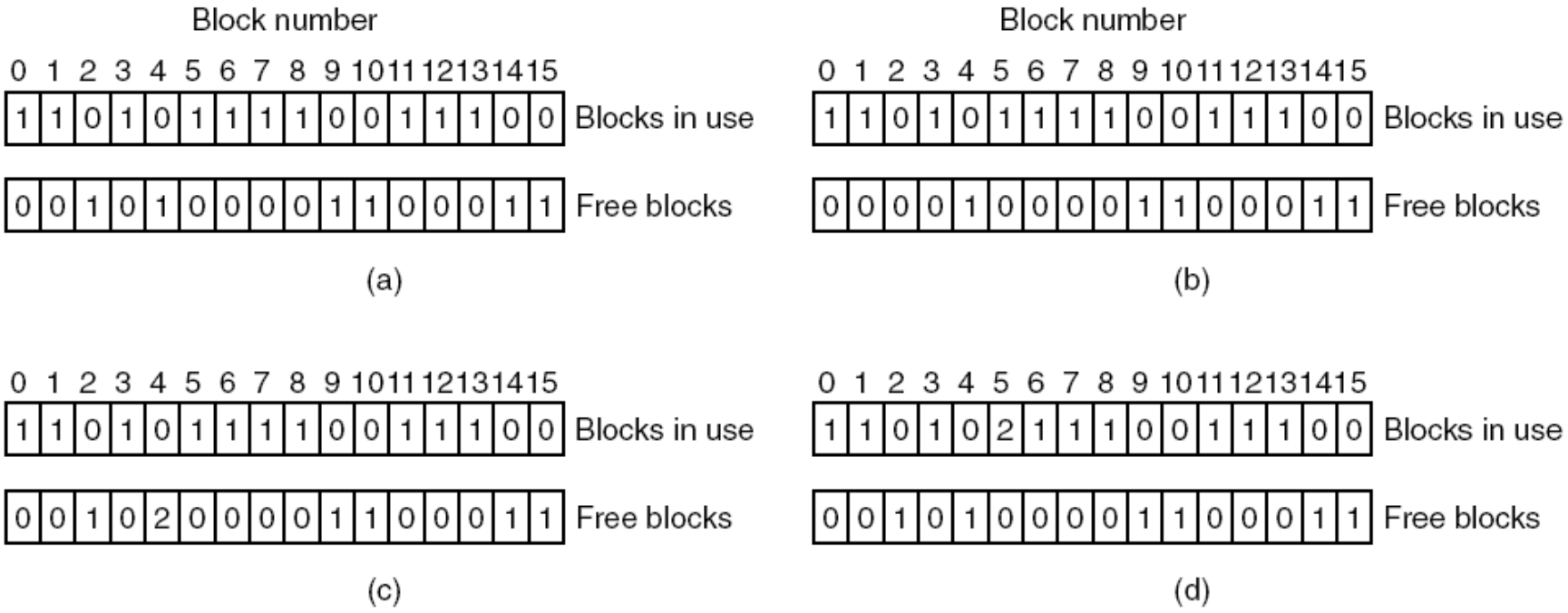
RAID 6



(Roles rotate as in RAID 5)

- Capacity
- Read effort/speed
- Write effort/speed
- Outages
- Generalizations

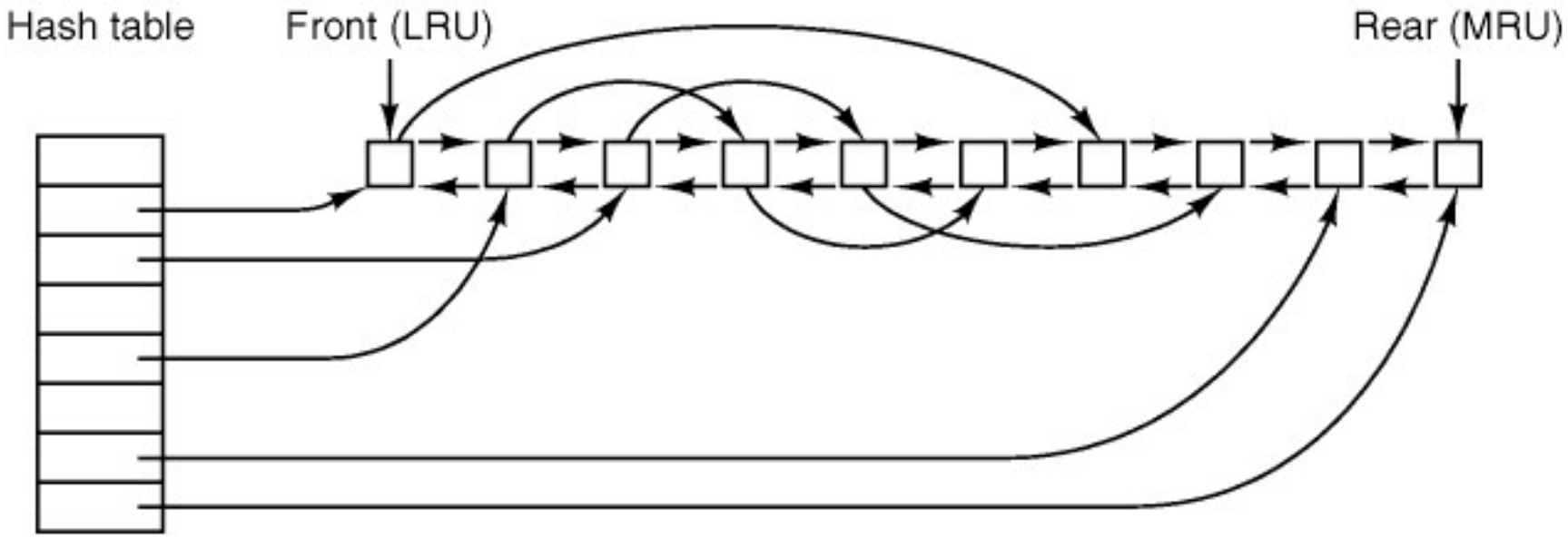
File System Consistency



Countermeasures?

Figure 5-19. File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

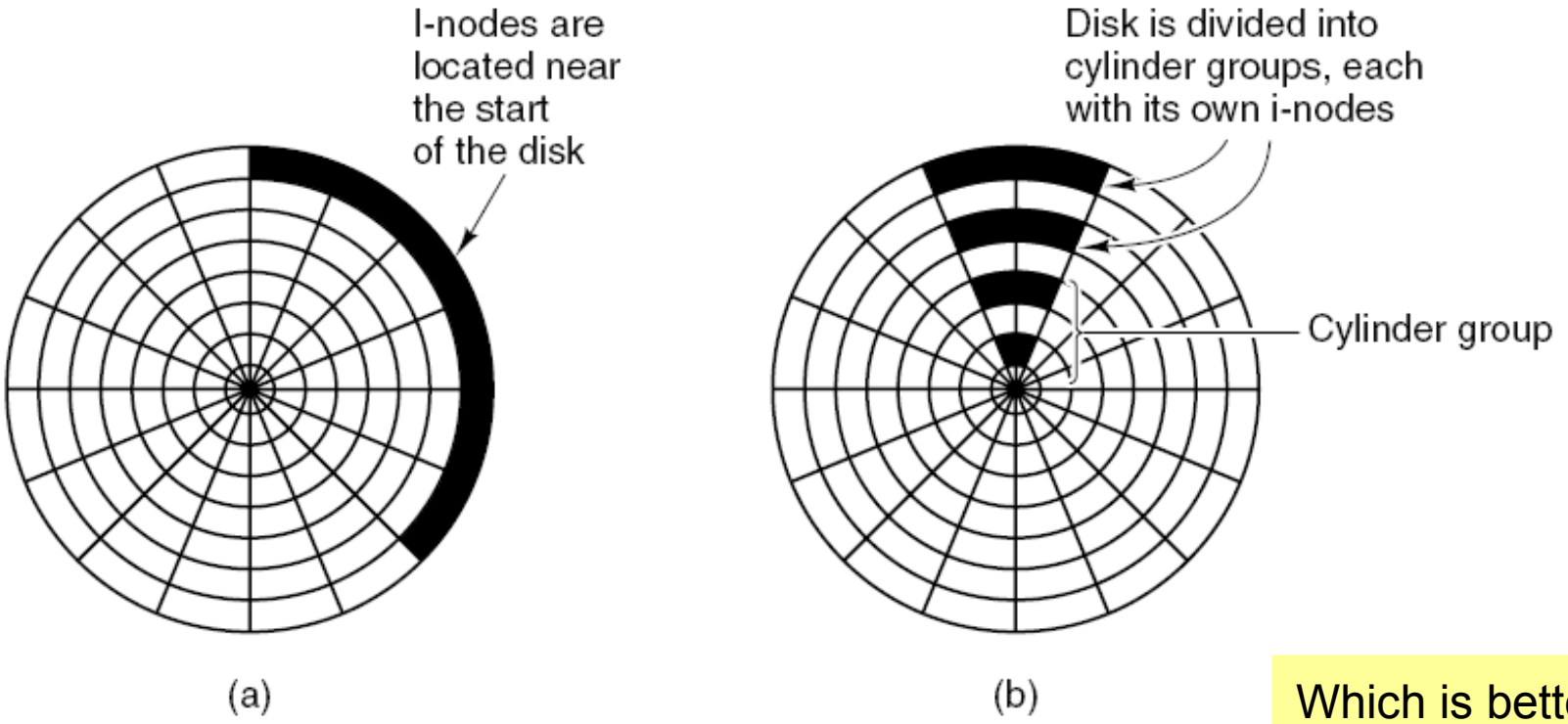
Caching



O(1) operations

Figure 5-20. The buffer cache data structures.

Reducing Disk Arm Motion



Which is better?

Figure 5-21. (a) I-nodes placed at the start of the disk. (b) Disk divided into cylinder groups, each with its own blocks and i-nodes.

MINIX 3 File System (1)

Messages from users	Input parameters	Reply value
access	File name, access mode	Status
chdir	Name of new working directory	Status
chmod	File name, new mode	Status
chown	File name, new owner, group	Status
chroot	Name of new root directory	Status
close	File descriptor of file to close	Status
creat	Name of file to be created, mode	File descriptor
dup	File descriptor (for dup2, two fds)	New file descriptor
fcntl	File descriptor, function code, arg	Depends on function
fstat	Name of file, buffer	Status
ioctl	File descriptor, function code, arg	Status
link	Name of file to link to, name of link	Status
lseek	File descriptor, offset, whence	New position
mkdir	File name, mode	Status
mknod	Name of dir or special, mode, address	Status

. . .

Figure 5-33. File system messages. File name parameters are always pointers to the name. The code status as reply value means *OK* or *ERROR*.

. . . MINIX 3 File System (2)

Messages from users	Input parameters	Reply value
mknod	Name of dir or special, mode, address	Status
mount	Special file, where to mount, ro flag	Status
open	Name of file to open, r/w flag	File descriptor
pipe	Pointer to 2 file descriptors (modified)	Status
read	File descriptor, buffer, how many bytes	# Bytes read
rename	File name, file name	Status
rmdir	File name	Status
stat	File name, status buffer	Status
stime	Pointer to current time	Status
sync	(None)	Always OK
time	Pointer to place where current time goes	Status
times	Pointer to buffer for process and child times	Status
umask	Complement of mode mask	Always OK
umount	Name of special file to unmount	Status
unlink	Name of file to unlink	Status
utime	File name, file times	Always OK
write	File descriptor, buffer, how many bytes	# Bytes written

Figure 5-33. File system messages. File name parameters are always pointers to the name. The code status as reply value means *OK* or *ERROR*.

MINIX 3 File System (3)

. . .

Messages from PM	Input parameters	Reply value
exec	Pid	Status
exit	Pid	Status
fork	Parent pid, child pid	Status
setgid	Pid, real and effective gid	Status
setsid	Pid	Status
setuid	Pid, real and effective uid	Status
Other messages	**Input parameters**	**Reply value**
revive	Process to revive	(No reply)
unpause	Process to check	(See text)

Figure 5-33. File system messages. File name parameters are always pointers to the name. The code status as reply value means *OK* or *ERROR*.

File System Layout (1)

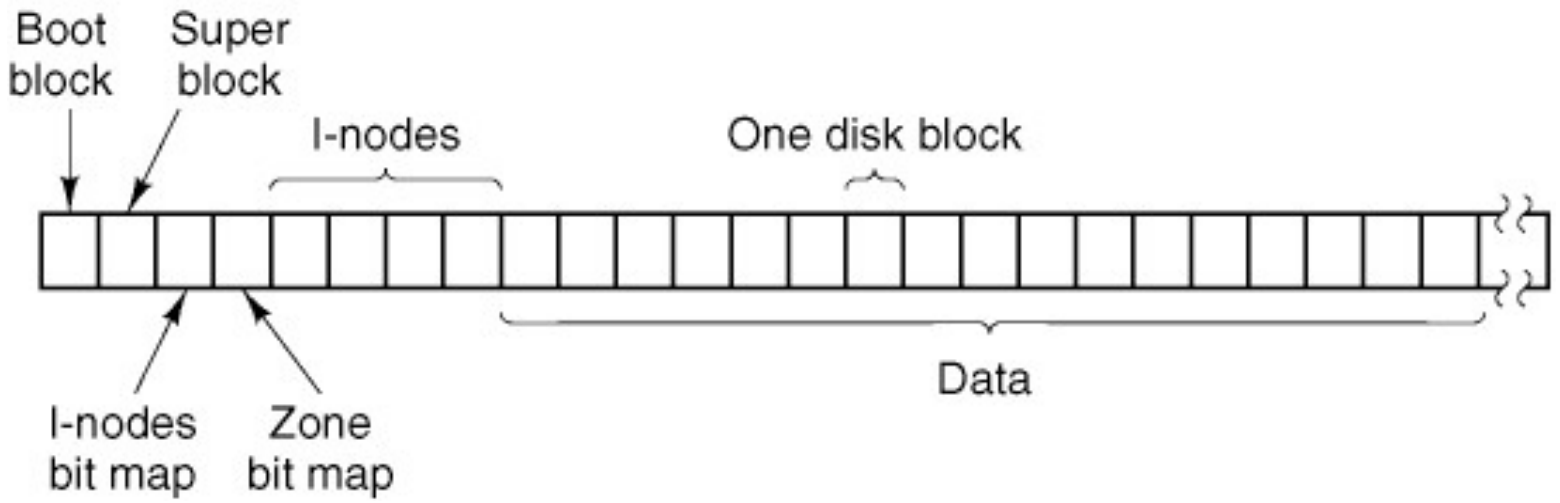


Figure 5-34. Disk layout for a floppy disk or small hard disk partition, with 64 i-nodes and a 1-KB block size (i.e., two consecutive 512-byte sectors are treated as a single block).

File System Layout (2)

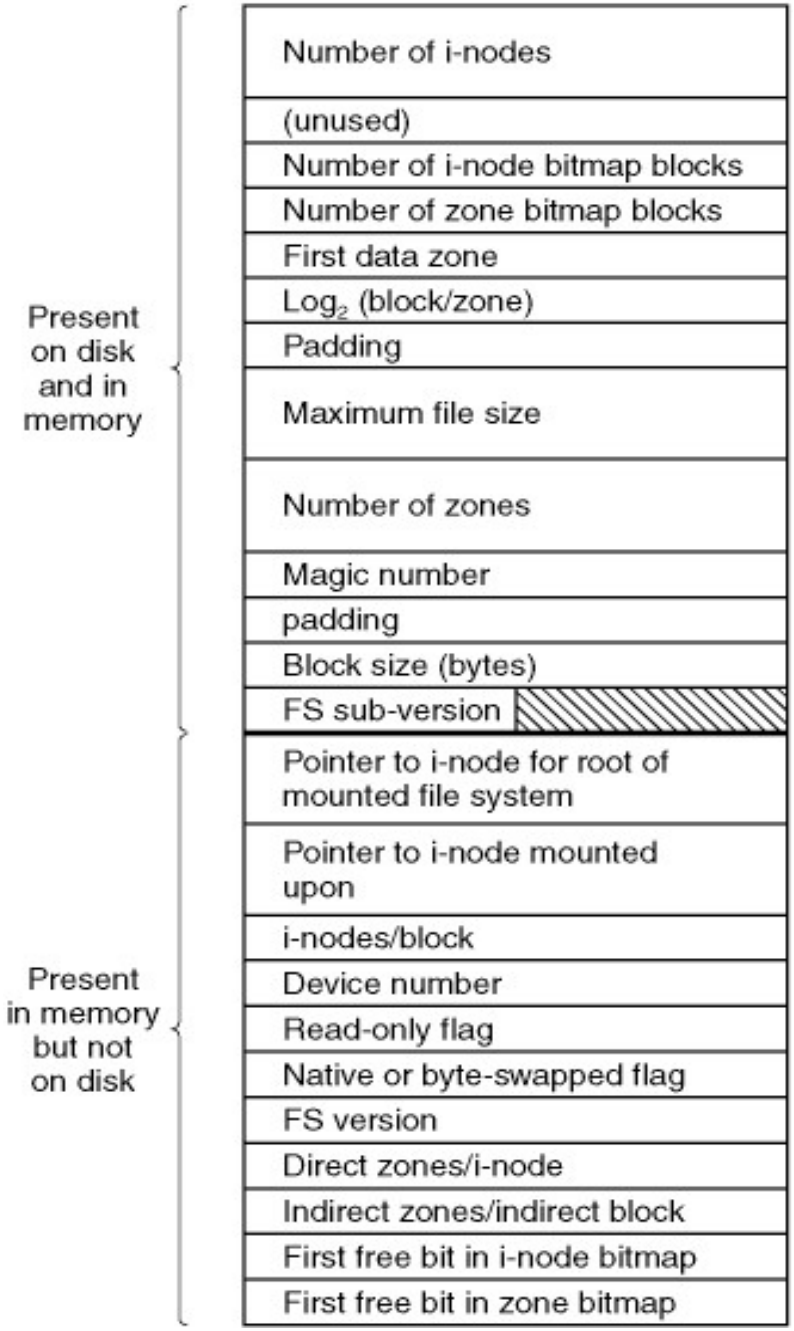


Figure 5-35. The MINIX 3 superblock.

I-Nodes

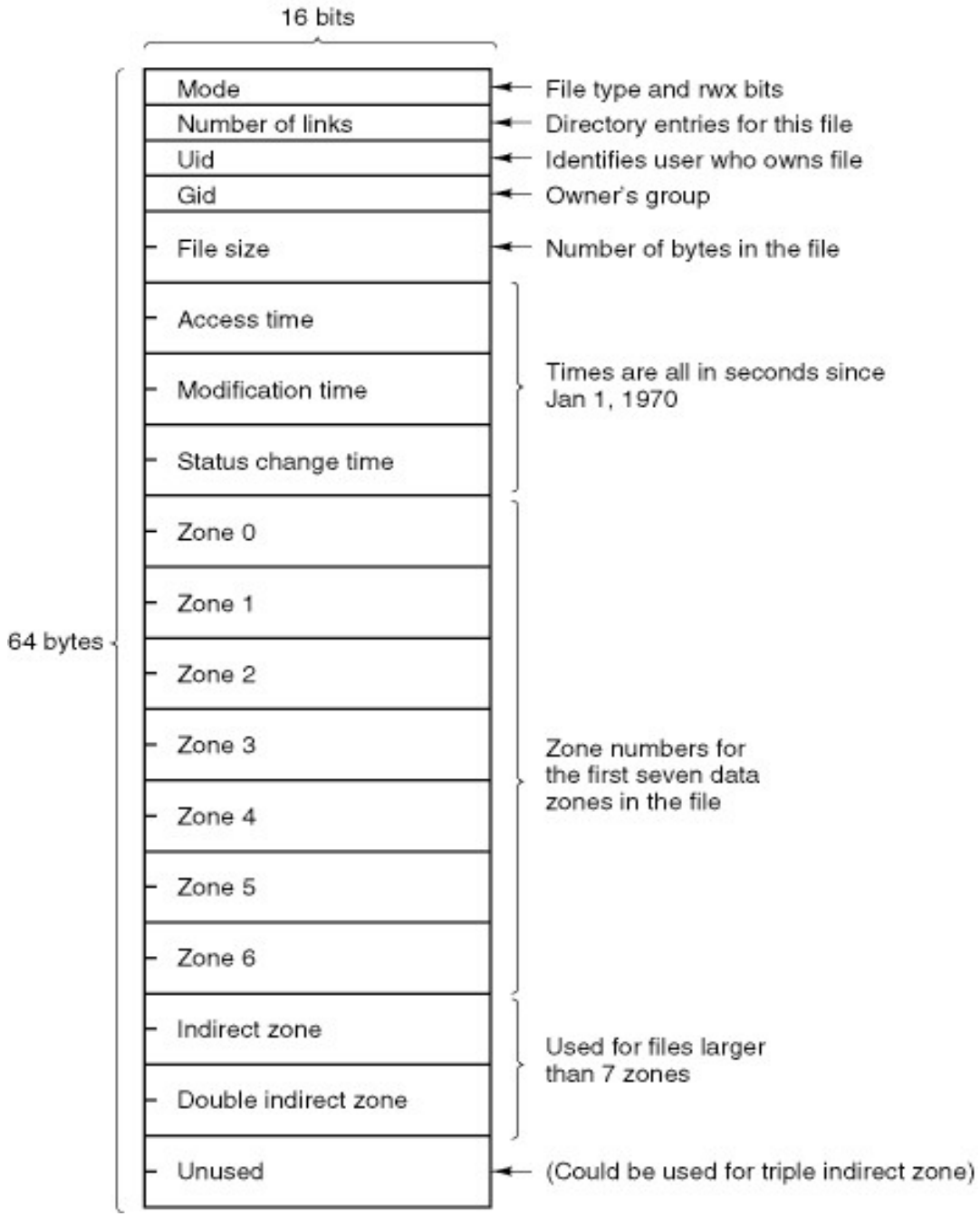


Figure 5-36. The MINIX i-node.

The Block Cache

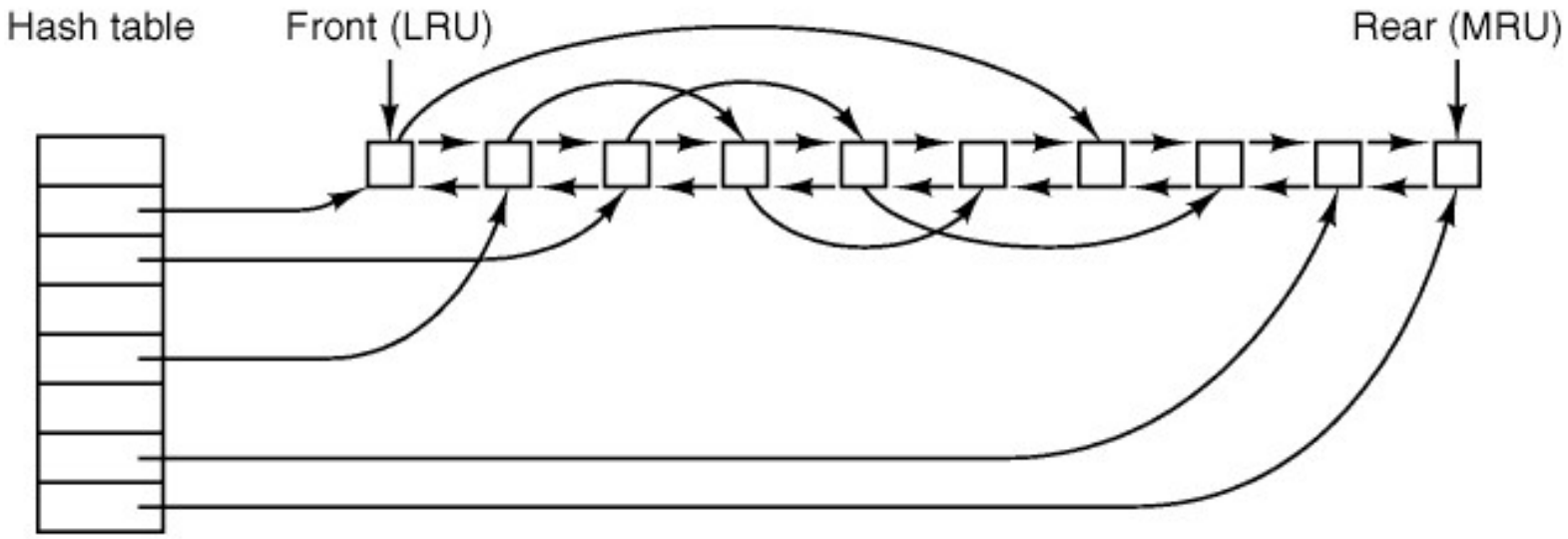


Figure 5-37. The linked lists used by the block cache.

Directories and Paths

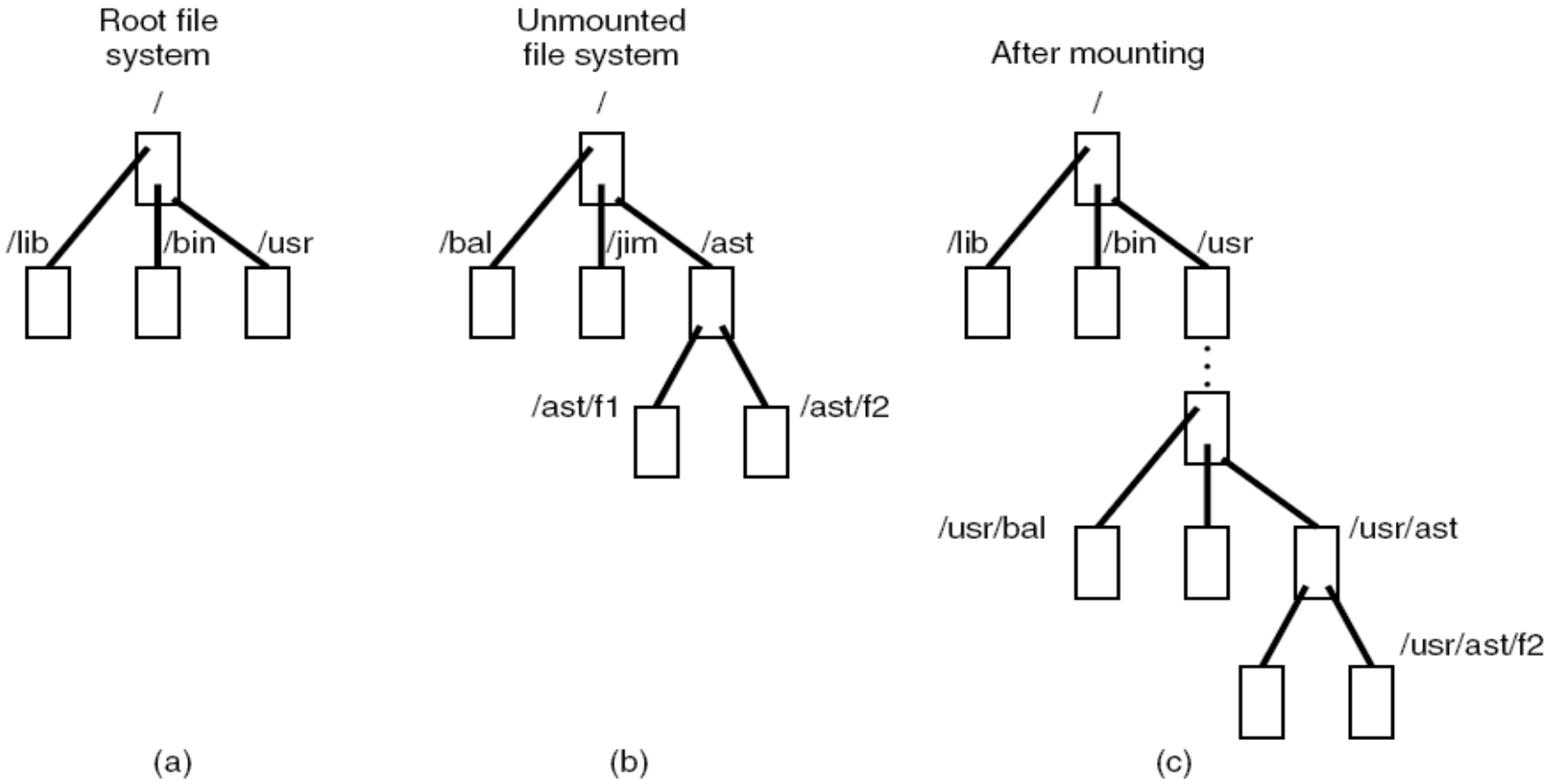


Figure 5-38. (a) Root file system. (b) An unmounted file system. (c) The result of mounting the file system of (b) on */usr/.*

File Descriptors

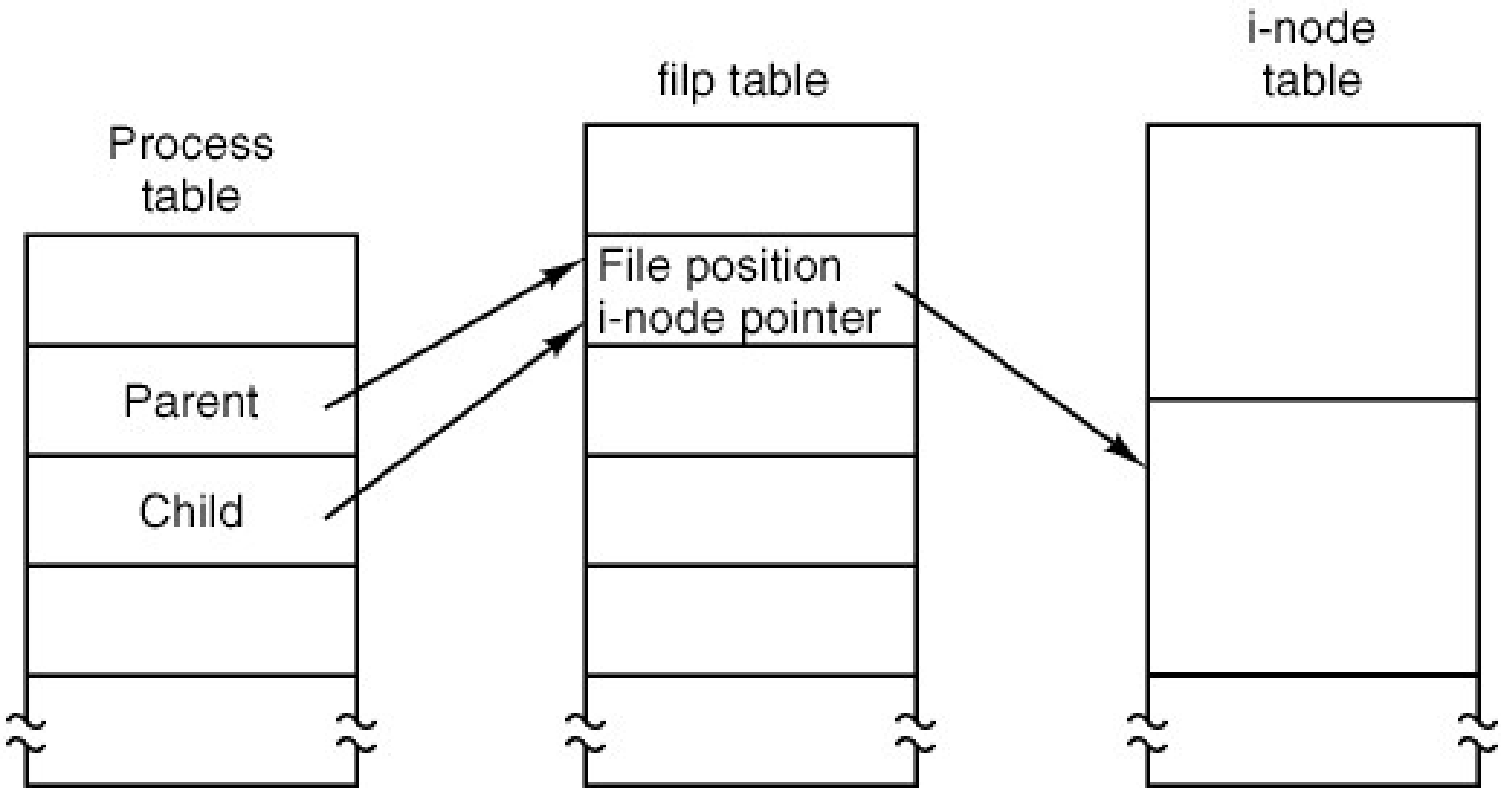


Figure 5-39. How file positions are shared between a parent and a child.

Block Management

Procedure	Function
get_block	Fetch a block for reading or writing
put_block	Return a block previously requested with get_block
alloc_zone	Allocate a new zone (to make a file longer)
free_zone	Release a zone (when a file is removed)
rw_block	Transfer a block between disk and cache
invalidate	Purge all the cache blocks for some device
flushall	Flush all dirty blocks for one device
rw_scattered	Read or write scattered data from or to a device
rm_lru	Remove a block from its LRU chain

Figure 5-40. Procedures used for block management.

I-Node Management

Procedure	Function
get_inode	Fetch an i-node into memory
put_inode	Return an i-node that is no longer needed
alloc_inode	Allocate a new i-node (for a new file)
wipe_inode	Clear some fields in an i-node
free_inode	Release an i-node (when a file is removed)
update_times	Update time fields in an i-node
rw_inode	Transfer an i-node between memory and disk
old_icopy	Convert i-node contents to write to V1 disk i-node
new_icopy	Convert data read from V1 file system disk i-node
dup_inode	Indicate that someone else is using an i-node

Figure 5-41. Procedures used for i-node management.

Superblock Management

Procedure	Function
alloc_bit	Allocate a bit from the zone or i-node map
free_bit	Free a bit in the zone or i-node map
get_super	Search the superblock table for a device
get_block_size	Find block size to use
mounted	Report whether given i-node is on a mounted (or root) file system
read_super	Read a superblock

Figure 5-42. Procedures used to manage the superblock and bitmaps.

File Locking

Operation	Meaning
F_SETLK	Lock region for both reading and writing
F_SETLKW	Lock region for writing
F_GETLK	Report if region is locked

Figure 5-43. The POSIX advisory record locking operations. These operations are requested by using an FCNTL system call.

Converting a Path to an I-Node

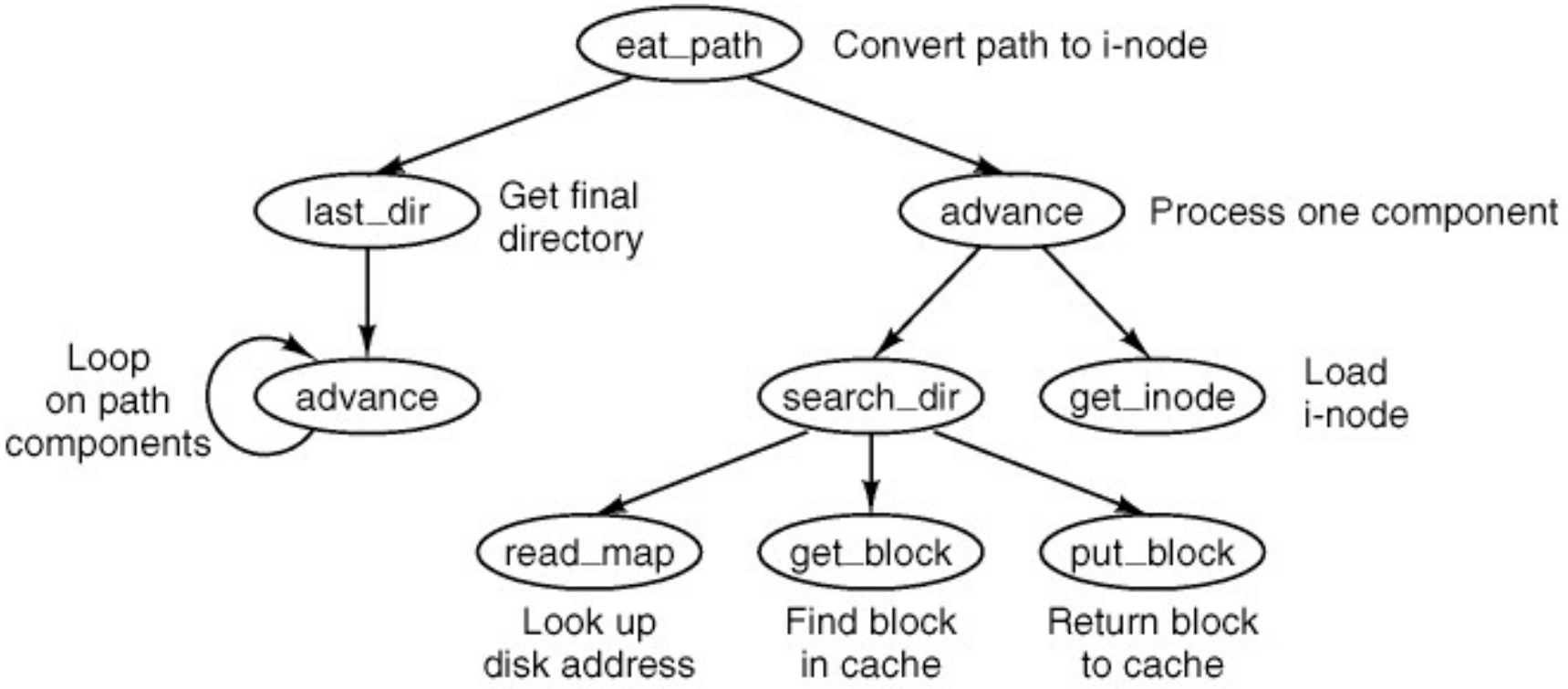


Figure 5-48. Some of the procedures used in looking up path names.

Mounting File Systems

Possible file system mounting errors:

- The special file given is not a block device.
- The special file is a block device but is already mounted.
- The file system to be mounted has a rotten magic number.
- The file system to be mounted is invalid (e.g., no i-nodes).
- The file to be mounted on does not exist or is a special file.
- There is no room for the mounted file system's bitmaps.
- There is no room for the mounted file system's superblock.
- There is no room for the mounted file system's root i-node.

Linking and Unlinking Files

Possible errors in a linking or unlinking call:

- *File_name* does not exist or cannot be accessed.
- *File_name* already has the maximum number of links.
- *File_name* is a directory (only superuser can link to it).
- *Link_name* already exists.
- *File_name* and link "name are on different devices.

Additional System Call Support

Operation	Meaning
F_DUPFD	Duplicate a file descriptor
F_GETFD	Get the close-on-exec flag
F_SETFD	Set the close-on-exec flag
F_GETFL	Get file status flags
F_SETFL	Set file status flags
F_GETLK	Get lock status of a file
F_SETLK	Set read/write lock on a file
F_SETLKW	Set write lock on a file

Figure 5-49. The POSIX request parameters for the FCNTL system call.