DIRECTORY BASICS

Module Number 4. Section 2a
COP4600 – Operating Systems
Richard Newman

FILE SYSTEMS TOPICS

- Introduction
- Directories
- File Allocation
- Block Management
- File System Reliability
- File System Optimization

WHAT IS A DIRECTORY?

- A directory is structure by which file names may be organized
- They are used in lookup mechanisms by which a file can be found
- In fact, they are just special files!
  - Include file name/extension
  - Include some other field(s) that allow file storage information and file attributes to be obtained
- Known to system as directory by a file attribute

HIERARCHICAL DIRECTORY SYSTEMS

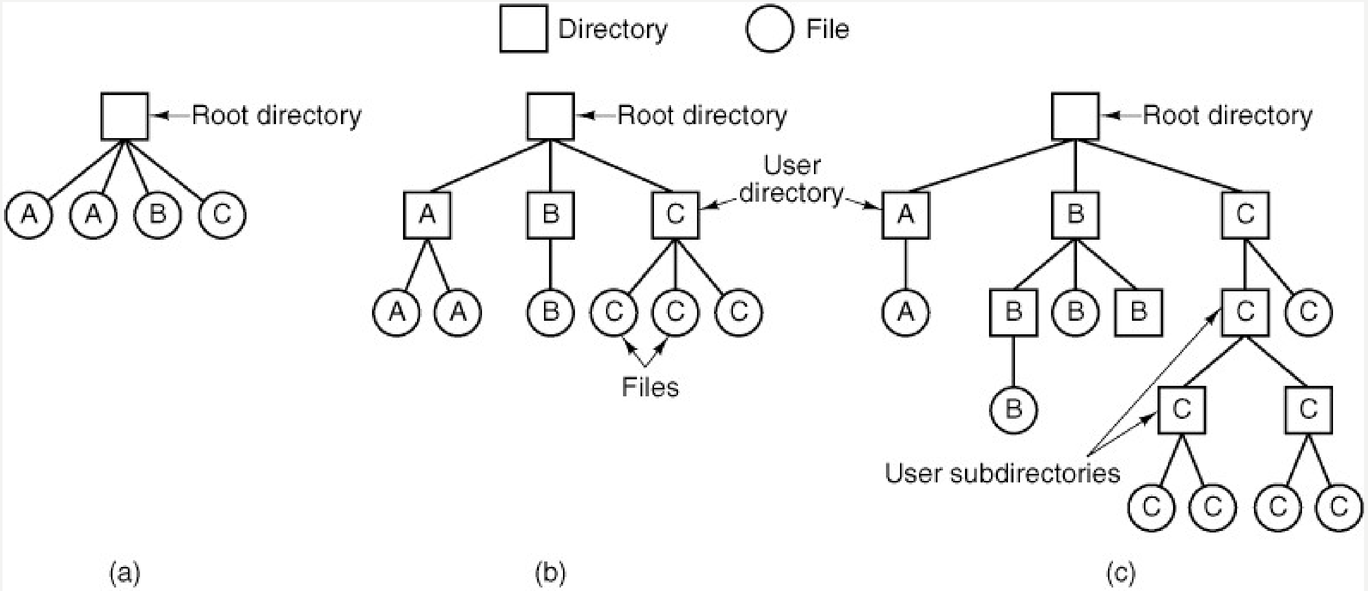
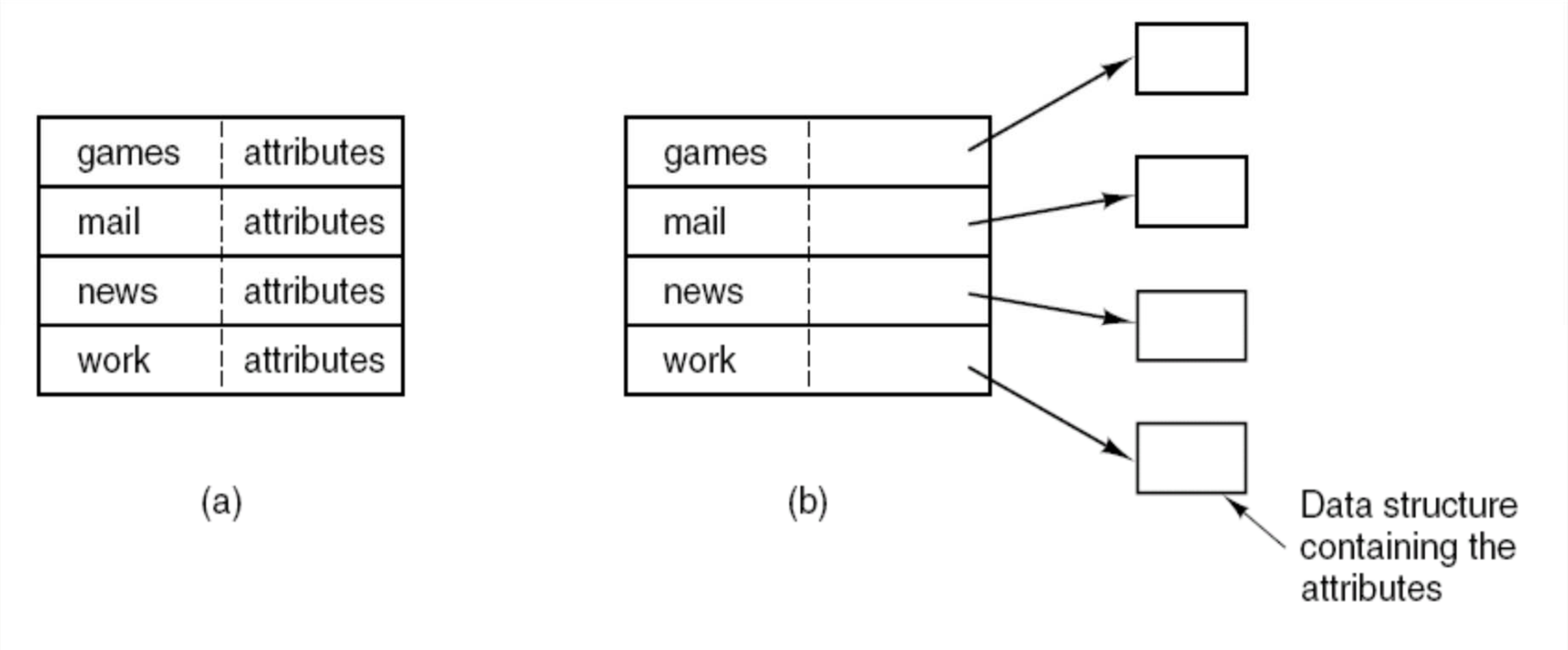


Figure 5-6. Three file system designs. (a) Single directory shared by all users. (b) One directory per user. (c) Arbitrary tree per user. The letters indicate the directory or file's owner.

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

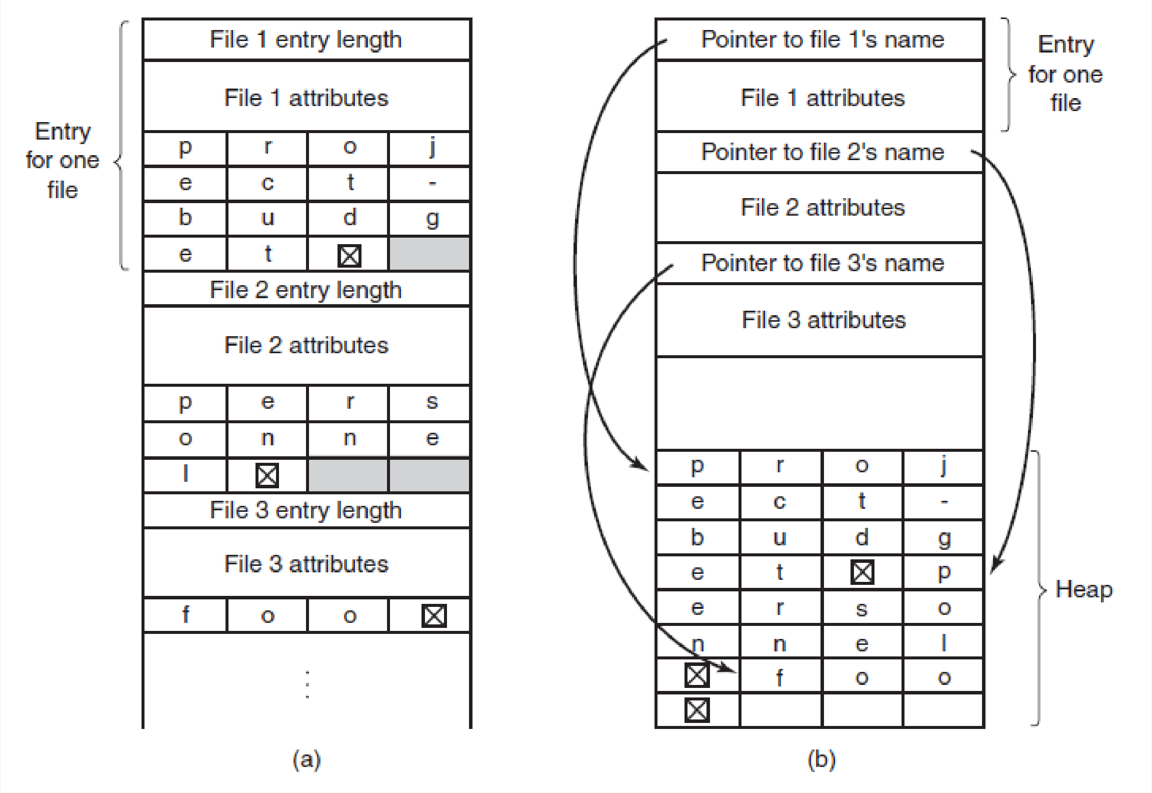
DIRECTORIES



- Figure 4-14. (a) A simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry. (b) A directory in which each entry just refers to an i-node.

Tanenbaum & Bos, Modern Operating Systems:4th ed., (c) 2013 Prentice-Hall, Inc. All rights reserved.

IMPLEMENTING DIRECTORIES (2)



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

Pointer to file 2's name

File 2 attributes

Pointer to file 3's name

File 3 attributes

Entry for one file

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

☒

Heap

(b)

Figure 4-15. Two ways of handling long file names in a directory. (a) In-line. (b) In a heap.

Tanenbaum & Bos, Modern Operating Systems:4th ed., (c) 2013 Prentice-Hall, Inc. All rights reserved.

USING DIRECTORIES FOR FILE LOOKUP

- Path names
  - Path = sequence of directories to directory holding file, then file or directory name
  - Absolute: path starts from root directory
  - Relative: path starts from current working directory (implies process has a CWD as part of its state)
- Path Resolution
  - Recursively parse path to find next element (directory or file name), then look up that name in current search directory, then update current search directory until file is found (or fail)

PATH NAMES

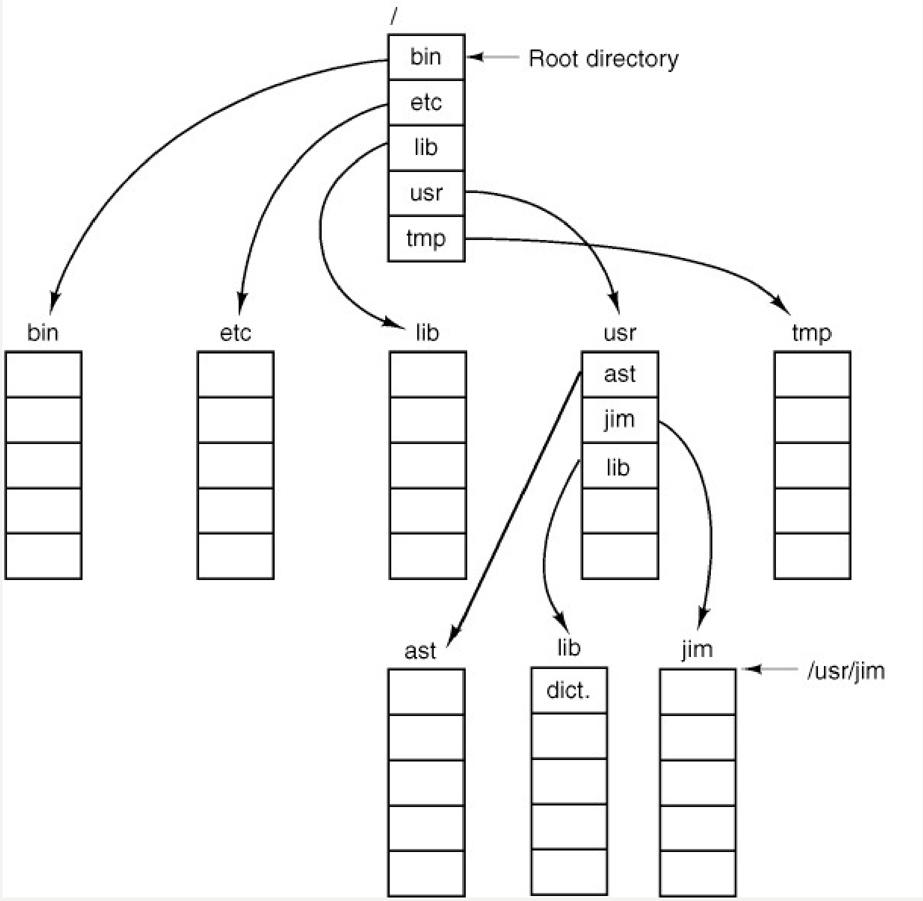


Figure 5-7.A UNIX directory tree.

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

SHARED FILES

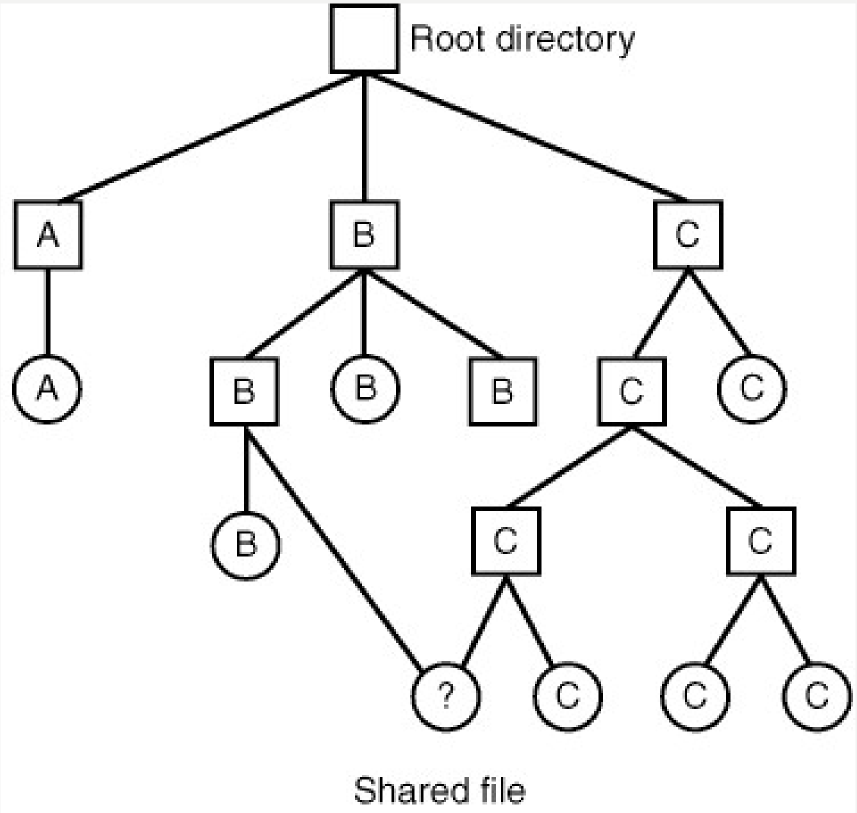
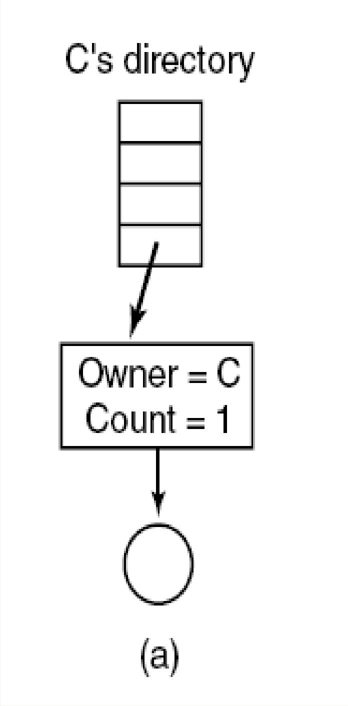


Figure 5-12. File system containing a shared file.

Multiple names for same file, what about directory?

SHARED FILES (2)



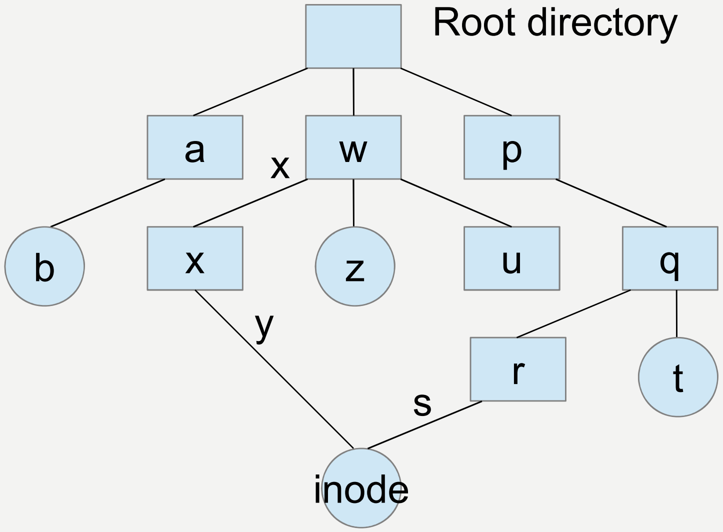
(a) Situation prior to linking.

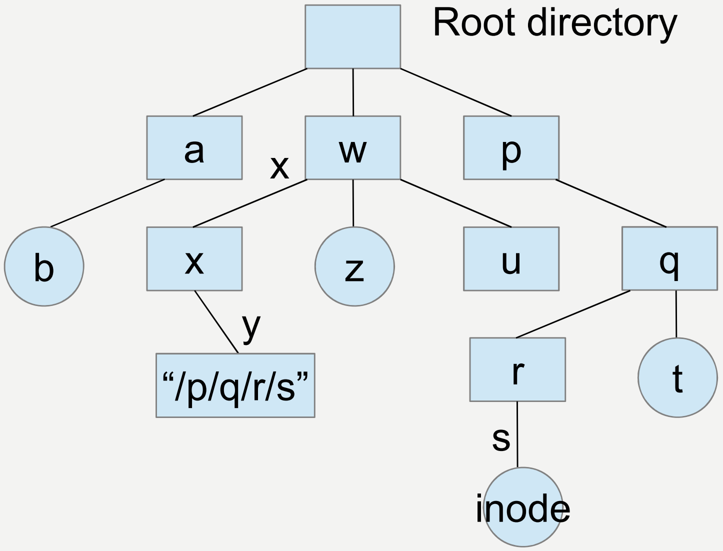
(b) After the link is created.

(c) After the original owner removes the file.

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

HARD AND SOFT (SYMBOLIC) LINKS





Hard link – directory holds
inode number of file

Must be on same file system

Why?

Soft link – directory holds
path of file

Can span file systems

Can get "link rot"

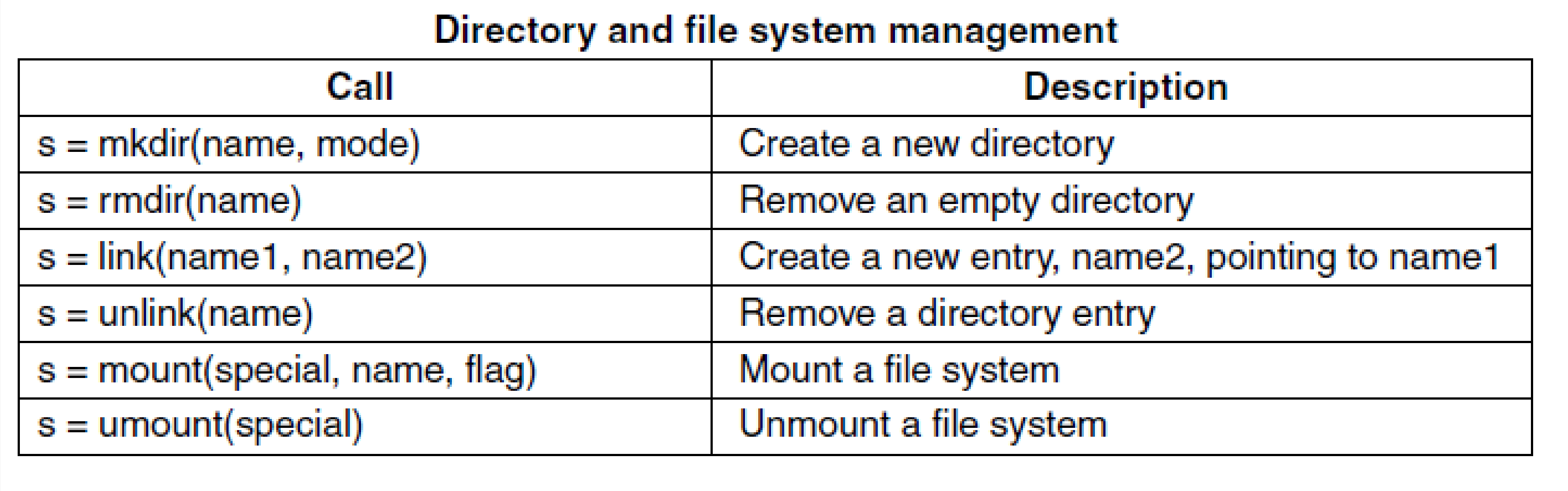
SHARED FILES (3)

- When does a file go away?
  - Allows the file's space to be reused…
- When file can no longer be accessed
  - Like garbage collection in Lisp
- System detects when count == 0, then there are no links to the i-node, hence no directory entries, so the file cannot be reached from the root directory
- What about if the file is currently open and being used by a process?
- When a file is opened, a count is also incremented, and decremented when closed
- !!! Can have a process use a file that is not accessible!!!
  - A hiding trick of the Internet Worm

DIRECTORY OPERATIONS

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

POSIX SYSTEM CALLS - DIRECTORIES



Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Figure 1-18. Some of the major POSIX system calls.
The return code *s* is −1 if an error has occurred.

Tanenbaum & Bos, Modern Operating Systems:4th ed., (c) 2013 Prentice-Hall, Inc. All rights reserved.

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.

INITIALIZING FILE SYSTEM

File system exists on persistent medium

- May be removable -> need way for system to know FS info
- Closure problem – where is root directory?
- May have multiple different file system types on same computer
  - Hard disk partitions
  - Floppy disk/removable HDD
  - Flash drive/USB stick
  - CD or DVD drive

FILE SYSTEM LAYOUT

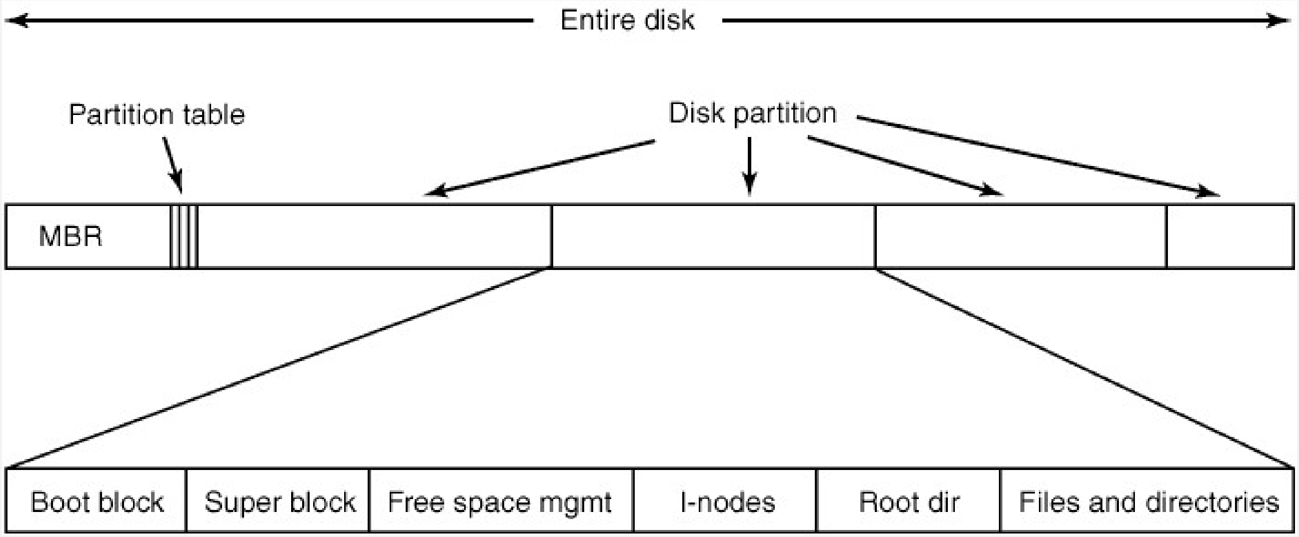


Figure 5-8. A possible file system layout.

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

INTEGRATING FILE SYSTEMS ON HOST

Multiple partitions/drives

- May name them explicitly
  - E.g., Windows A:, C:, etc.
  - BSD r-commands host:path
- May use mounting to integrate multiple file systems into single view for users (and processes)
  - Mount point

DIRECTORIES AND PATHS

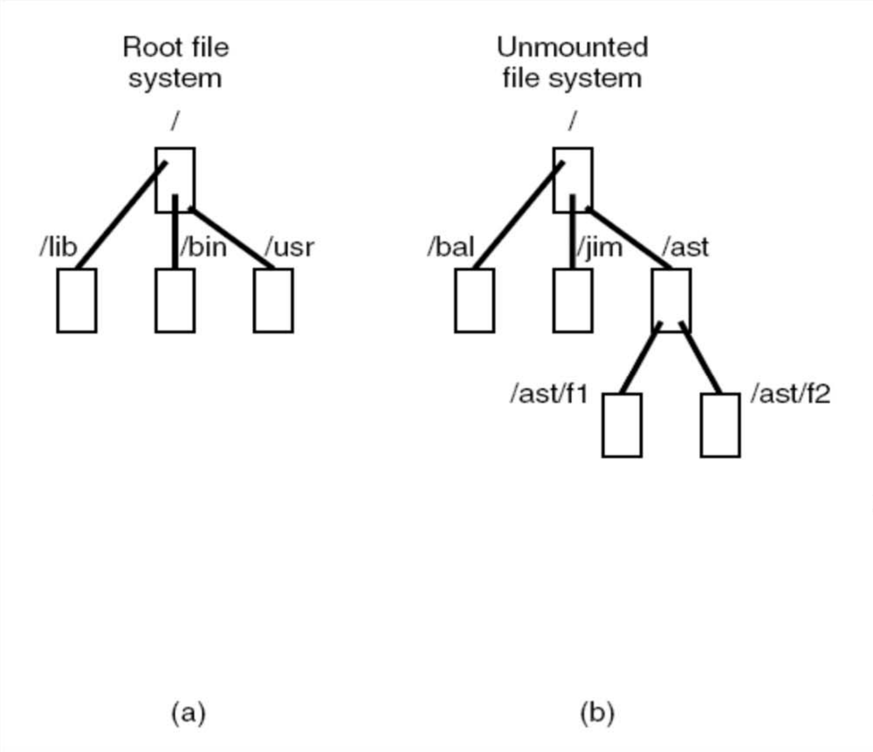


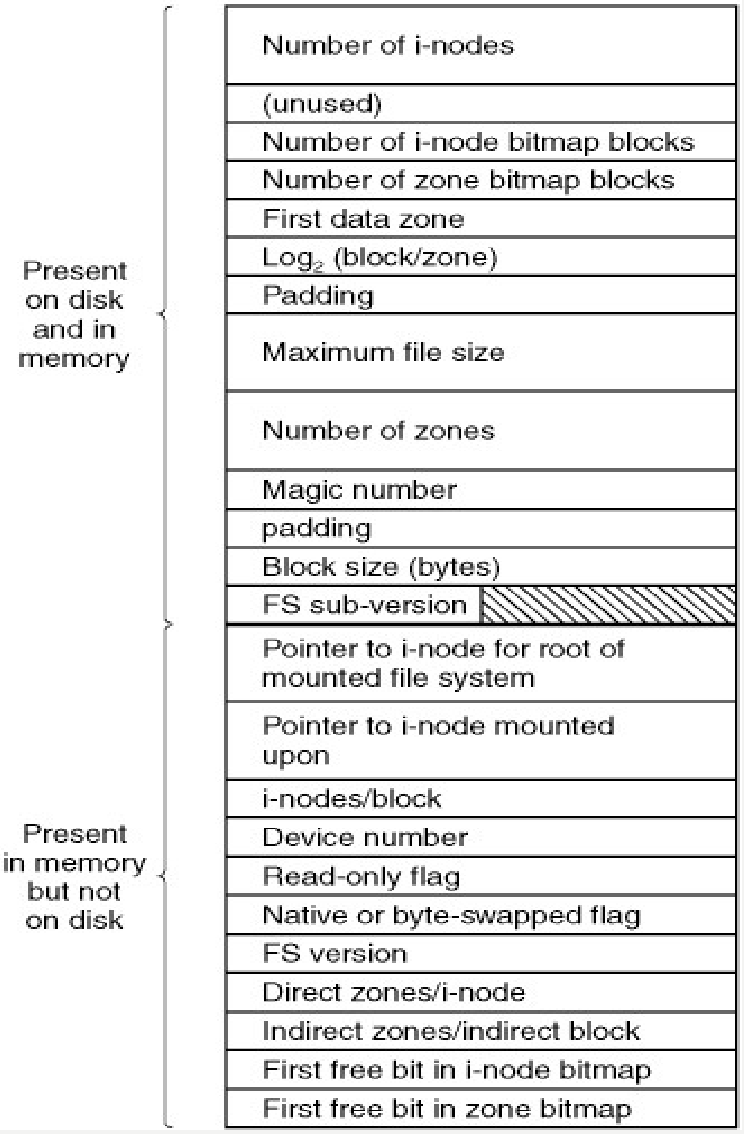
Figure 5-38. (a) Root file system. (b) An unmounted file system. (c) The result of mounting the file system of (b) on */usr/*.

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

MOUNTING AND SUPERBLOCK

Mount call copies superblock from disk into superblock table, sets pointer to i-node for root of mounted FS, sets pointer to i-node where new FS is mounted, sets flag in i-node where new FS is mounted

If flag set, look for i-node number matching mount point in superblock table, then get device and i-node number of root of FS (and subtree)



Present on disk and in memory

- Number of i-nodes
- (unused)
- Number of i-node bitmap blocks
- Number of zone bitmap blocks
- First data zone
- Log_2 (block/zone)
- Padding
- Maximum file size
- Number of zones
- Magic number
- padding
- Block size (bytes)
- FS sub-version

Present in memory but not on disk

- Pointer to i-node for root of mounted file system
- Pointer to i-node mounted upon
- i-nodes/block
- Device number
- Read-only flag
- Native or byte-swapped flag
- FS version
- Direct zones/i-node
- Indirect zones/indirect block
- First free bit in i-node bitmap
- First free bit in zone bitmap

Tanenbaum & Woodhull, Operating Systems: Design and Implementation, (c) 2006 Prentice-Hall, Inc. All rights reserved. 0-13-142938-8

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.

SUMMARY

- Types of file system directory structures
  - Flat/2-level/hierarchical/DAG
- Types of directories
  - FCB in directory/Indirection
  - Long file names
- Path resolution
- Shared files
- Directory operations
- File system layout
- Mounting