5

FILE SYSTEMS

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.f77	Fortran 77 program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
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.

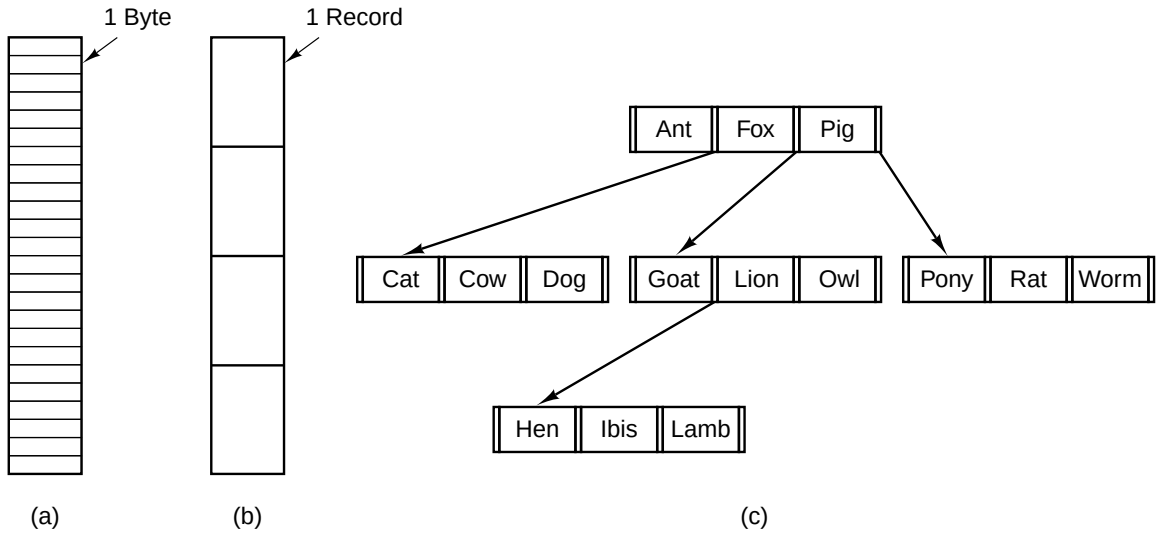

Figure 5-2. Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

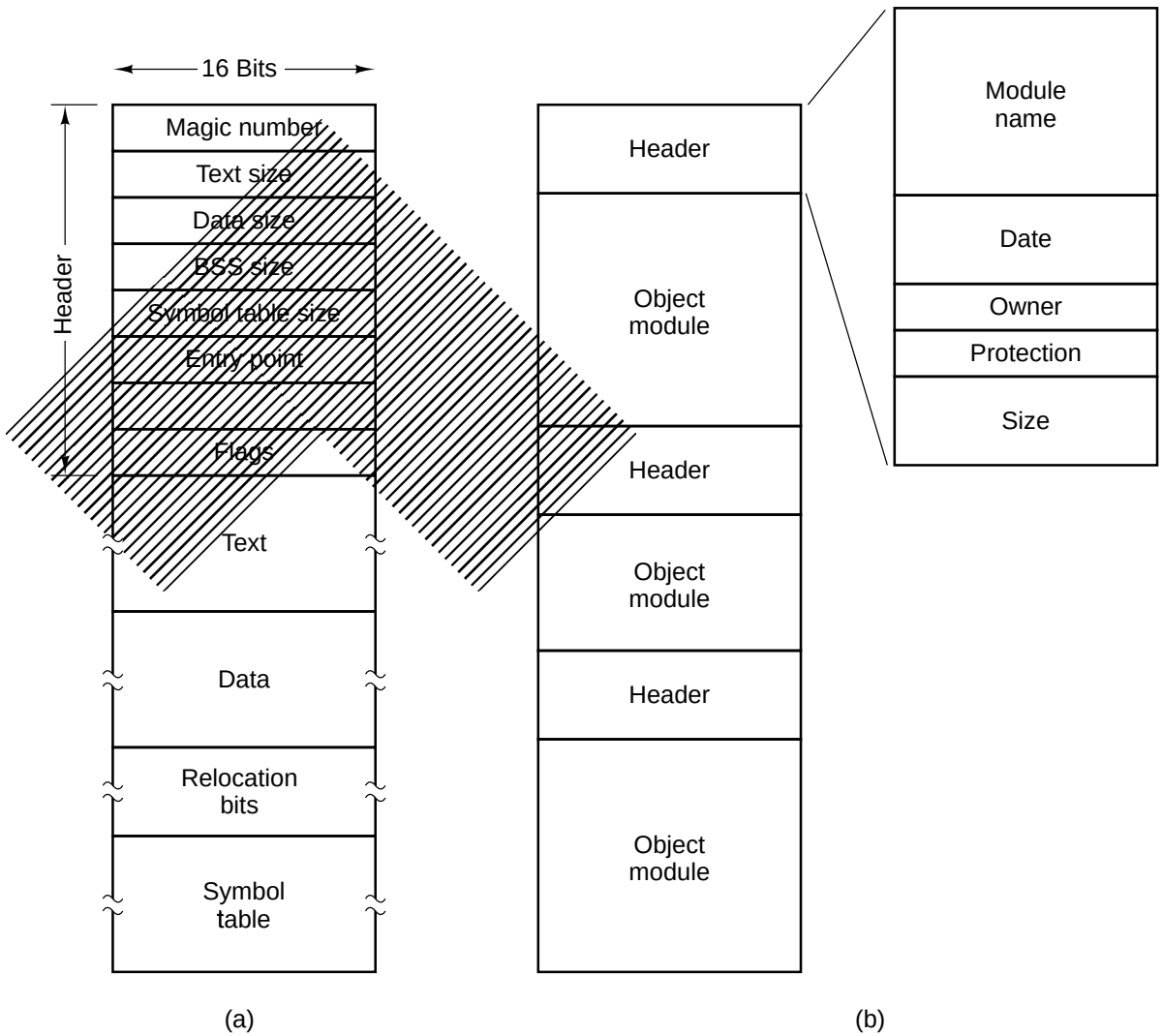

Figure 5-3. (a) An executable file. (b) An archive.

Field	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

Figure 5-4. Some possible file attributes.

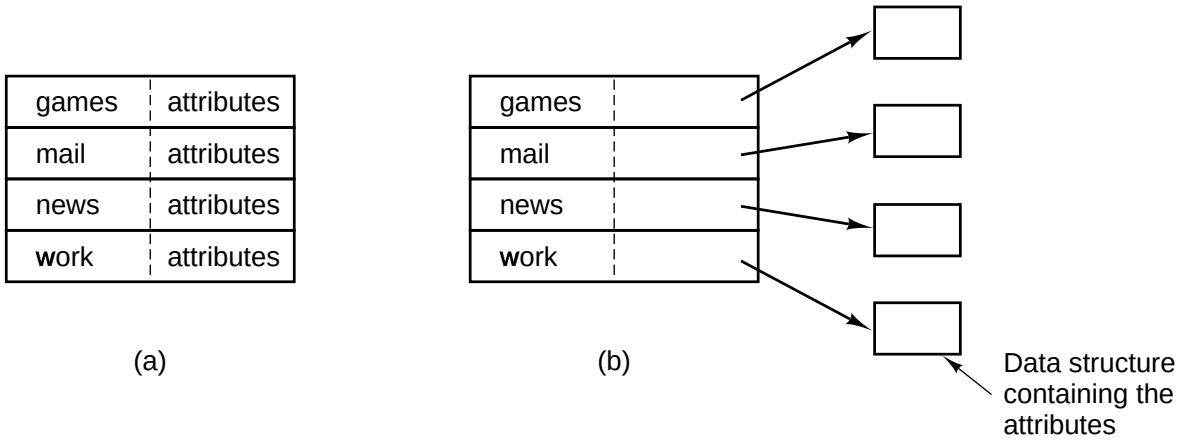

Figure 5-5. (a) Attributes in the directory entry. (b) Attributes elsewhere.

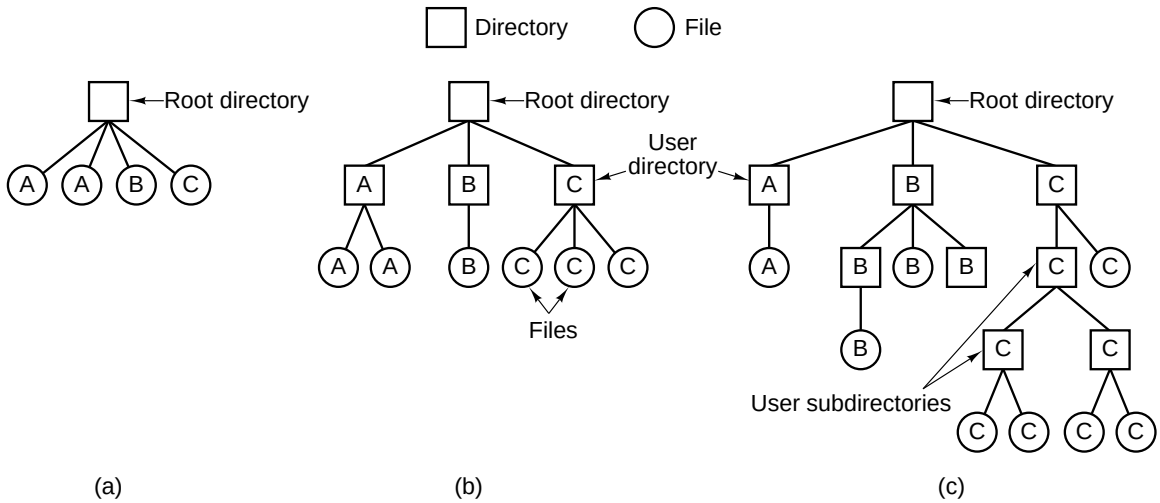

Figure 5-6. Three file system designs. (a) Single directory shared by all users. (b) One directory per user. (c) Arbitrary tree per user. The letters indicate the directory or file's owner.

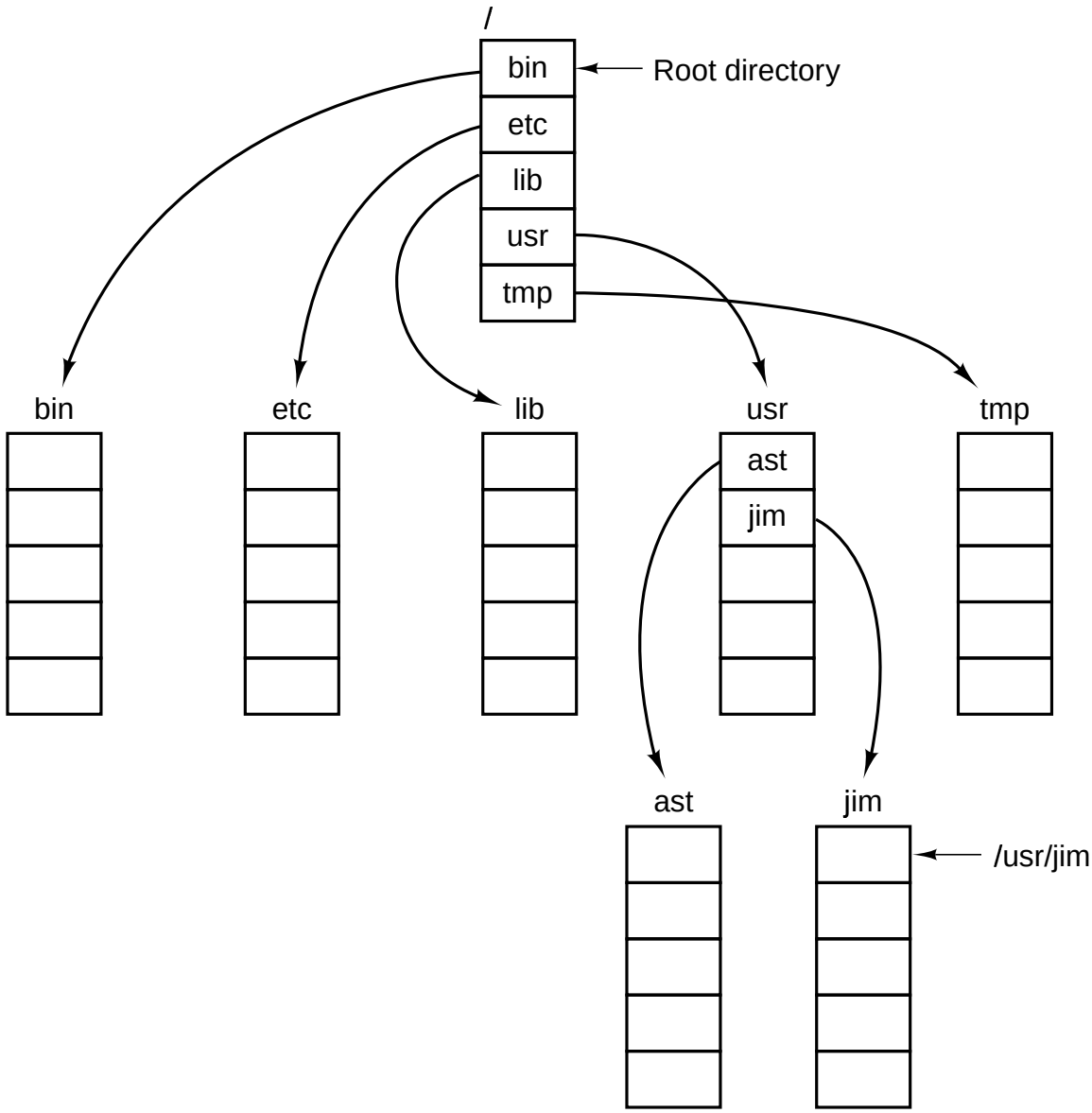

Figure 5-7. A UNIX directory tree.

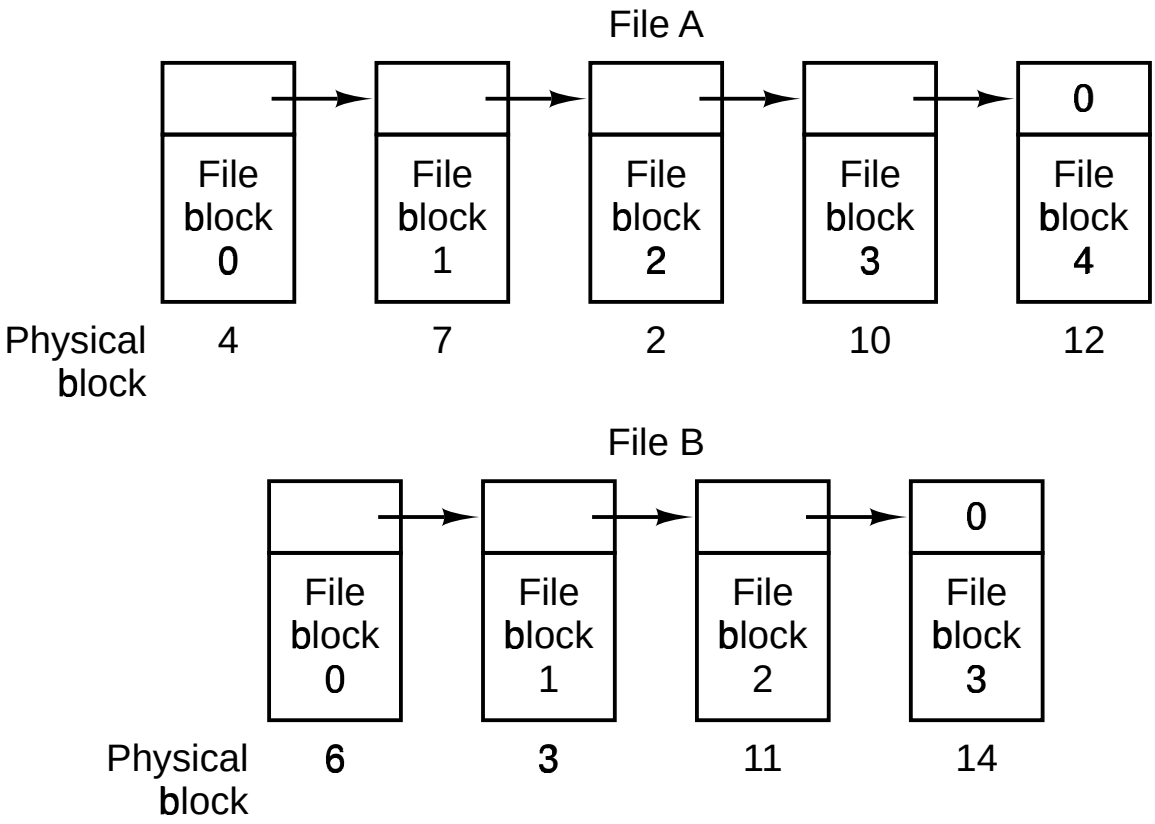

Figure 5-8. Storing a file as a linked list of disk blocks.

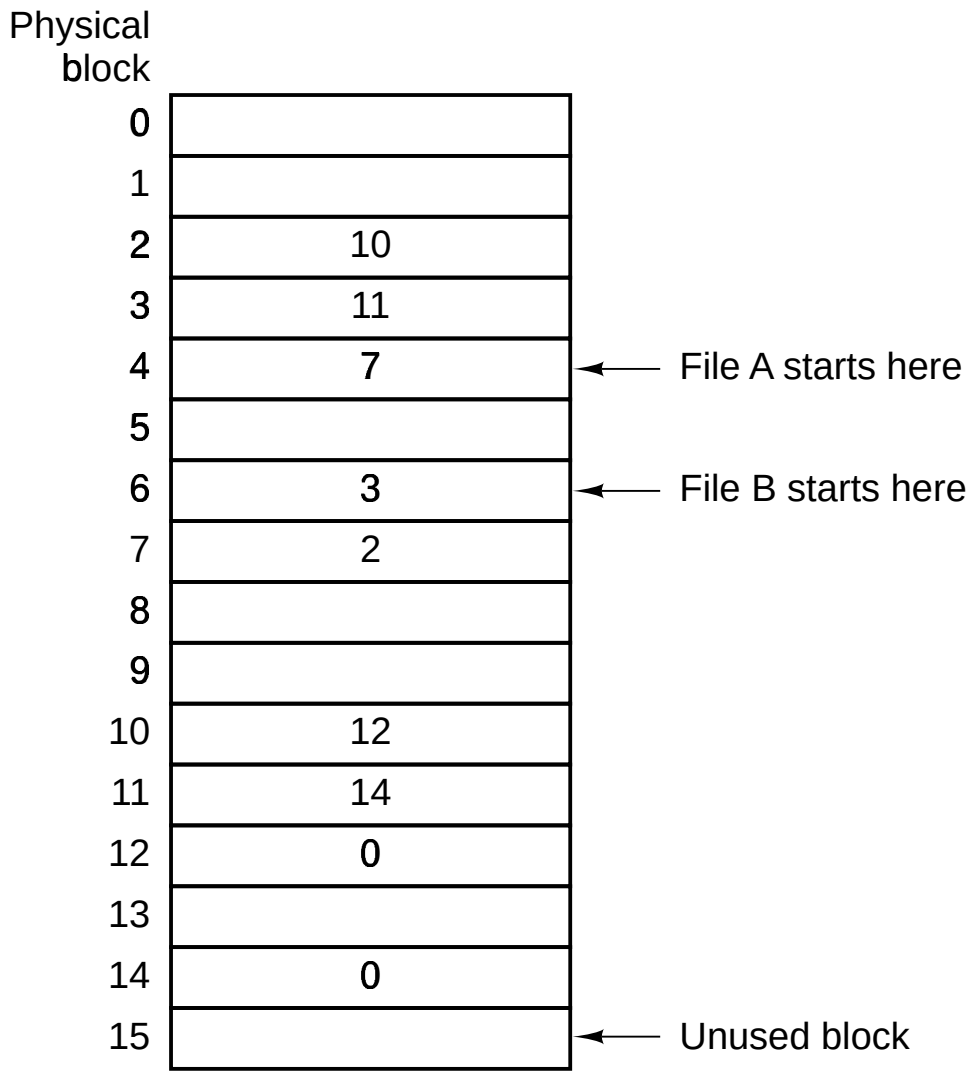

Figure 5-9. Linked list allocation using a table in main memory.

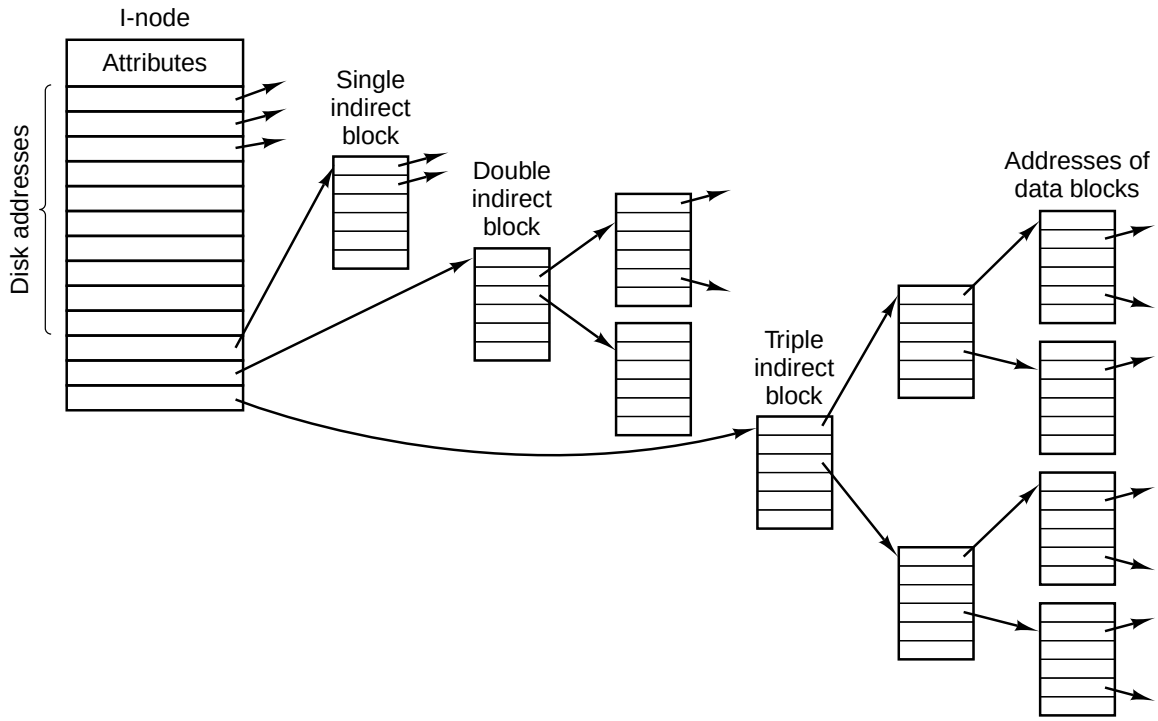

Figure 5-10. An i-node.

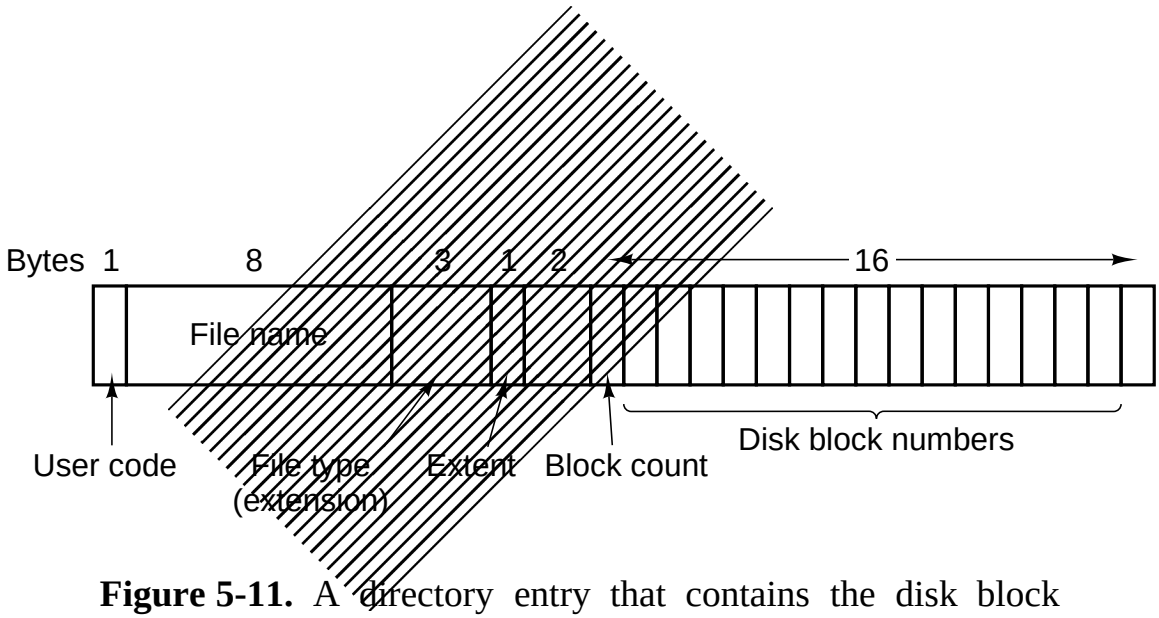

Figure 5-11. A directory entry that contains the disk block numbers for each file.

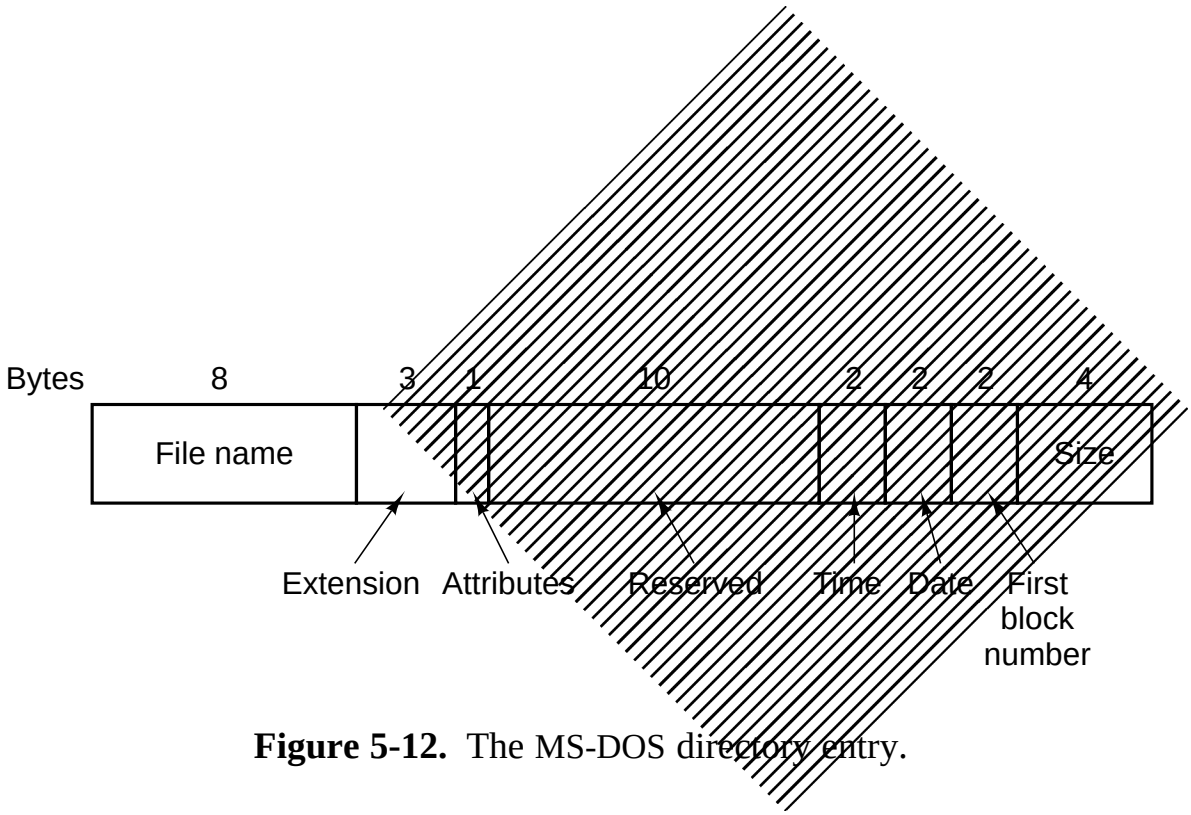

Figure 5-12. The MS-DOS directory entry.

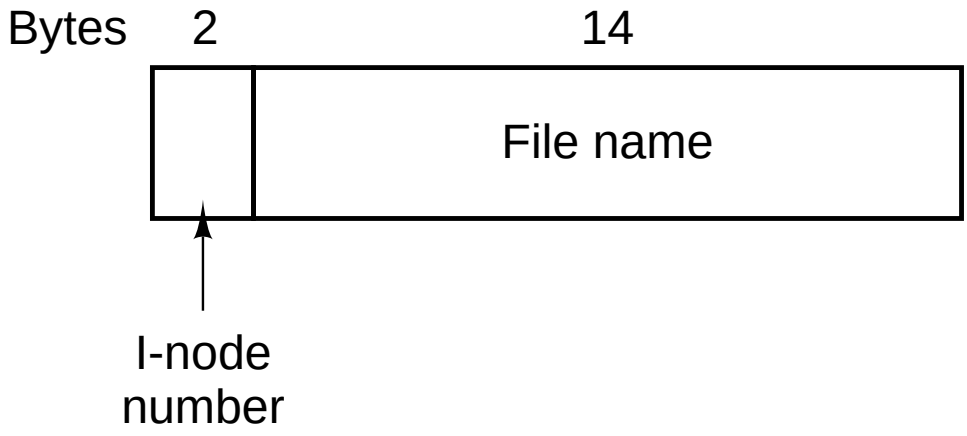

Figure 5-13. A UNIX directory entry.

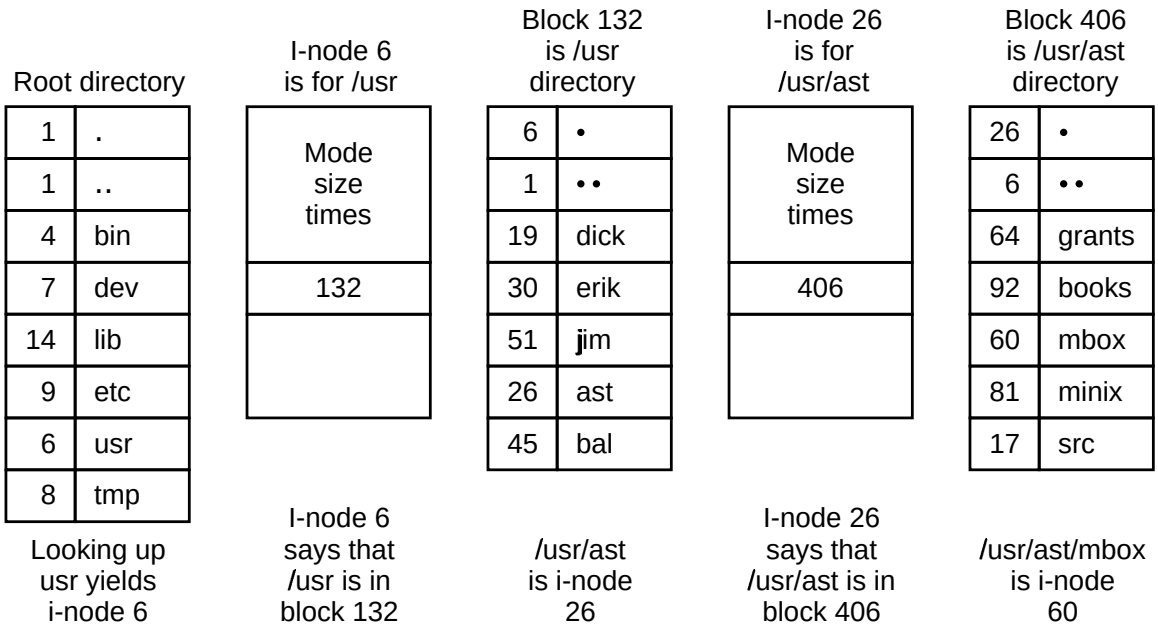

Figure 5-14. The steps in looking up */usr/ast/mbox*.

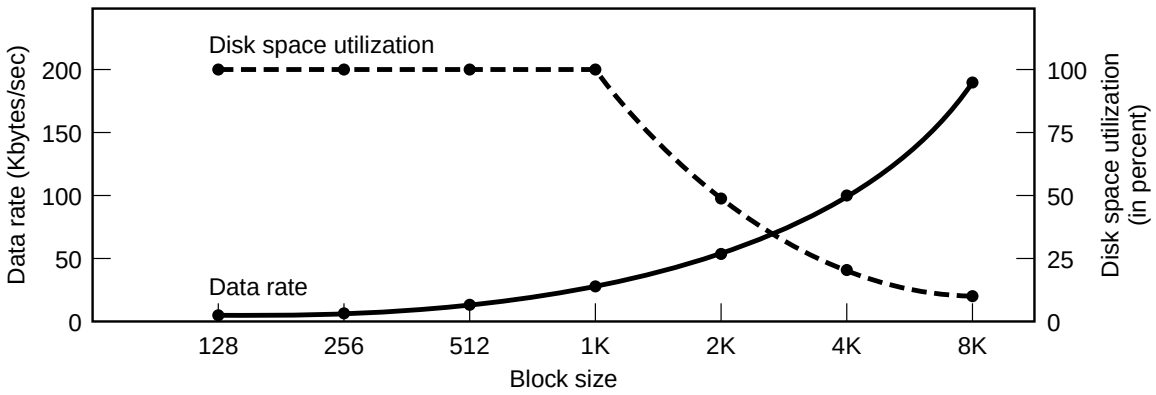

Figure 5-15. The solid curve (left-hand scale) gives the data rate of a disk. The dashed curve (right-hand scale) gives the disk space efficiency. All files are 1K.

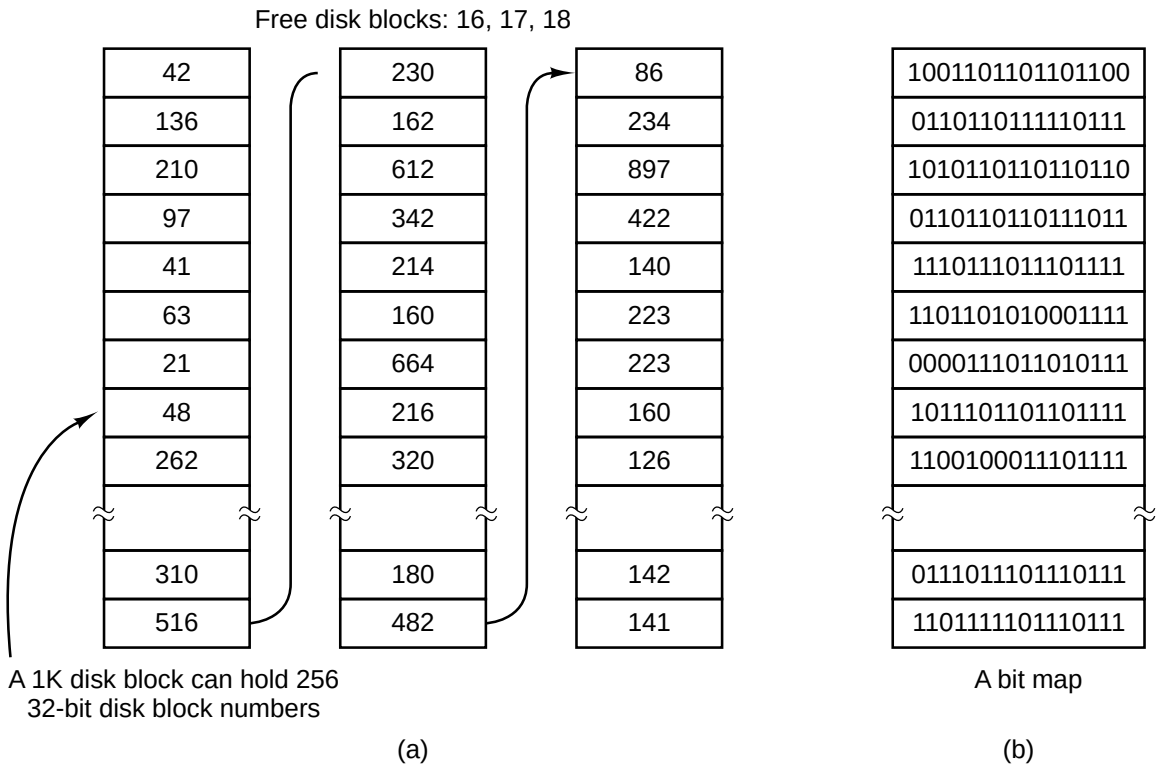

Figure 5-16. (a) Storing the free list on a linked list. (b) A bit map.

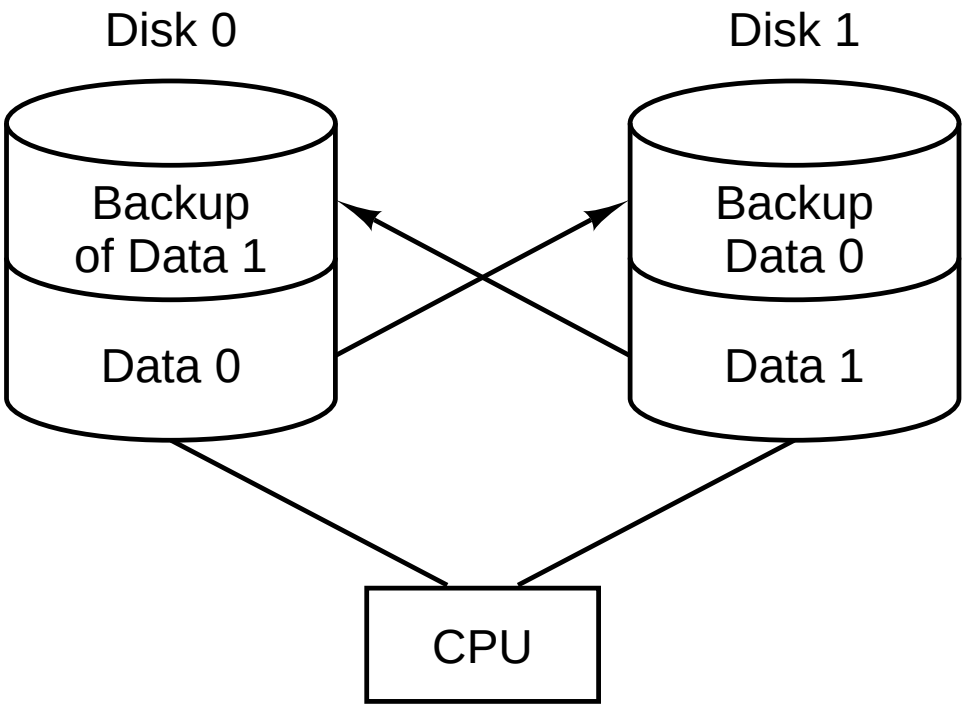

Figure 5-17. Backing up each drive on the other one wastes half the storage.

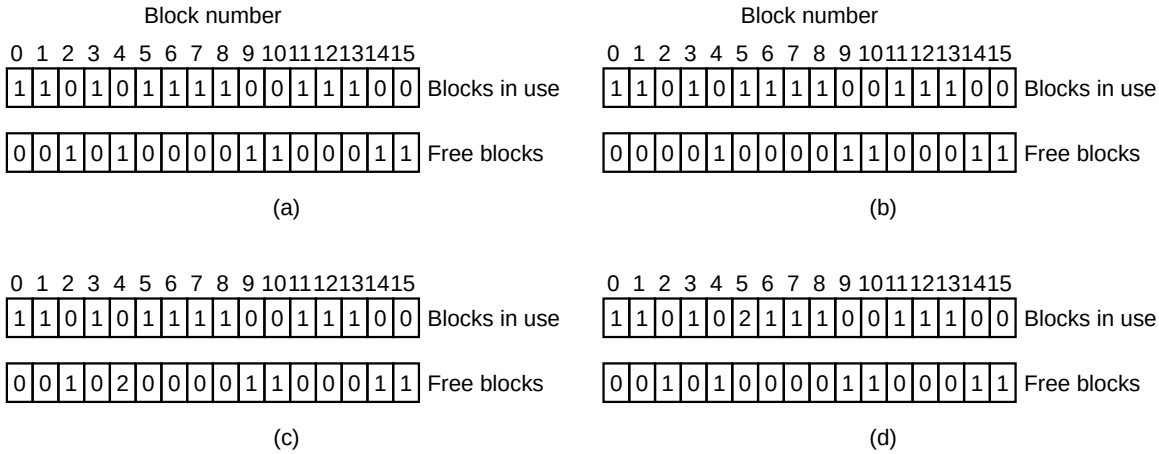

Figure 5-18. File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

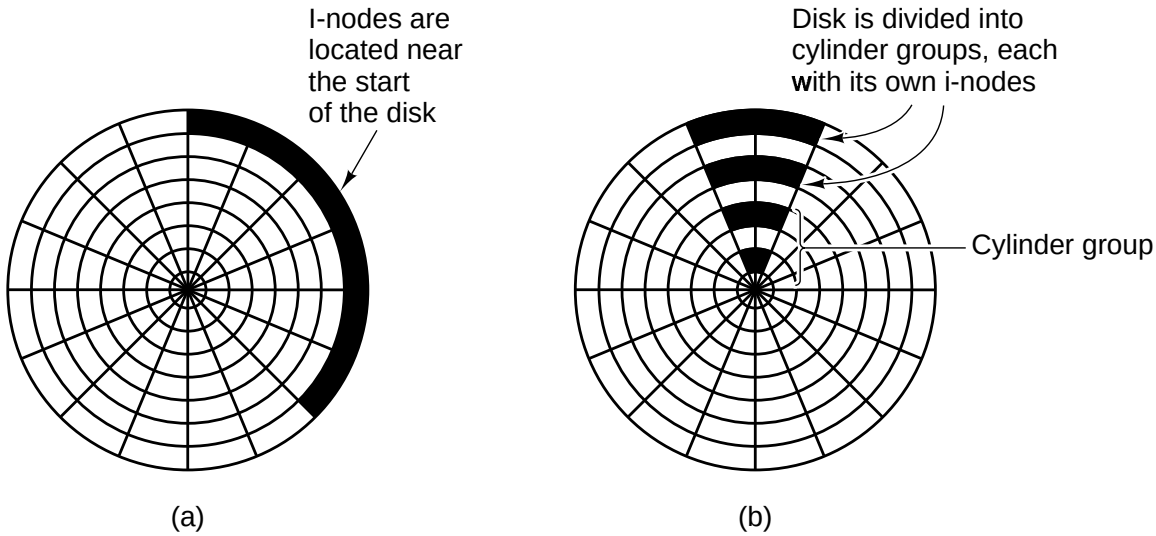

Figure 5-19. (a) I-nodes placed at the start of the disk. (b) Disk divided into cylinder groups, each with its own blocks and i-nodes.

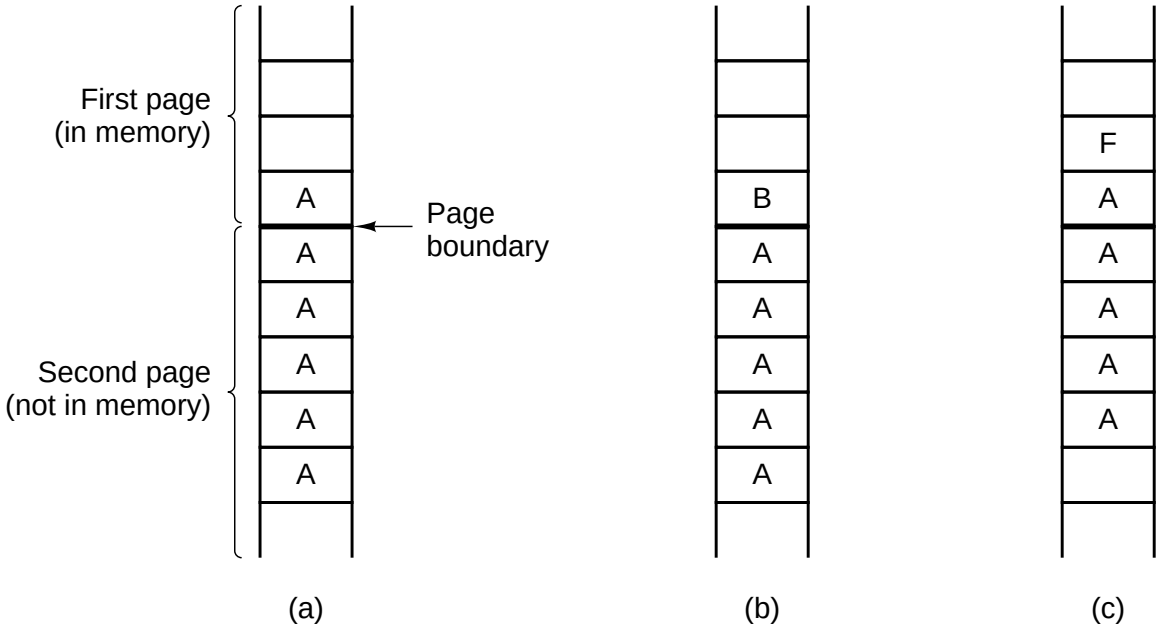

Figure 5-20. The TENEX password problem.

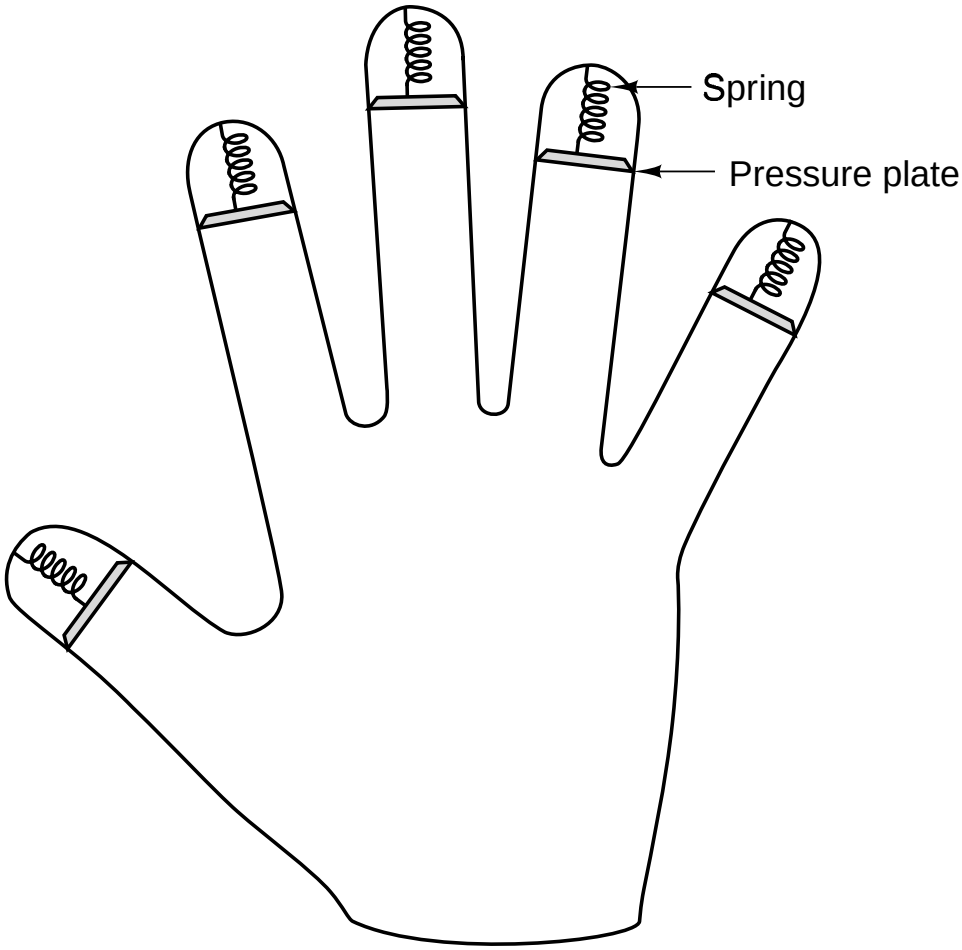

Figure 5-21. A device for measuring finger length.

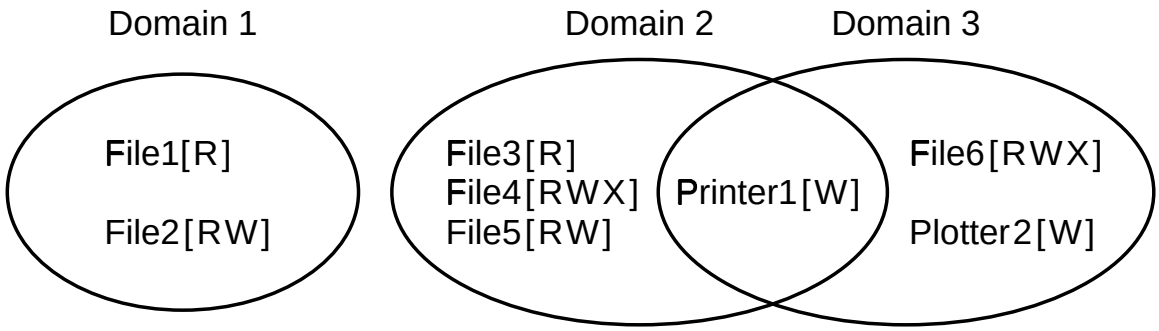

Figure 5-22. Three protection domains.

	Object							
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
Domain 1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

Figure 5-23. A protection matrix.

	Object										
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
Domain 1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

Figure 5-24. A protection matrix with domains as objects.

#	Type	Rights	Object
0	File	R – –	Pointer to File3
1	File	RWX	Pointer to File4
2	File	RW–	Pointer to File5
3	Pointer	–W–	Pointer to Printer1

Figure 5-25. The capability list for domain 2 in Fig. 5-23.

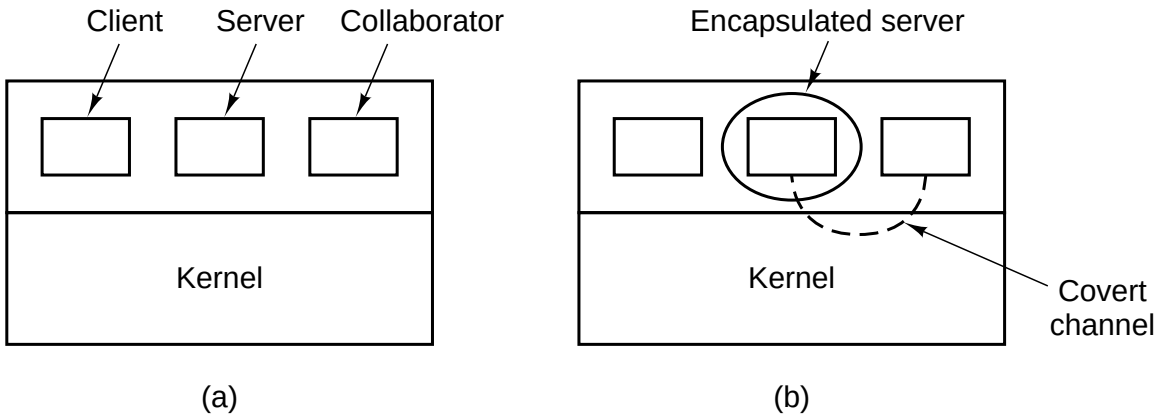

Figure 5-26. (a) The client, server, and collaborator processes. (b) The encapsulated server can still leak to the collaborator via covert channels.

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
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
Messages from MM	**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 5-27. File system messages.

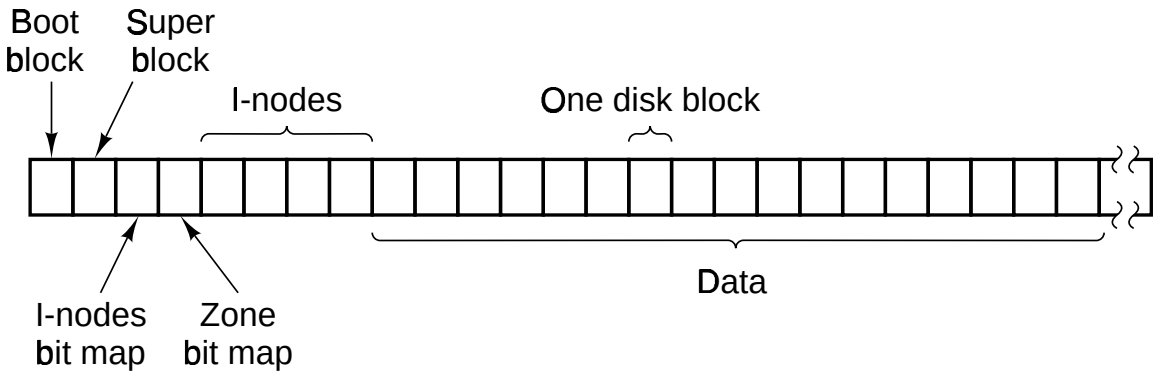

Figure 5-28. Disk layout for the simplest disk: a 360K floppy disk, with 128 i-nodes and a 1K block size (i.e., two consecutive 512-byte sectors are treated as a single block).

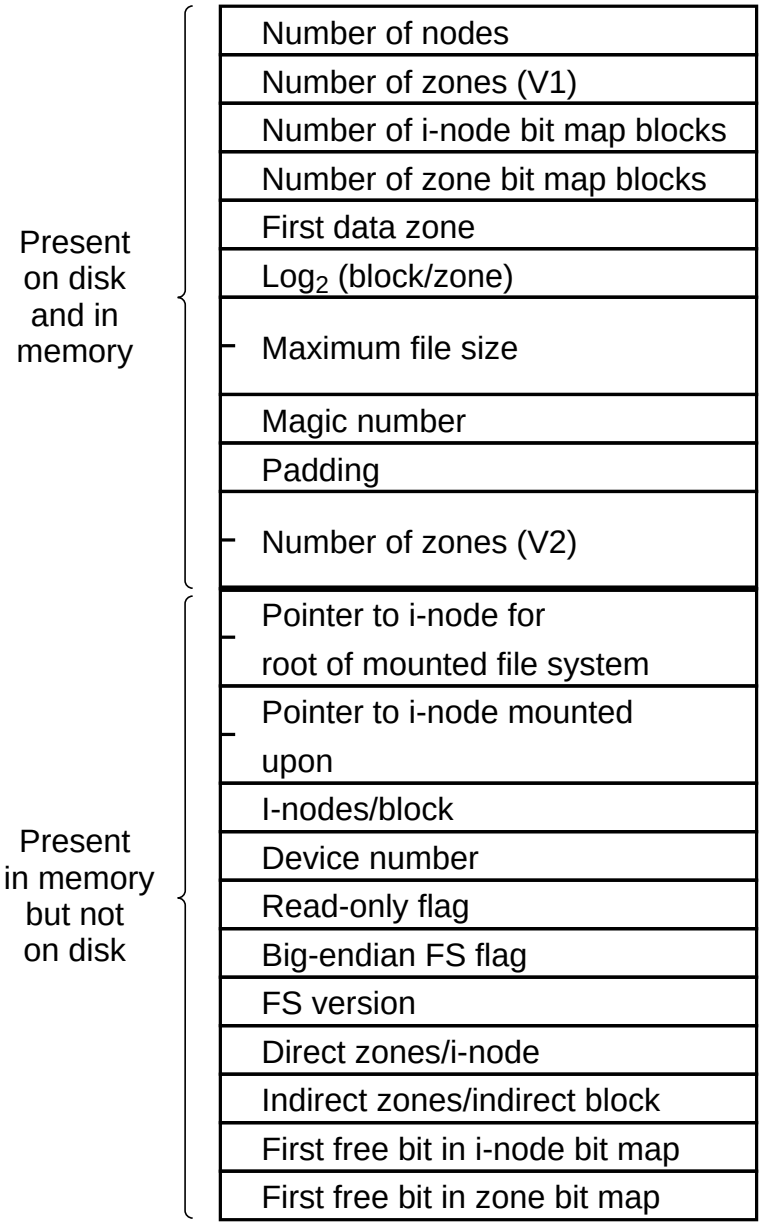

Figure 5-29. The MINIX super-block.

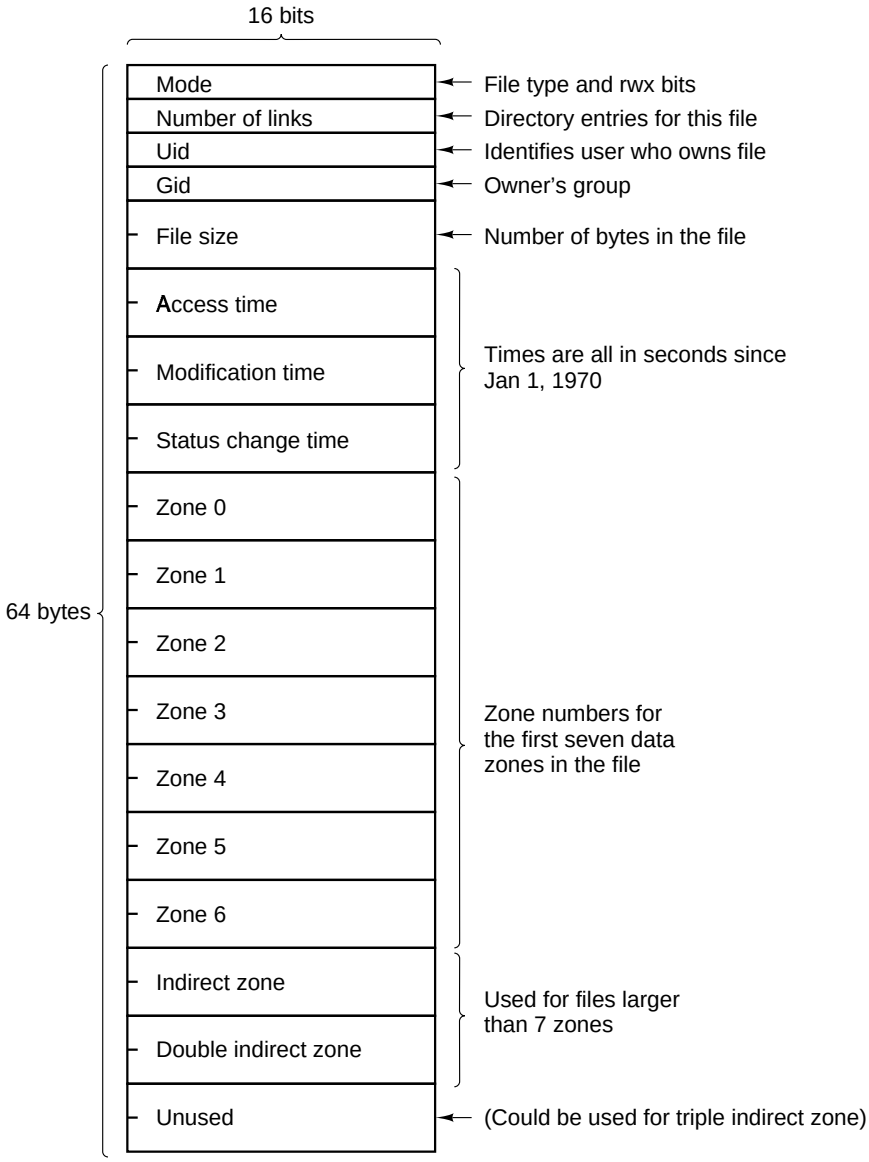

Figure 5-30. The MINIX i-node.

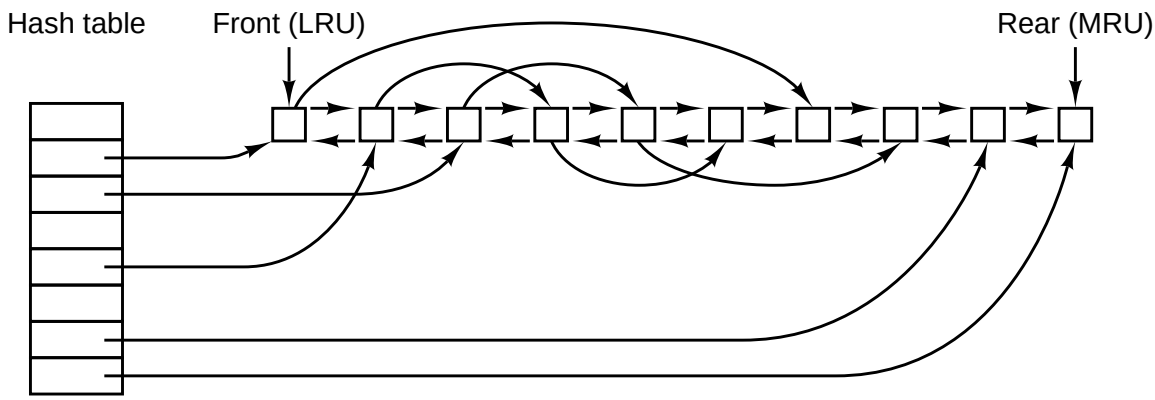

Figure 5-31. The linked lists used by the block cache.

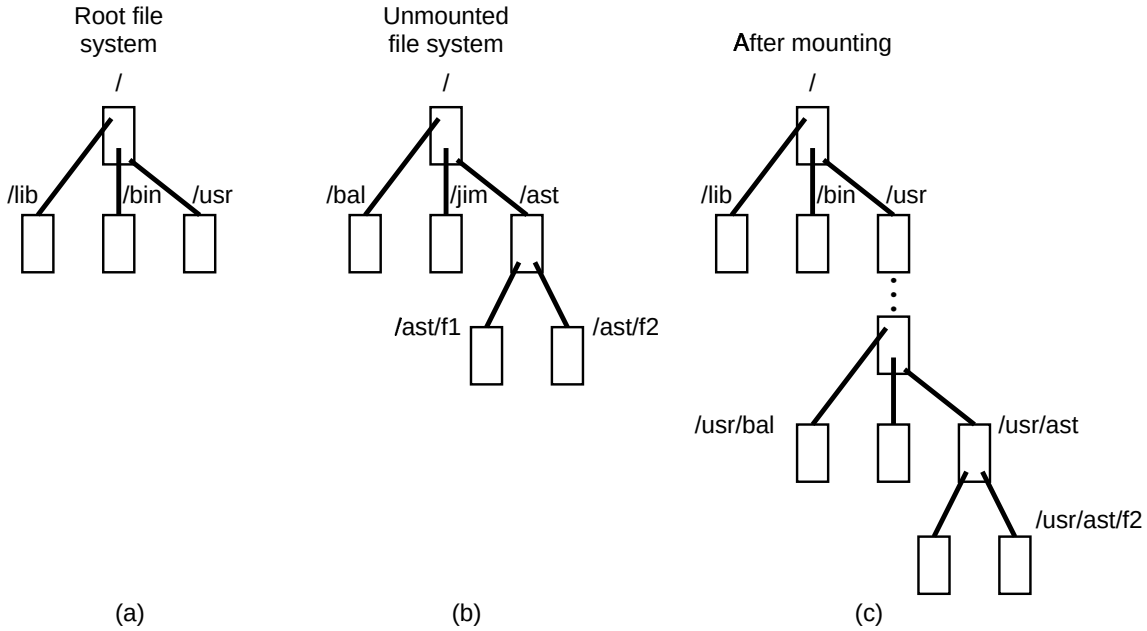

Figure 5-32. (a) Root file system. (b) An unmounted file system. (c) The result of mounting the file system of (b) on */usr*.

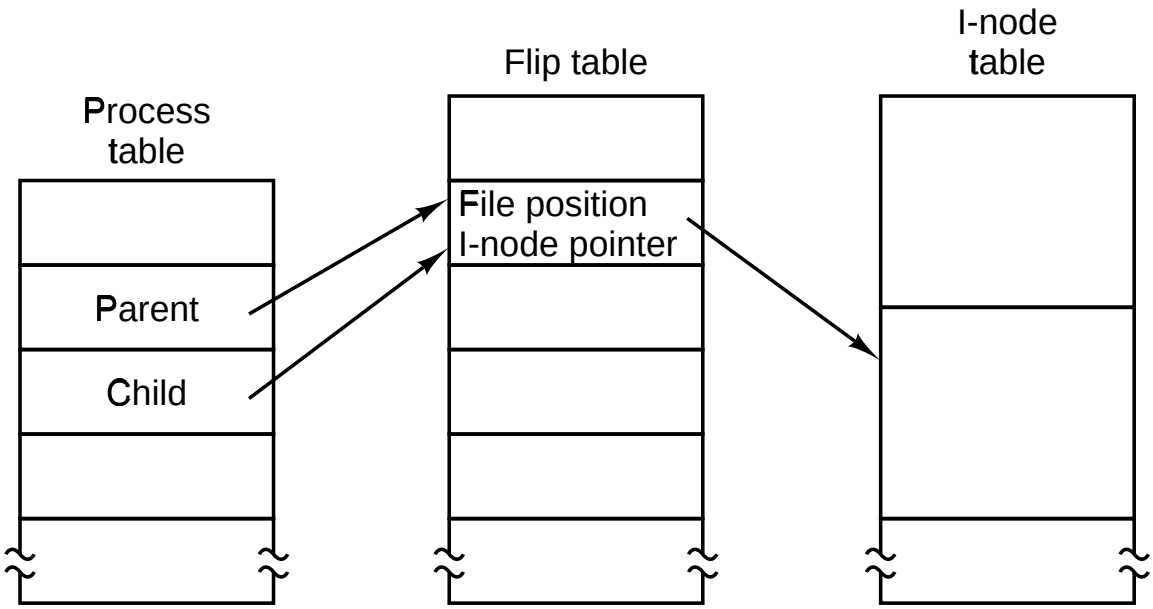

Figure 5-33. How file positions are shared between a parent and a child.

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 5-34. Procedures used for block 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 5-35. Procedures used for i-node 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 super-block table for a device
mounted	Report whether given i-node is on a mounted (or root) f.s.
read_super	Read a super-block

Figure 5-36. Procedures used to manage the super-block and bit maps.

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 5-37. The POSIX advisory record locking operations. These operations are requested by using an FCNTL system call.

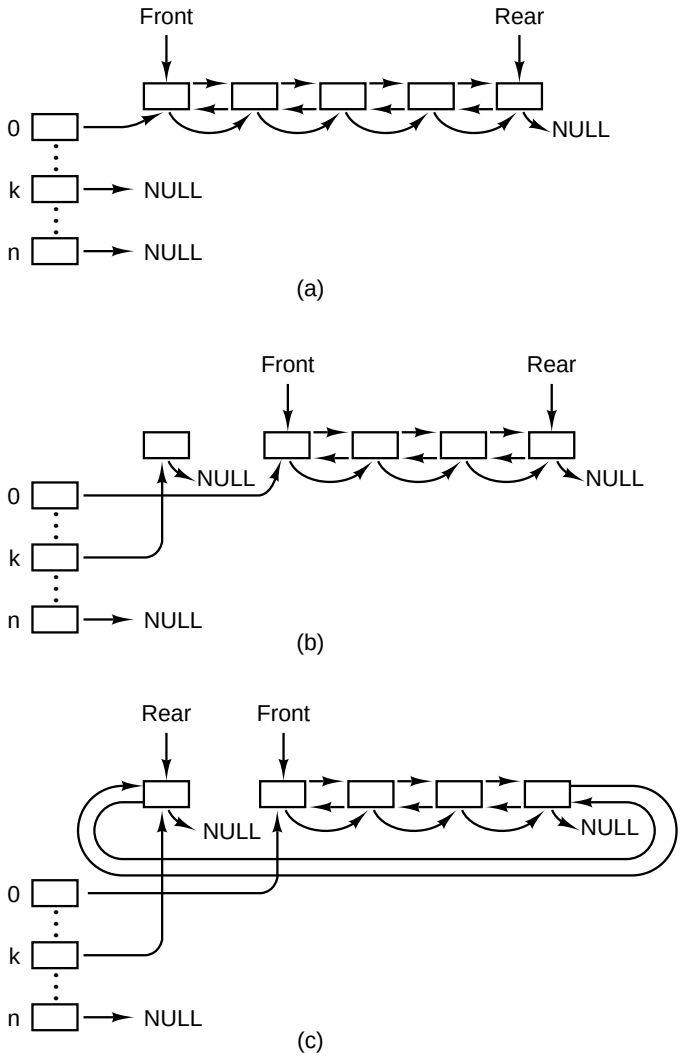

Figure 5-38. Block cache initialization. (a) Before any buffers have been used. (b) After one block has been requested. (c) After the block has been released.

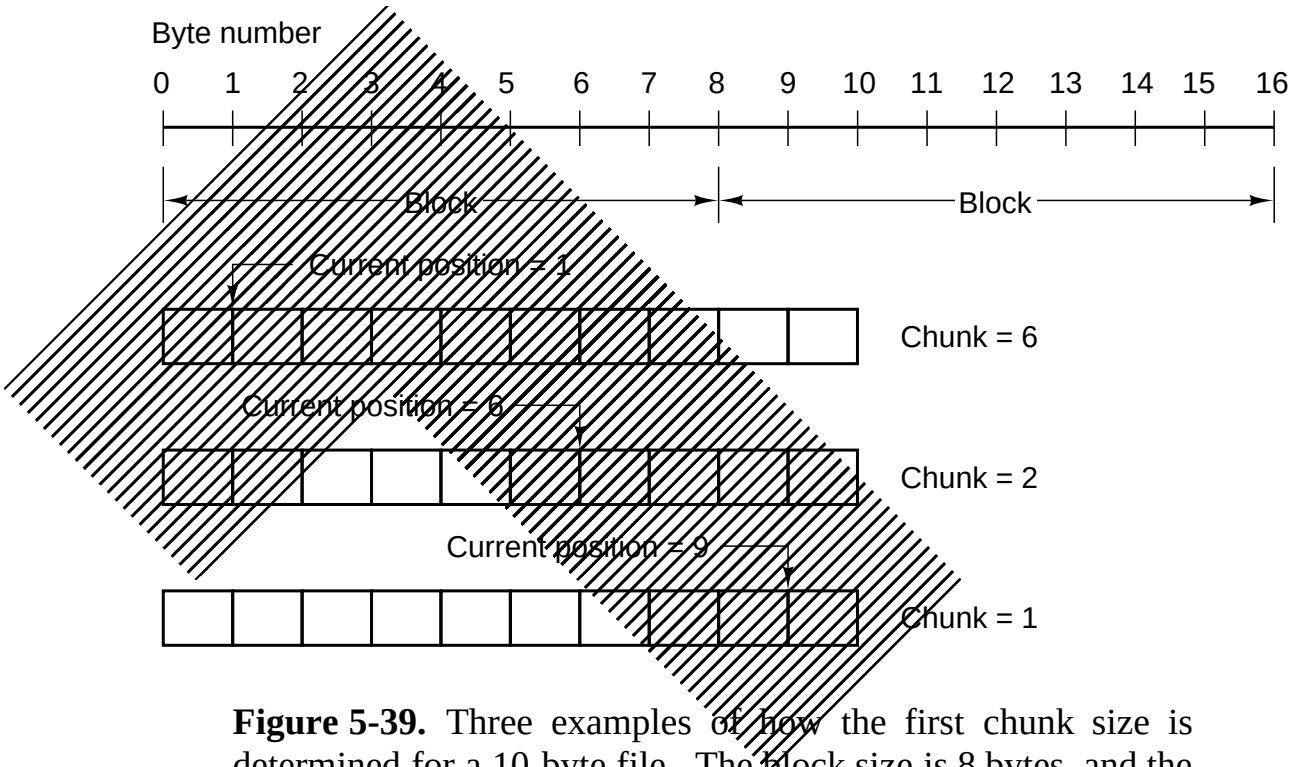

Figure 5-39. Three examples of how the first chunk size is determined for a 10-byte file. The block size is 8 bytes, and the number of bytes requested is 6. The chunk is shown shaded.

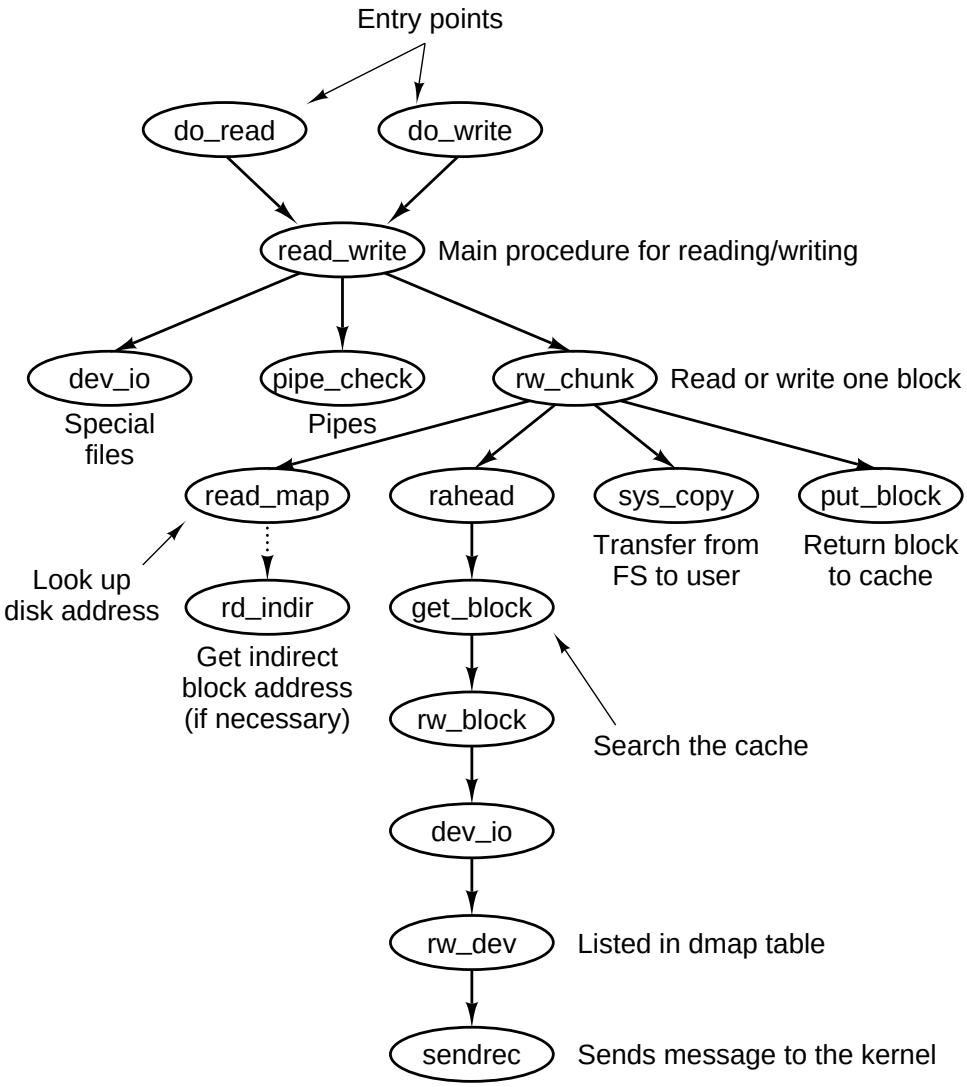

Figure 5-40. Some of the procedures involved in reading a file.

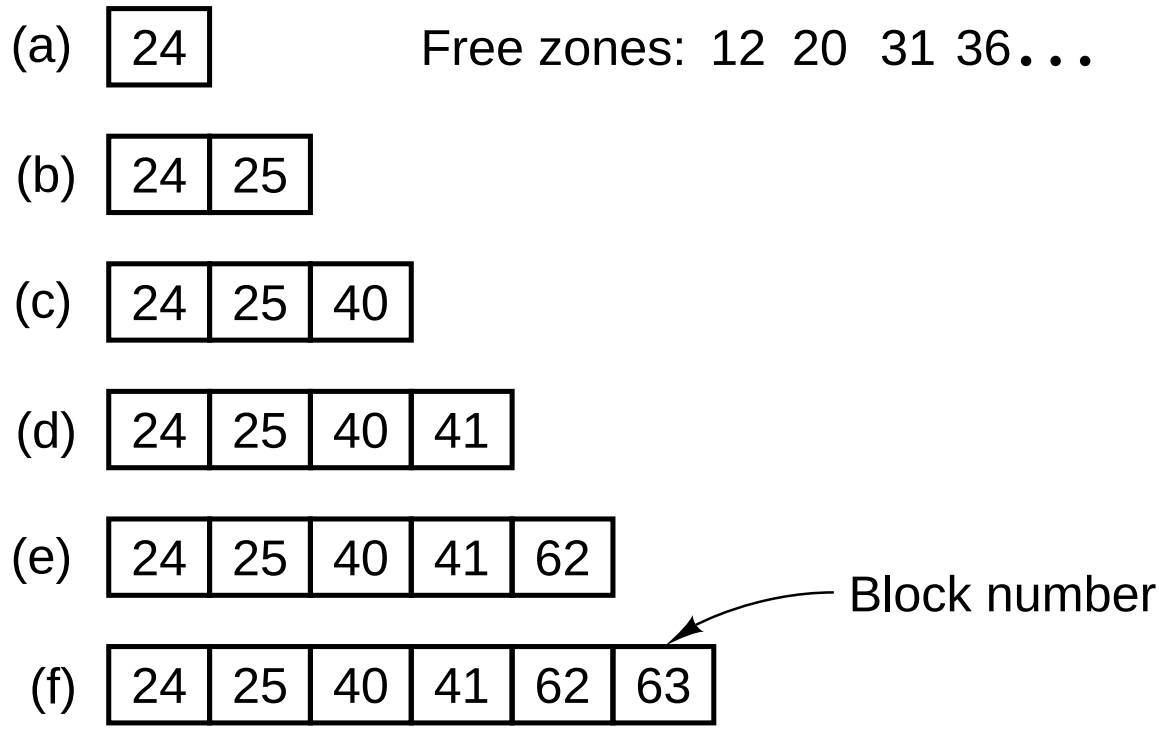

Figure 5-41. (a) - (f) The successive allocation of 1K blocks with a 2K zone.

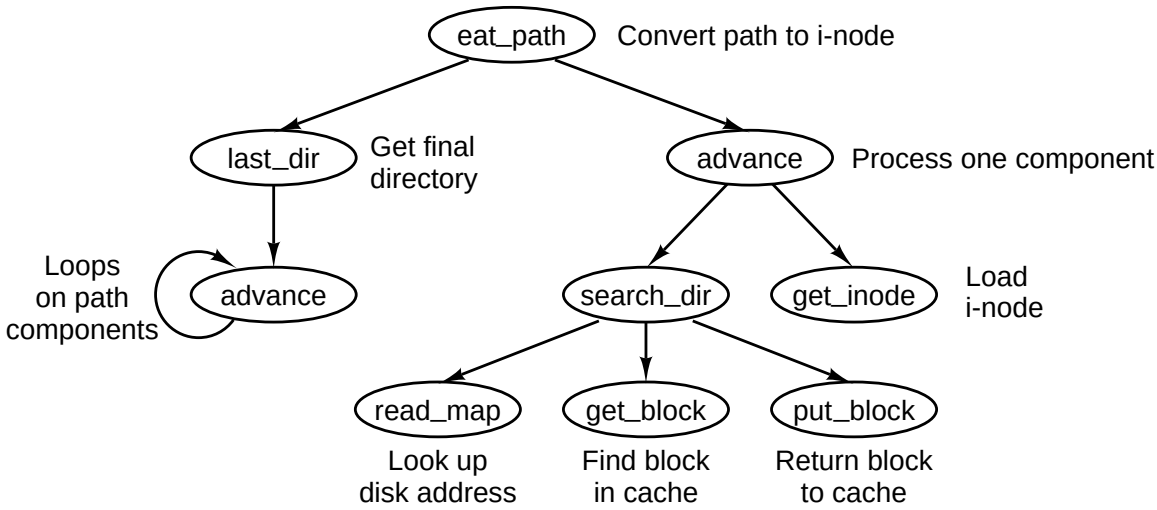

Figure 5-42. Some of the procedures used in looking up path names.

Call	Function
UTIME	Set a file's time of last modification
TIME	Set the current real time in seconds
STIME	Set the real time clock
TIMES	Get the process accounting times

Figure 5-43. The four system calls involving time.

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 5-44. The POSIX request parameters for the FCNTL system call.