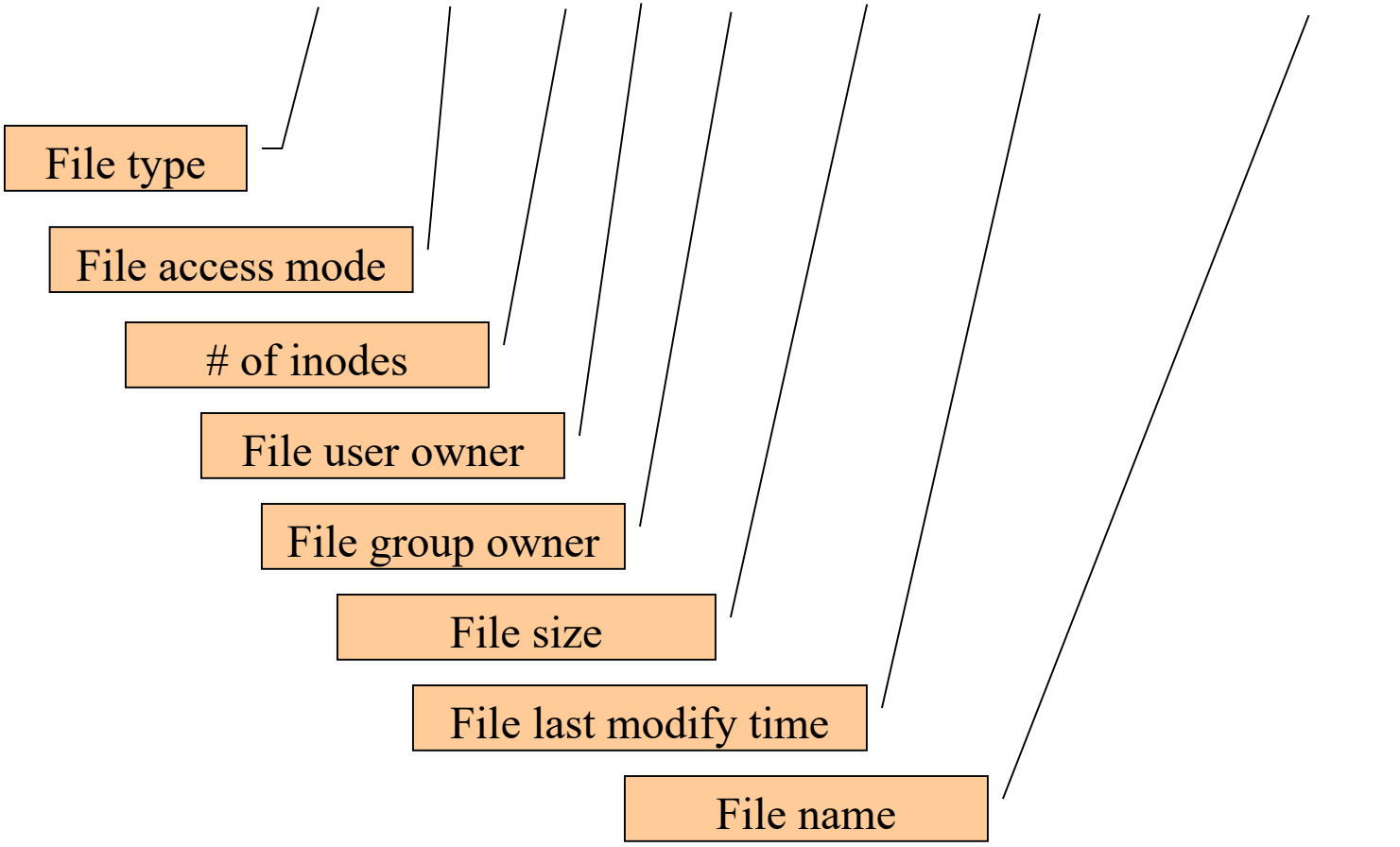
File System

Files

- % ls –l
  - d rwx--x--x 7 liuyh gcs 1024 Sep 22 17:25 public_html

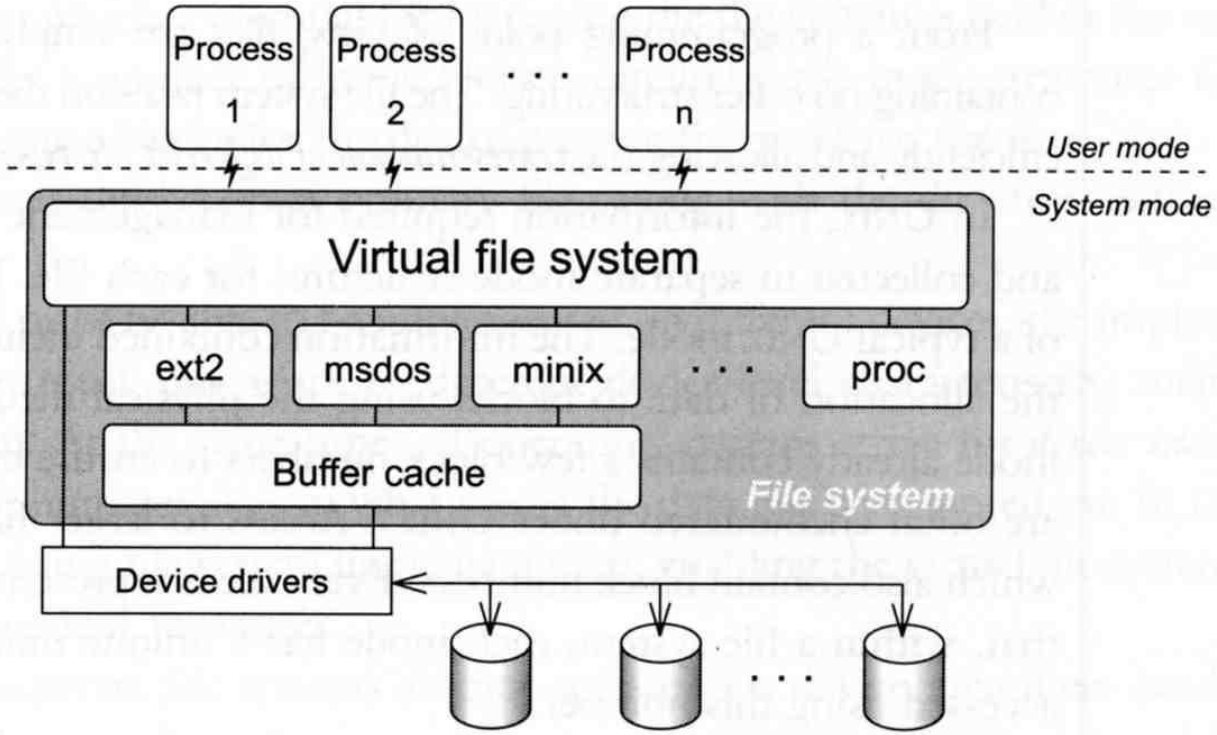


Outline

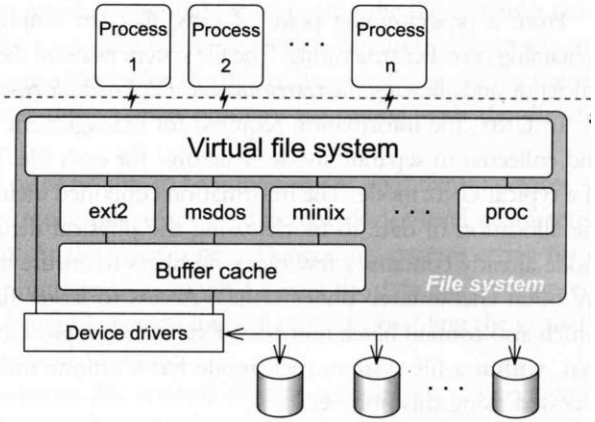
- File System Architecture
  - Pathname
  - File Tree
  - Mounting
  - File Types
- inode and file
  - Link
- File Access Mode
  - Changing File Owner
  - FreeBSD bonus flags

File System Architecture (1)

- Application ⇔ Kernel ⇔ Hardware
  - Applications call system-calls to request service
  - Kernel invokes corresponding drivers to fulfill this service



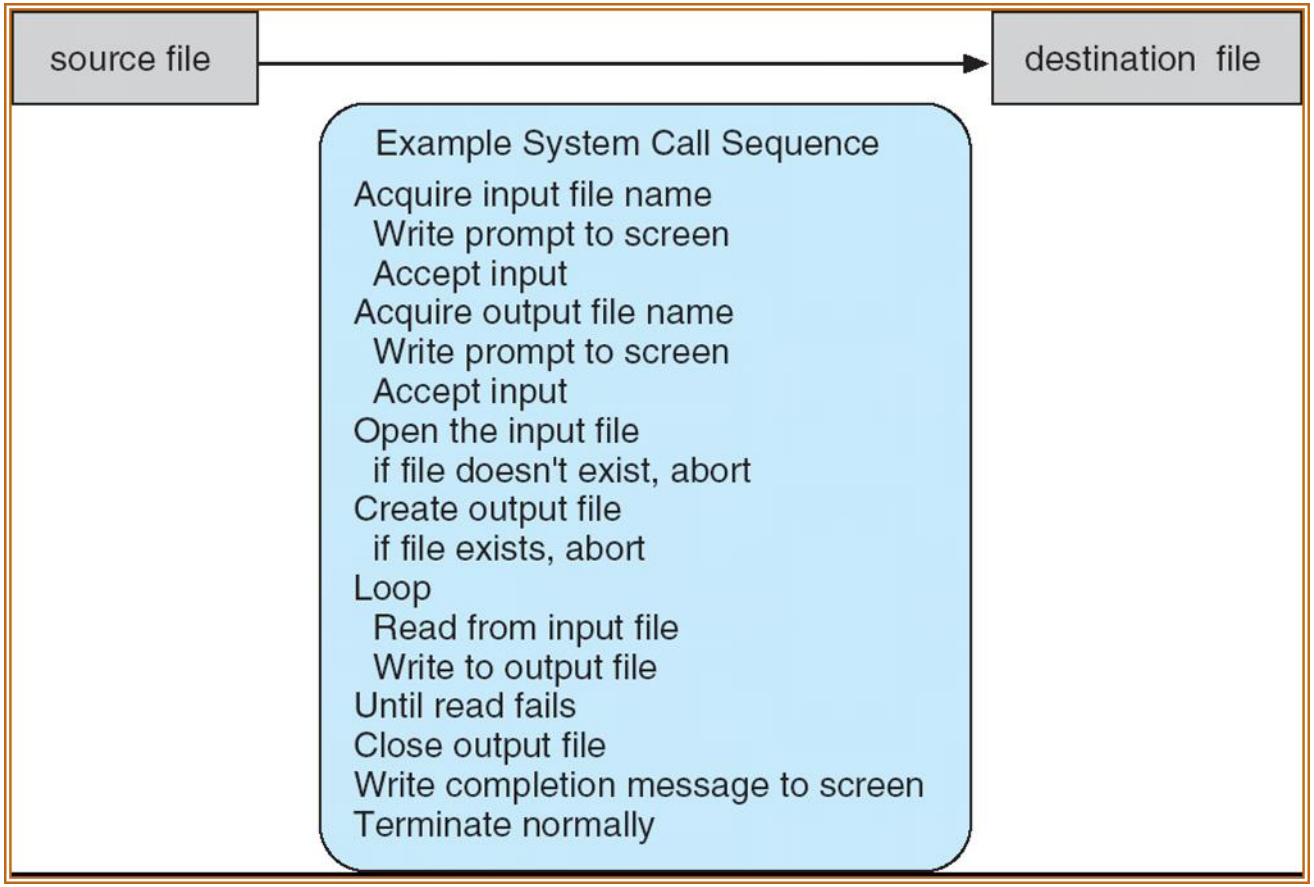
File System Architecture (2)



- The basic purpose of filesystem
  - Represent and organize the system's storage
  - Four main components:
    - Namespace
      - A way of naming things and arranging them in a hierarchy
    - Application Programming Interface (API)
      - A set of system calls for navigating and manipulating nodes
    - Security model
      - A scheme for protecting, hiding and sharing things
    - Implementation
      - Code that ties the logical model to an actual disk

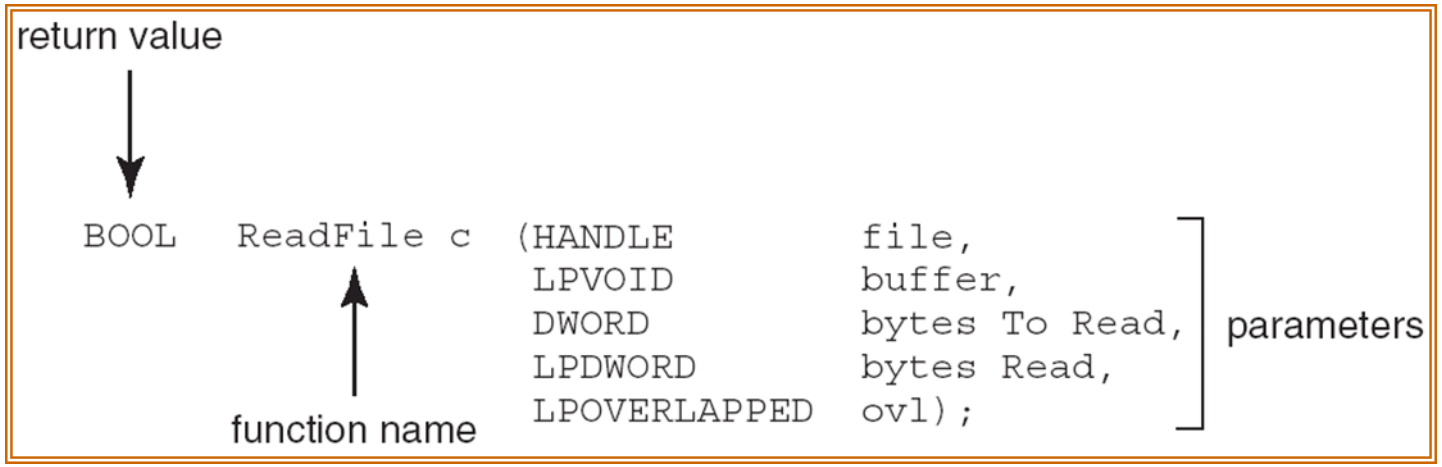
File System Architecture (2)

- System call sequence to copy the contents of one file to another file



File System Architecture (2)

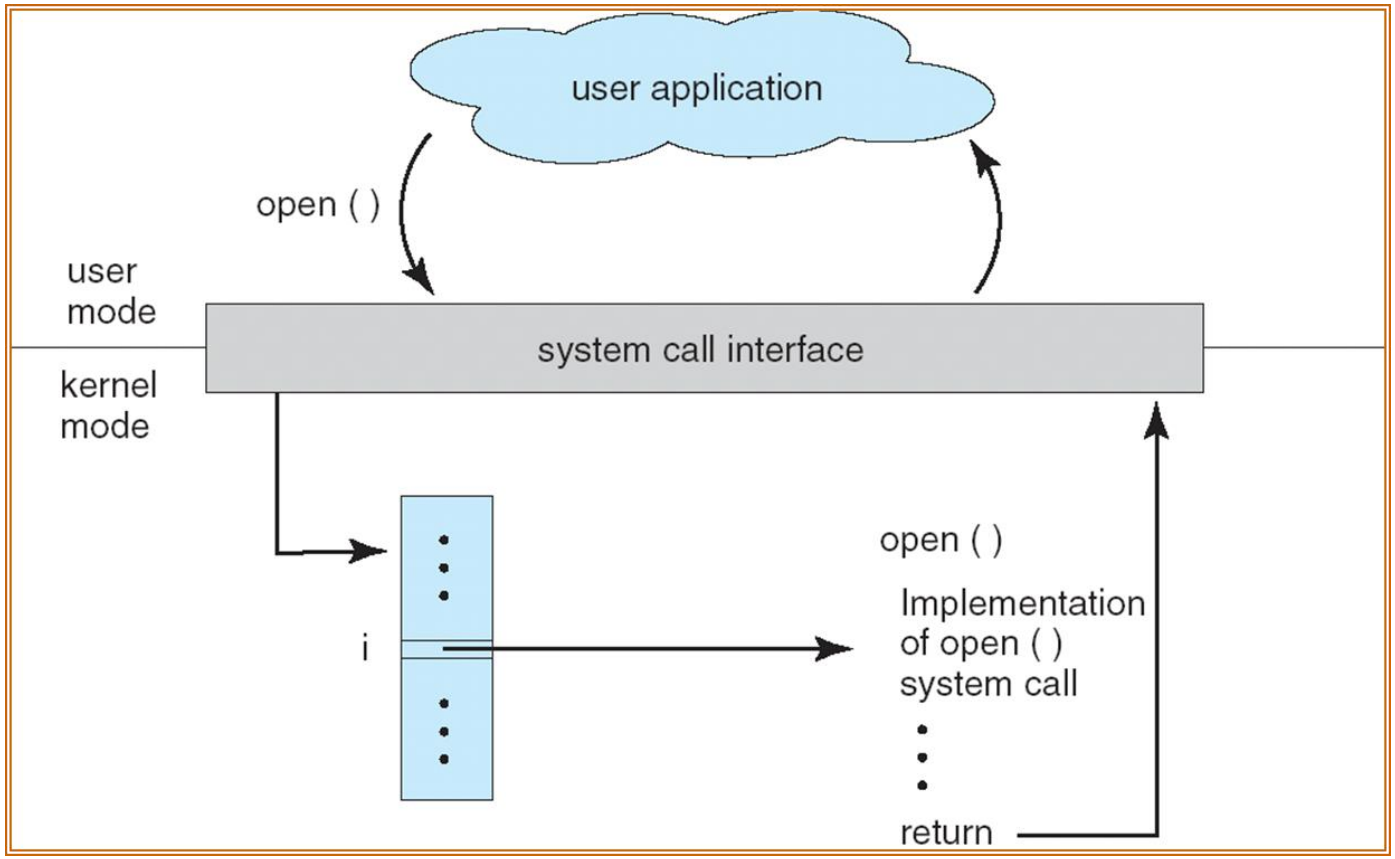
- Consider the ReadFile() function in the Win32 API – a function for reading from a file



- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

File System Architecture (2)

- API – System Call – OS Relationship



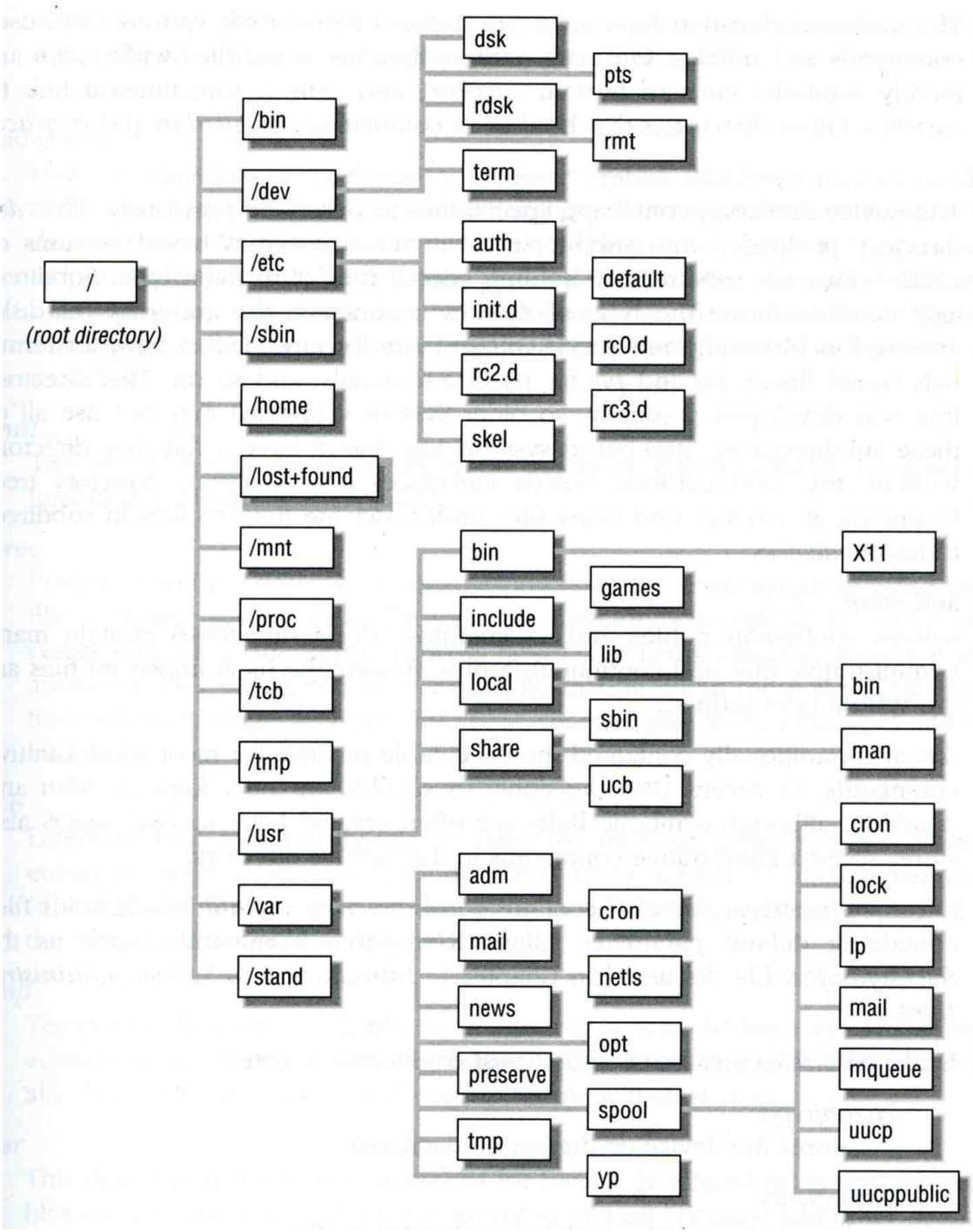
File System Architecture (3)

- Objects in the filesystem:
  - What you can find in a filesystem:
    - Files and directories
    - Hardware device files
    - Processes information
    - Interprocess communication channel (IPC)
    - Shared memory segments (SHM)
  - We can use common filesystem interface to access such “object”
    - open、read、write、close、seek、ioctl, fcntl, …

Pathname

- Two kinds of path
  - Absolute path ➔ start from /
    - Ex. /u/dcs/97/9755806/test/hehe.c
  - Relative path ➔ start from your current directory
    - Ex. test/hehe.c
- Constrains of pathname
  - Single component: ≦ 255 characters
  - Single absolute path: ≦ 1023 characters

File Tree


/
(root directory)
/bin
/dev
/etc
/sbin
/home
/lost+found
/mnt
/proc
/tcb
/tmp
/usr
/var
/stand
dsk
rdsk
term
pts
rmt
auth
init.d
rc2.d
skel
default
rc0.d
rc3.d
bin
include
local
share
games
lib
sbin
ucb
X11
bin
man
adm
mail
news
preserve
tmp
cron
netls
opt
spool
yp
cron
lock
lp
mail
mqueue
uucp
uucppublic

Layout of File Systems (1)

- hier(7)

pathname	Contents
/	The root directory of the file system
/bin & /sbin	User utilities & system programs fundamental to both single-user and multi-user environments
/usr	User utilities and applications
/usr/bin & /usr/sbin	Local executable
/lib	Shared and archive libraries
/libexec	Critical system utilities needed for binaries in /bin and /sbin
/mnt	Empty directory commonly used by system administrators as a temporary mount point
/tmp	Temporary files that are not guaranteed to persist across sys- tem reboots, also, there is /var/tmp
/usr/lib	Support libraries for standard UNIX programs
/usr/libexec	System daemons & system utilities (executed by other programs)
/usr/include	Libraries Header files
/usr/local	local executables, libraries, etc

Layout of File Systems (2)

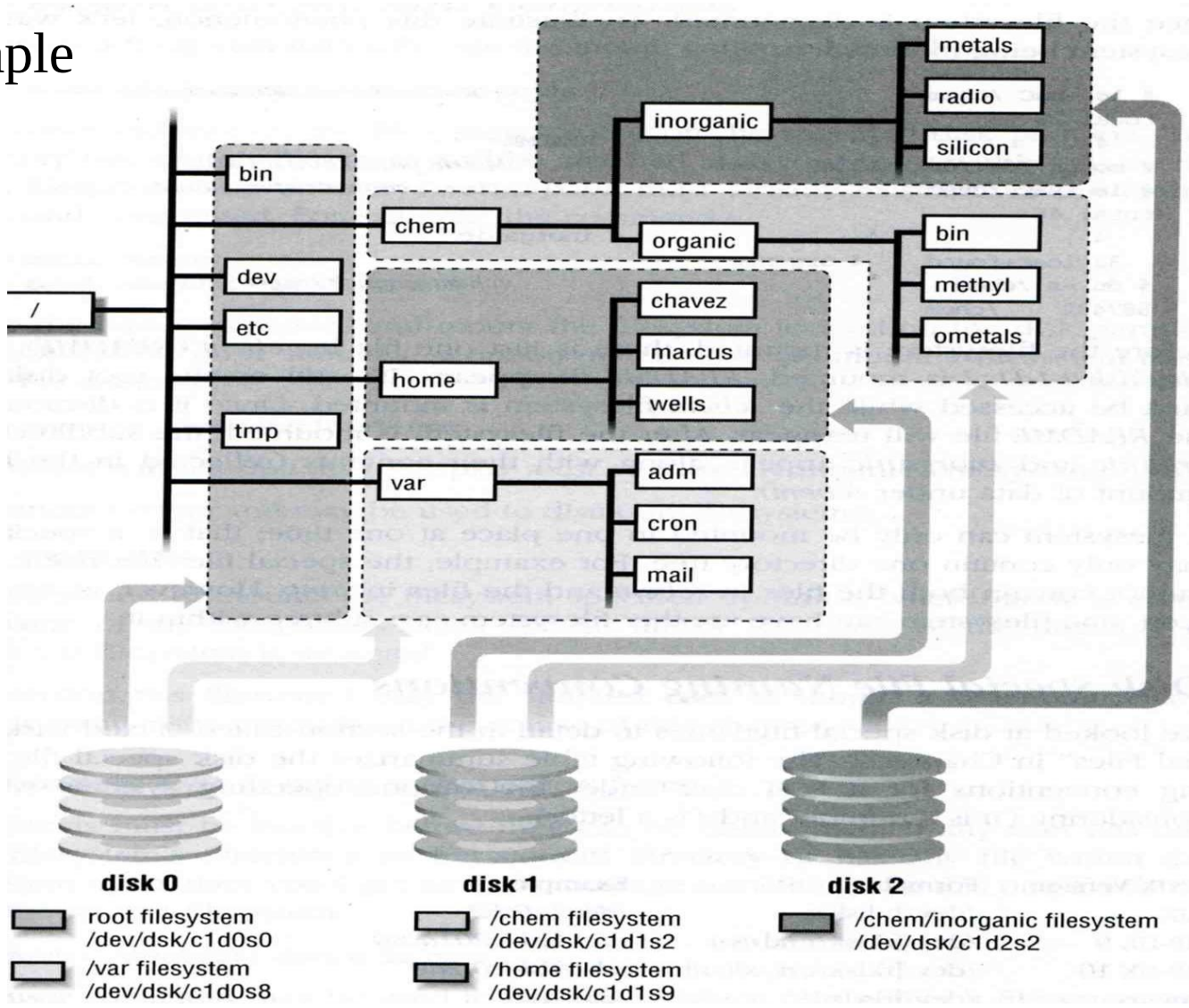
pathname	Contents
/usr/src	BSD, third-party, and/or local source files
/usr/obj	architecture-specific target tree produced by building the /usr/src tree
/etc	system configuration files and scripts
/usr/local/etc	/etc of /usr/local, mimics /etc
/dev	Device entries for disks, terminals, modems, etc
/proc	Images of all running process
/var	Multi-purpose log, temporary, transient, and spool files
/var/db	Database files
/var/db/pkg & /var/db/ports	Ports Collection management files. ports(7)
/var/log	Various system log files
/var/mail	user mailbox files
/var/spool	Spooling directories for printers, mails, etc

Mounting file system (1)

- mount(8)
- The filesystem in composed of chunks
  - Most are disk partitions
  - Network file servers
  - Memory disk emulators
  - Kernel components
  - Etc,…
- "mount" command
  - Map the mount point of the existing file tree to the root of the newly attached filesystem
  - % mount /dev/ad2s1e /home2
  - The previous contents of the mount point become inaccessible

Mounting file system (2)

- Example



Mounting file system (3)

- fstab(5)
- Filesystem table – fstab
  - Automatically mounted at boot time
  - /etc/fstab
    - Filesystem in this file will be checked and mounted automatically at boot time

Ex.

```
# Device        Mountpoint  FStype  Options      Dump  Pass#
/dev/ad0s1a     /           ufs     rw           1     1
/dev/ad0s1b     none        swap    sw           0     0
/dev/ad0s1d     /home       ufs     rw           2     2
/dev/acd0       /cdrom      cd9660  ro,noauto    0     0
csduty:/bsdhome /bsdhome    nfs     rw,noauto    0     0
```

Mounting file system (4)

- umount(8)
- Unmounting File Stsyem
  - "umount" command
    - % umount { node | device }
      - Ex: umount /home, umount /dev/ad0s1e
  - Busy filesystem
    - Someone's current directory is there or there are opened files
    - Use "umount -f"
    - We can use "lsof" or "fstat" like utilities to figure out who makes it busy

Mounting file system (5)

- fstat

```
liuyh@NASA ~ $ fstat
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DV R/W
liuyh    fstat      94218   wd /        234933 drwxr-xr-x      16   r
root     screen     87838    4 /tmp       9947 prwx------       0   r
```

- lsof (/usr/ports/sysutils/lsof) – list open files

```
liuyh@NASA ~ $ lsof
COMMAND   PID USER   FD TYPE SIZE/OFF    NODE NAME
screen  87838 root  cwd VDIR        7  522069 /usr/ports/sysutils/screen
screen  87838 root  rtd VDIR       26       3 /
screen  87838 root  txt VREG   337968  424757 /usr/local/bin/screen
screen  87838 root  txt VREG   245976  679260 /libexec/ld-elf.so.1
screen  87838 root  txt VREG   314504  678109 /lib/libncurses.so.8
screen  87838 root  txt VREG    64952  678438 /lib/libutil.so.8
screen  87838 root  txt VREG    33536  677963 /lib/libcrypt.so.5
screen  87838 root  txt VREG  1255568  677294 /lib/libc.so.7
```

File Types (1)

❑File types

Symbol	File types
-	Regular file
b	Block device file
c	Character device file
d	Directory
l	Symbolic link
s	UNIX domain socket
p	Named pipe

❑**file** command

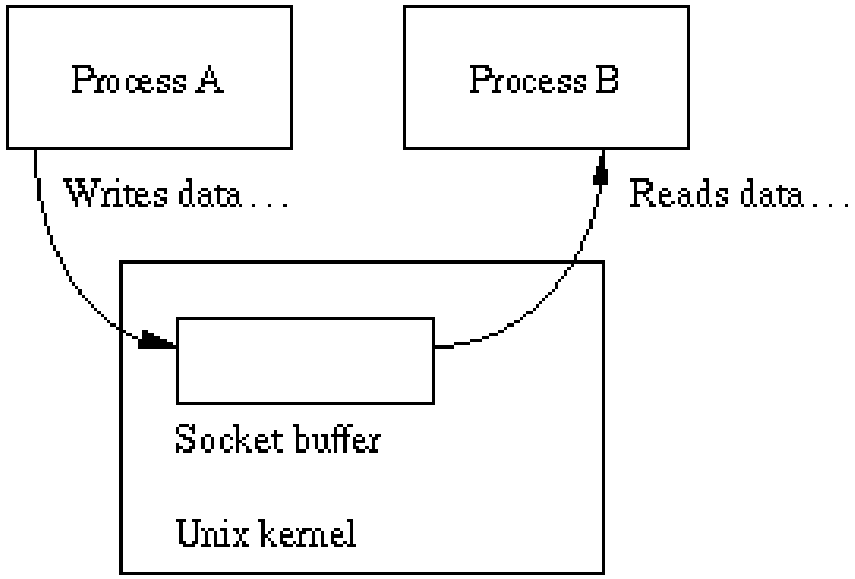
- determine file type
  - % file .tcshrc ➔ .tcshrc: ASCII text
  - % file /bin ➔ /bin: directory
  - % file /bin/sh ➔ /bin/sh: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically linked (uses shared libs), stripped
- /usr/ports/sysutils/file

File Types (2)

- Directory
  - . and ..
  - mkdir / rmdir

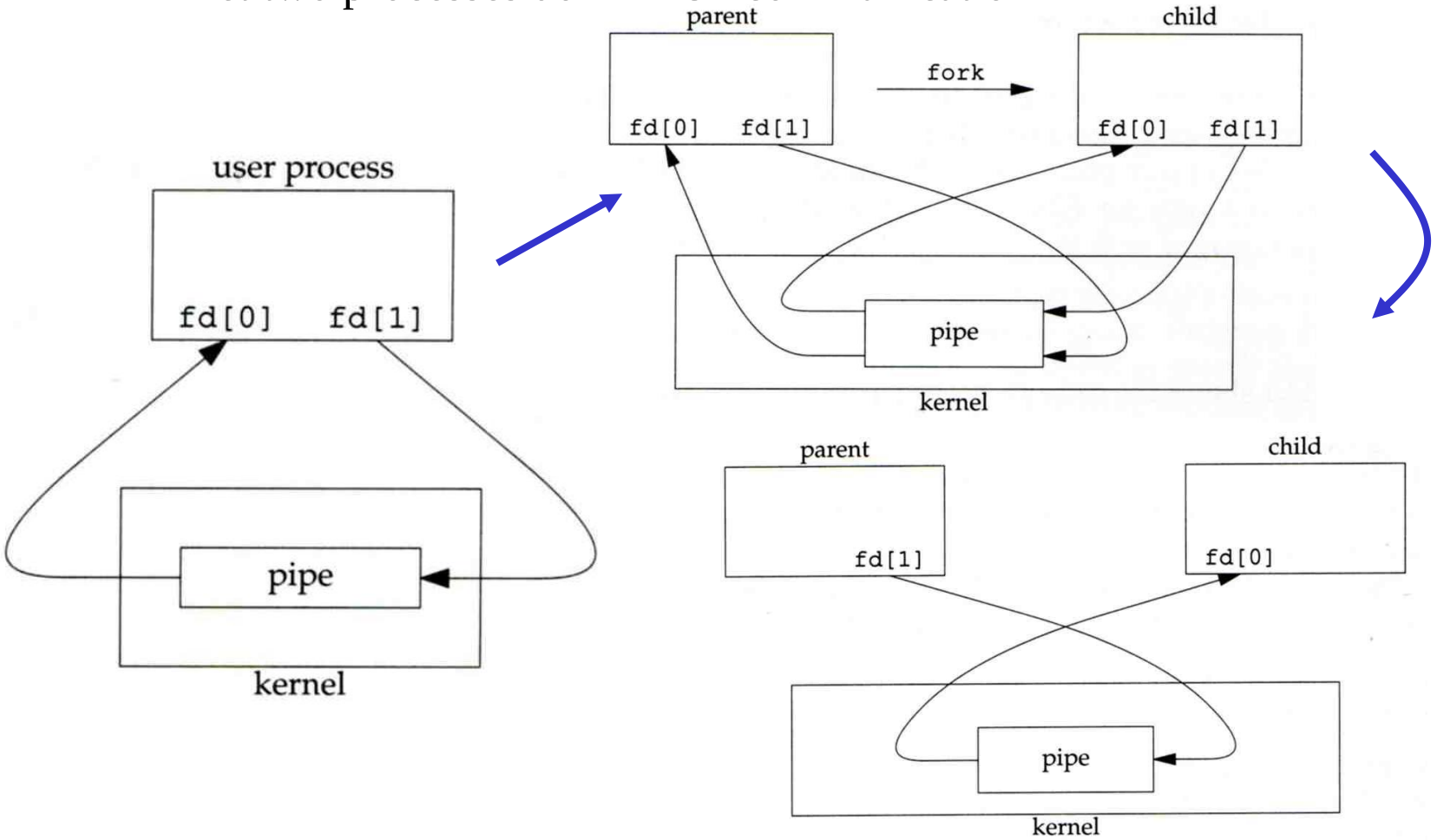
File Types (3)

- UNIX domain socket
  - Created by socket()
  - Local to a particular host
  - Be referenced through a filesystem object rather than a network port



File Types (4)

- Named Pipes
  - Let two processes do "FIFO" communication



File Types (5)

- Named Pipe
  - $ mkfifo [-m mode] fifo_name ...

```
$ mkfifo pipe
$ du >> pipe
(another process)
$ sort -n pipe
```

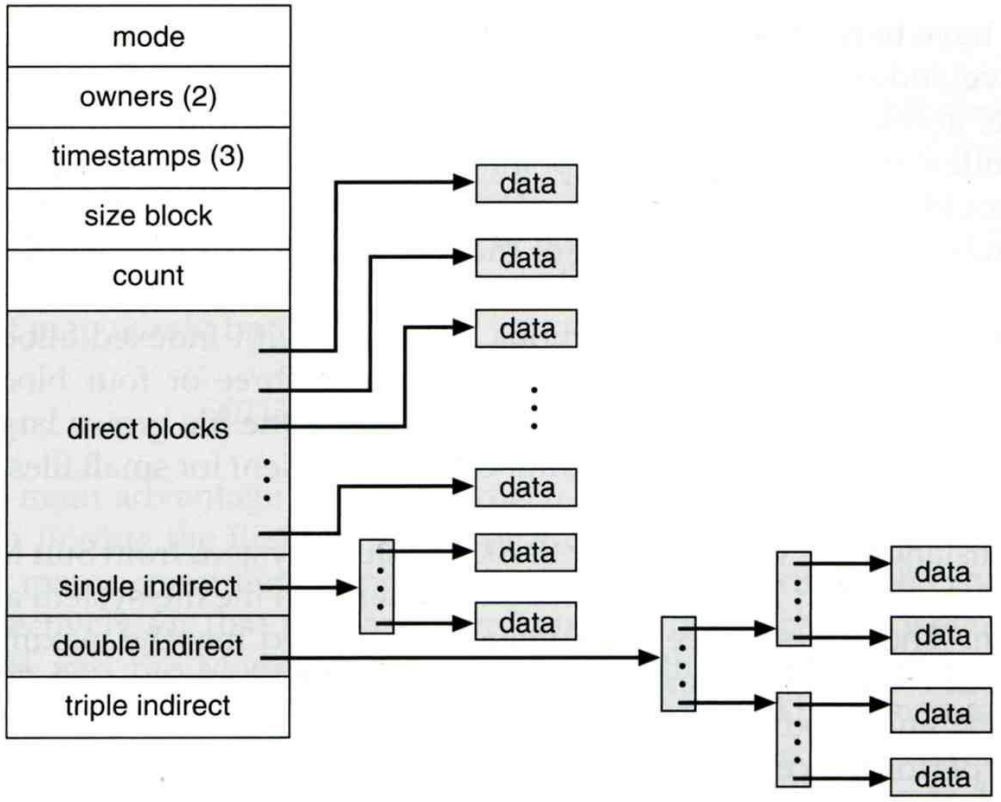
File Types (6)

- Symbolic Link
  - A file which points to another pathname
  - % ln -s ori-file soft-file
  - Like “short-cut” in Windows

inode and file (1)

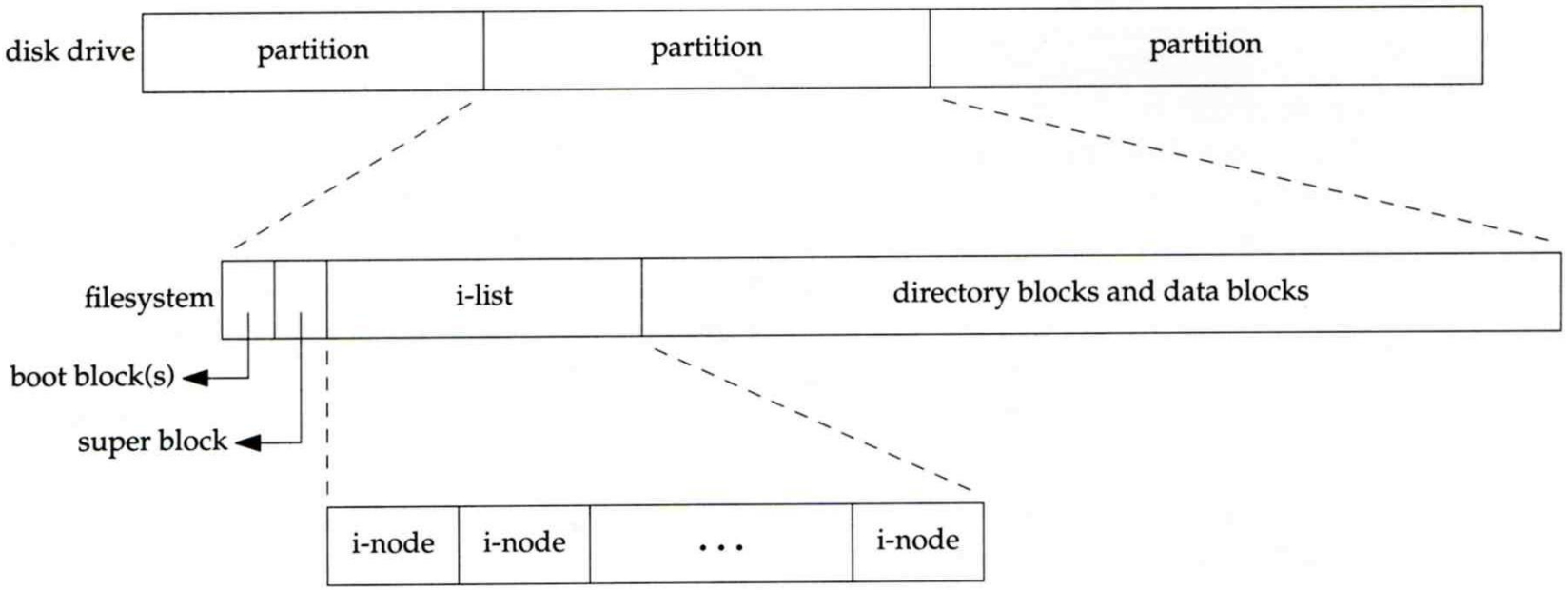
- inode
  - A structure that records information of a file
    - You can use "ls -i" to see each file's inode number

```
liuyh@NASA ~ $ ls -i
19255327 public_html
```



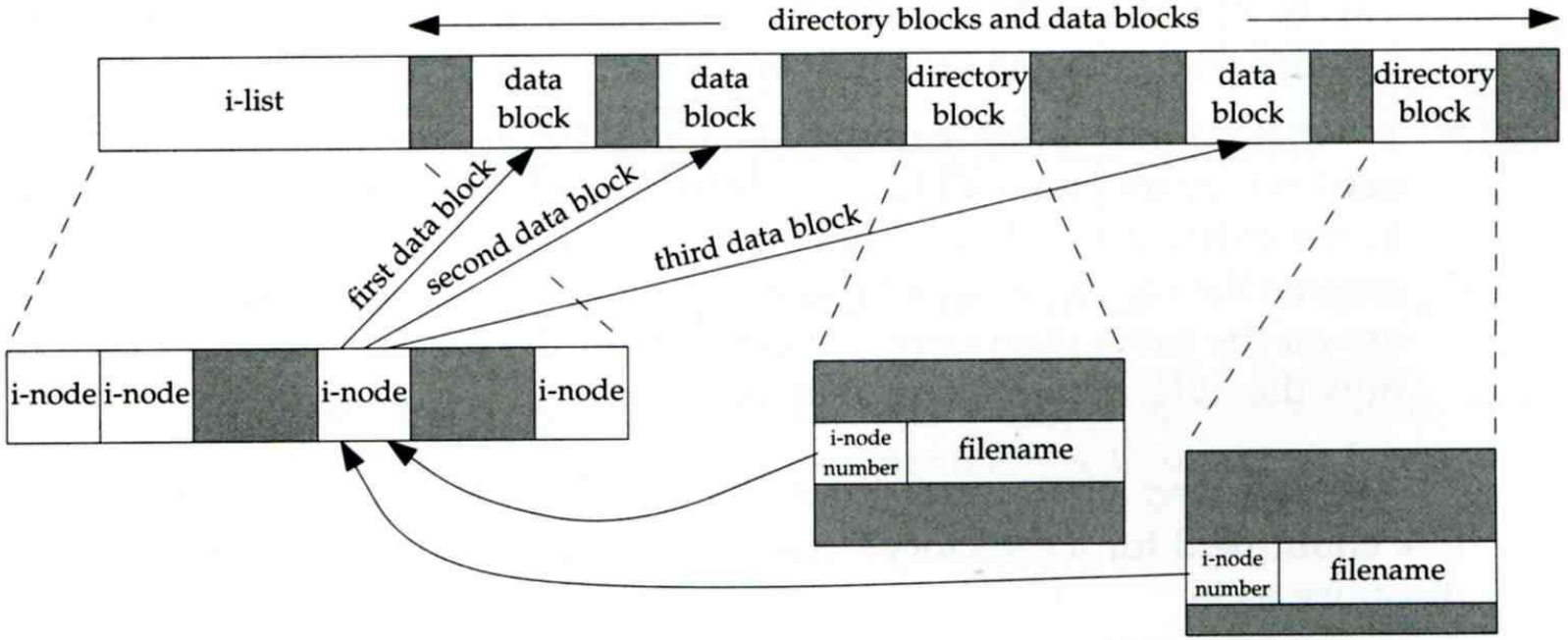
inode and file (2)

- Filesystem
  - Boot blocks
  - Super block
  - Inode list
  - Data block



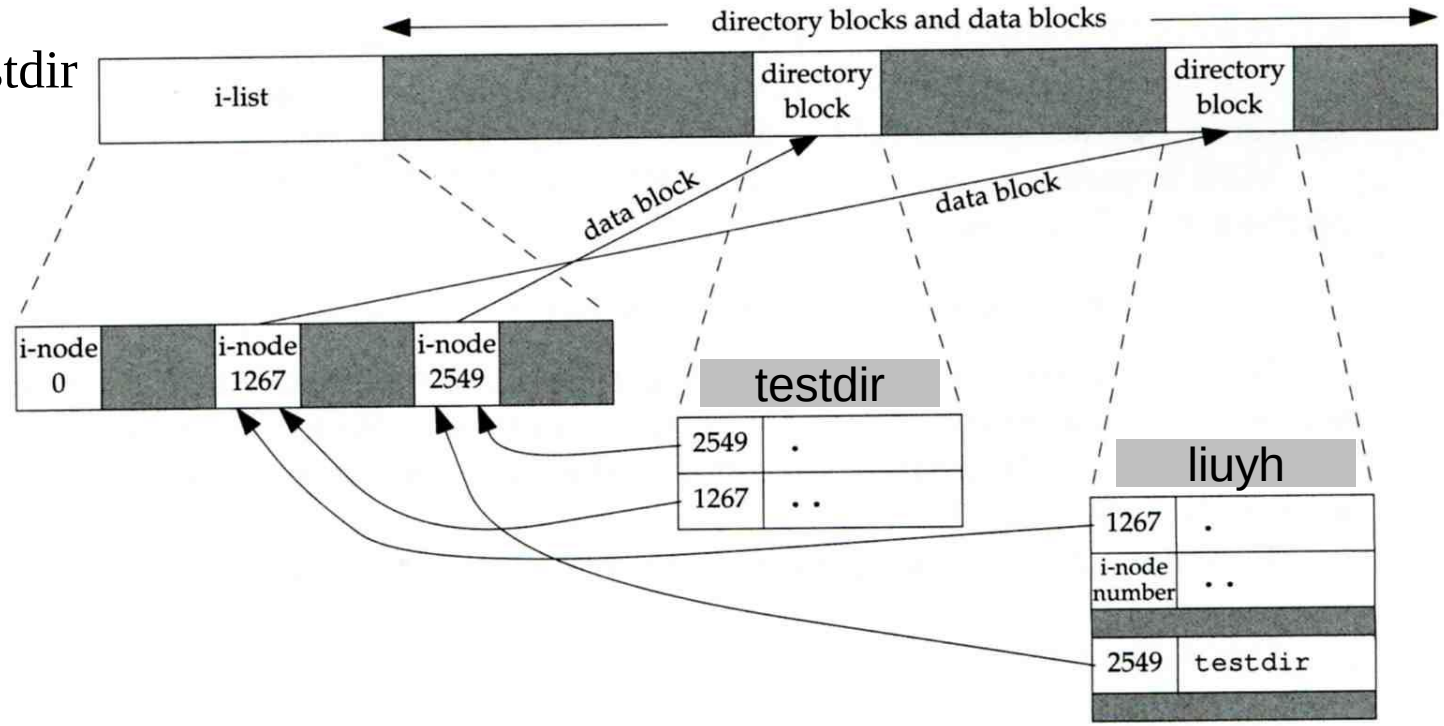
inode and file (3)

- More detail of inode and data block



inode and file (4)

- Example
  - .
  - ..
  - testdir

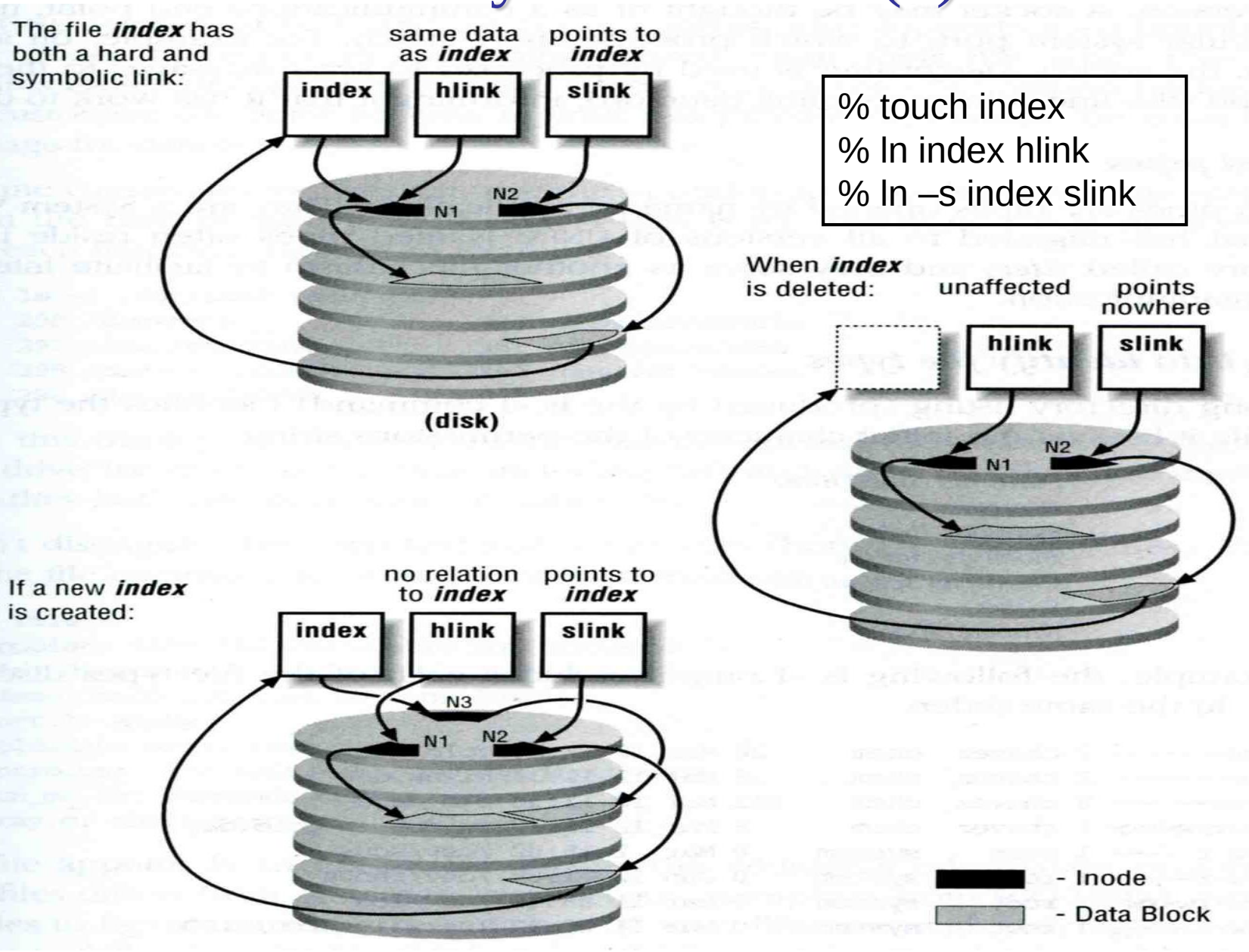


/home/liuyh/testdir

Hard Link V.S. Symbolic Link (1)

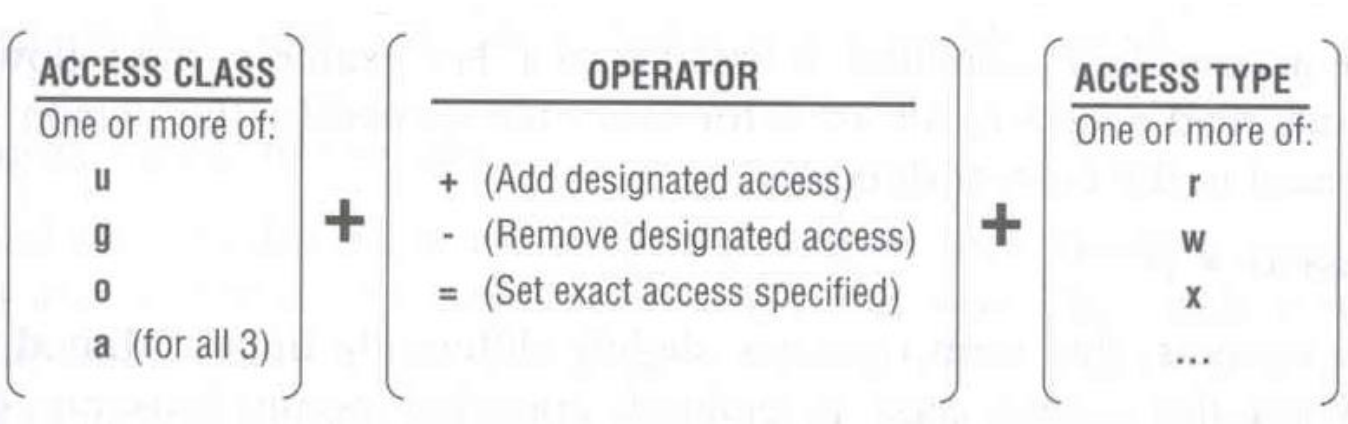
- Link
  - Hard link
    - associate two or more filenames with the same inode
      - Must in the same partition
    - % ln ori-file hard-file
  - Soft (symbolic) link
    - A file which points to another pathname
    - % ln -s ori-file soft-file

Hard Link V.S. Symbolic Link (2)



File Access Mode (1)

- rwx r-x r-x
  - User, group, other privileges
- chmod command
  - chmod(1), "MODES" section
  - % **chmod** *access-string* *file*
    - **% chmod u+x test.sh**
    - **% chmod go-w .tcshrc**
    - **% chmod u+w,g-w hehe haha**
    - **% chmod −R 755 public_html/**



File Access Mode (2)

- setuid, setgid, sticky bit
  - setuid, setgid on file
    - The effective uid/gid of resulting process will be set to the UID/GID of the file
    - setuid
      - passwd, chsh, crontab
    - setgid
      - top, fstat, write
  - setgid on directory
    - Cause newly created files within the directory to be the same group as directory
  - sticky on directory (/tmp)
    - Do not allow to delete or rename a file unless you are
      - The owner of the file
      - The owner of the directory
      - root

File Access Mode (3)

- Decimal argument of chmod
  - setuid: 4000
  - setgid: 2000
  - stiky : 1000

Mode	Attribute	Mode	Attribute
755	- rwx r-x r-x	644	- rw- r-- r--
4755	- rws r-x r-x	600	- rw- --- ---
2755	- rwx r-s r-x	400	- r-- r-- r--
2775	d rwx rws r-x	1777	d rwx rwx rwt
755	d rwx r-x r-x	4555	- r-s r-x r-x
750	d rwx r-x ---	711	- rwx --x --x
700	d rwx --- ---	711	d rwx --x --x

File Access Mode (4)

- Assign default permissions: umask
  - Shell built-in command
  - Inference the default permissions given to the files newly created.
  - The newly created file permission:
    - Use full permission bit (file: 666, dir: 777) xor umask value.
  - Example:

umask	New File	New Dir
022	- rw- r-- r--	d rwx r-x r-x
033	- rw- r-- r--	d rwx r-- r--
066	- rw- --- ---	d rwx --x --x
000	- rw- rw- rw-	d rwx rwx rwx
477	- r-- --- ---	d r-x --- ---
777	- --- --- ---	d --- --- ---

File Protection

Command	Minimum Access Needed	
	On file itself	On directory file is in
cd /home/test		x
ls /home/test/*.c		r
ls -s /home/test/*.c		rx
cat runme	r	x
cat >> runme	w	x
run-binary	x	x
run-script	rx	x
rm rumme		wx

Changing File Owner

- Changing File Owner
  - Commands:
    - chown -- change user owner
    - chgrp -- change group owner
- Change the file ownership and group ownership
  - % chown -R liuyh /home/liuyh
  - % chgrp -R cs /home/liuyh
  - % chown -R liuyh:dcs /home/liuyh
  - % chown -R :dcs /home/liuyh

FreeBSD bonus flags

chflags(1)

- chflags command
  - schg — system immutable flag — (root only)
  - sunlnk — system undeletable flag — (root only)
  - sappnd — system append-only flag — (root only)
  - uappend — user append-only flag — (root, user)
  - uunlnk — user undeletable flag — (root, user)
  - …
- ls -ol

```
liuyh@NASA ~ $ ls -ol /libexec/
total 1034
-r-xr-xr-x  1 root  wheel  schg 238472 Sep 21 12:50 ld-elf.so.1*
-r-xr-xr-x  1 root  wheel  -    238512 Jul 24 17:15 ld-elf.so.1.old
-r-xr-xr-x  1 root  wheel  schg 212204 Sep 21 12:51 ld-elf32.so.1
-r-xr-xr-x  1 root  wheel  -    212248 Jul 24 17:17 ld-elf32.so.1.old
```