

# File System

tsaimh (2022-2024, CC BY-SA)

jnlin(2019-2021, CC BY-SA)

? (1996-2018)

國立陽明交通大學資工系資訊中心

Information Technology Center of Department of Computer Science, NYCU

# Handbook and Manual pages

- Official guide and be found at
  - <https://www.freebsd.org/doc/en/books/handbook/permissions.html>

# Files

- \$ ls -l

```
drwx--x--x  7 tsaimh  dcs          1024 Sep 22 17:25 public_html
```

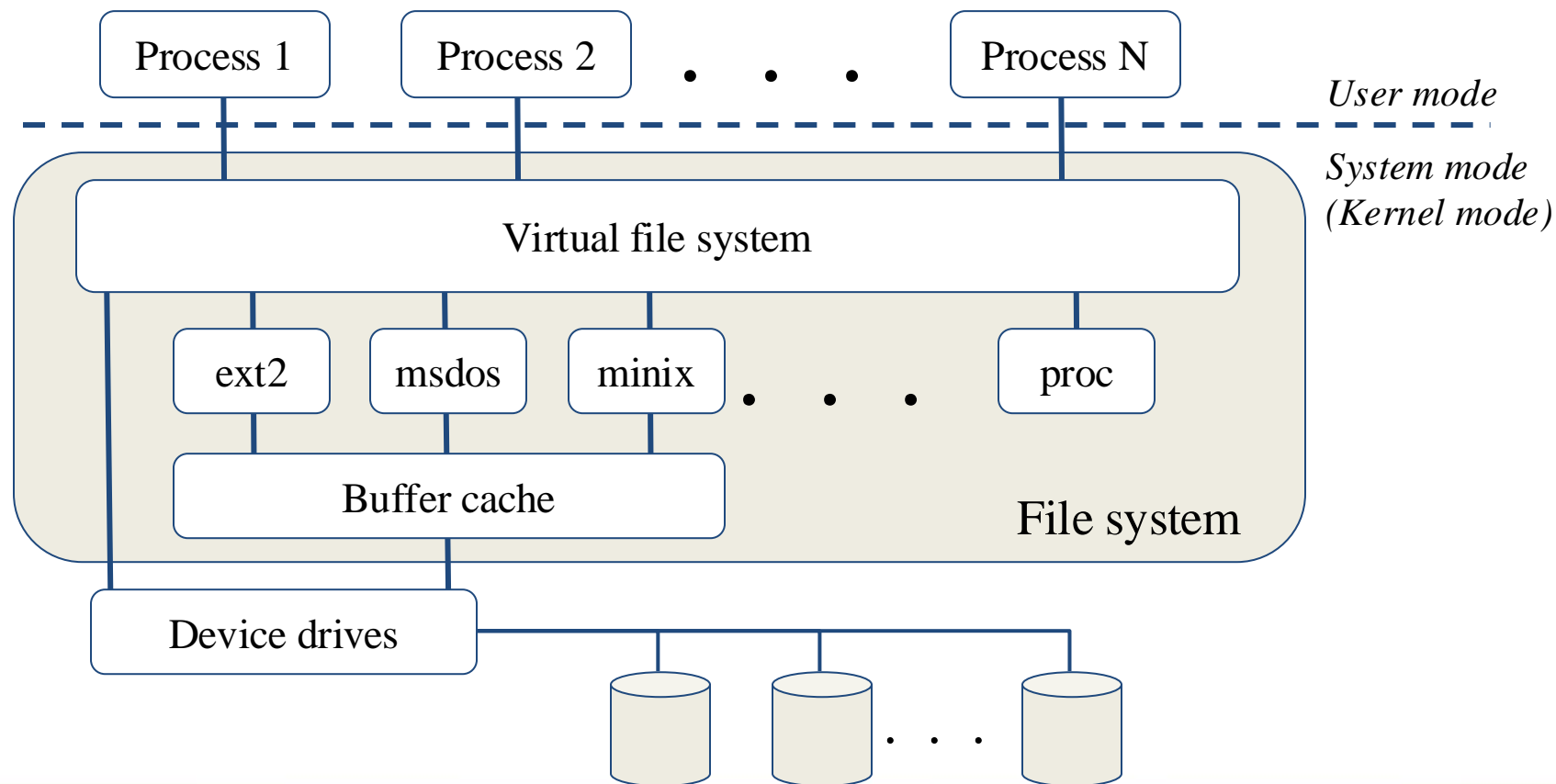
File Type	d
File Access Mode	rwX--X--X
inodes	7
File User Owner	tsaimh
File Group Owner	dcs
File Size	1024
File Last Modify Time	Sep 22 17:25
File Name	public_html

# 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 (3)

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

```
$ cp file1 file2
```

## Example System Call Sequence

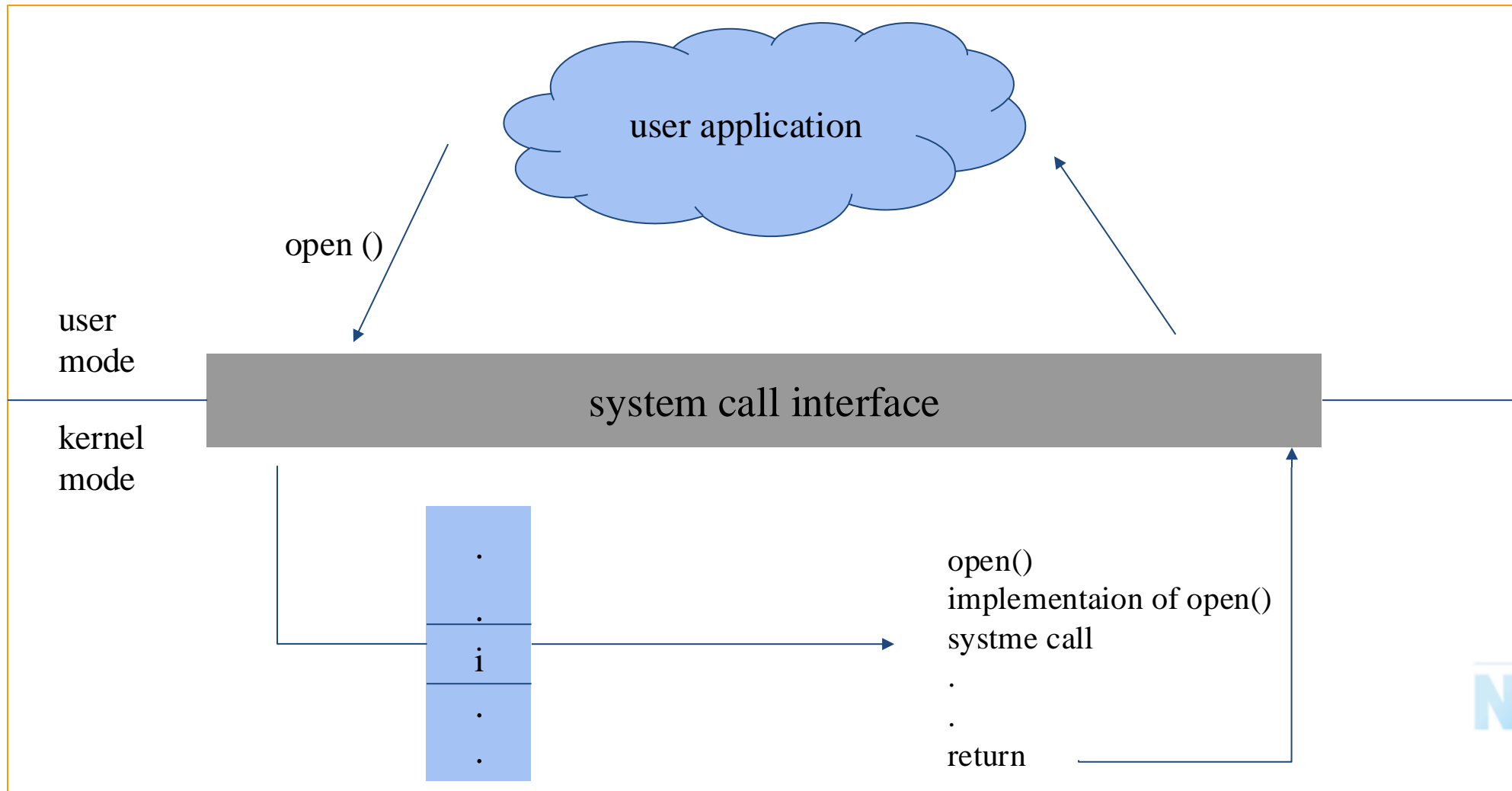
```
Open the input file
    if file doesn't exist, abort
Create output file
    if file exists, overwrite it
Loop
    Read from input file
    Write to output file
Until read EOF
Close input and output files
Write completion message to screen
Terminate normally
```

Source file

destination file

# File System Architecture (4)

## □ API – System Call – OS Relationship





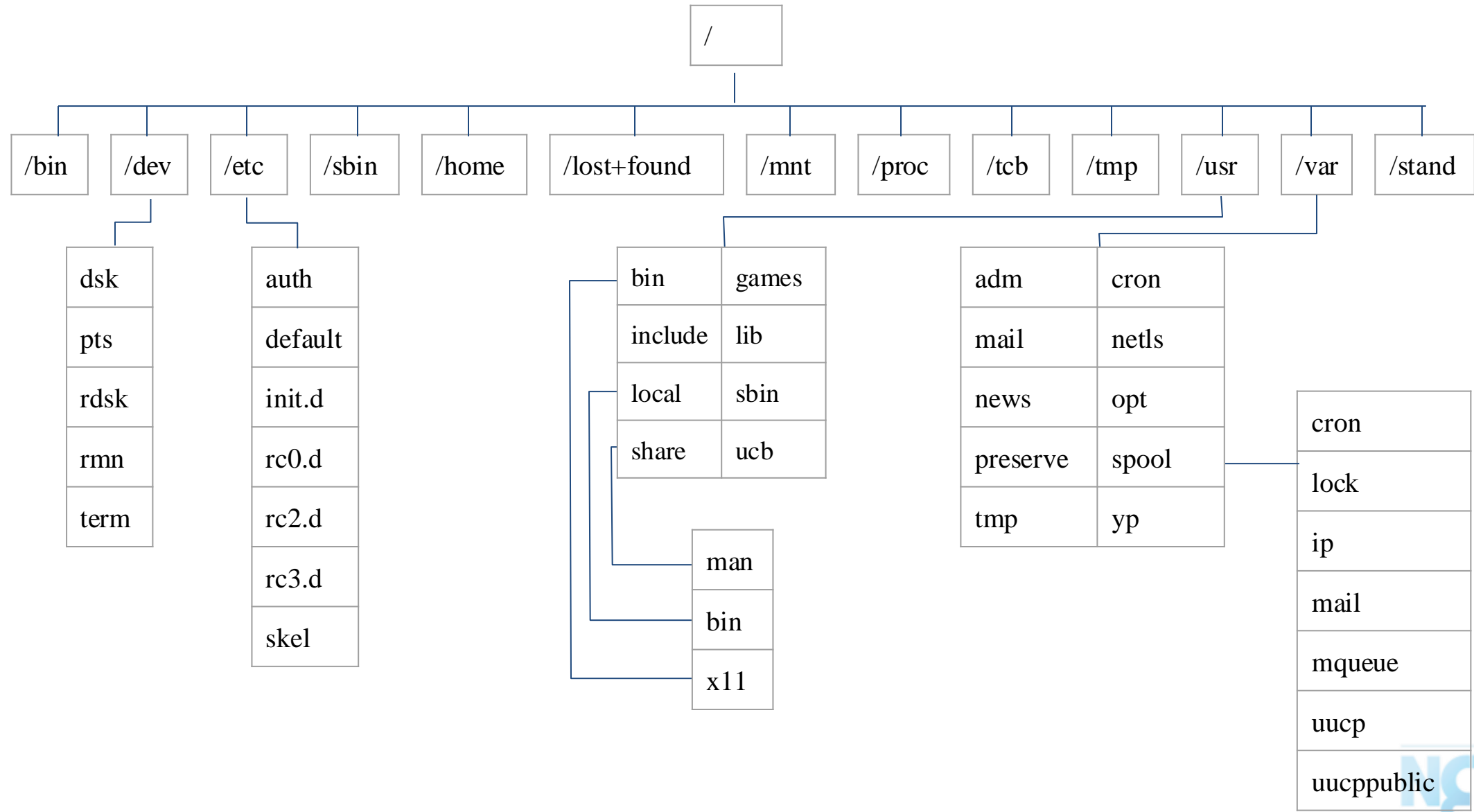
# File System Architecture (5)

- 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 file system interface to access such "object"
    - open, read, write, close, seek, ioctl, fcntl, ...

# Pathname

- Two kinds of path
  - Absolute path → start from /
    - E.g. /net/dcs/93/9317807/test/haha.c
  - Relative path → start from your current directory
    - E.g. test/haha.c
- Constraints of pathname
  - Single component:  $\leq 255$  characters
  - Single absolute path:  $\leq 1023$  characters

# File Tree



# Layout of File Systems (1)

- [hier\(7\)](#)

Path Name	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 system 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

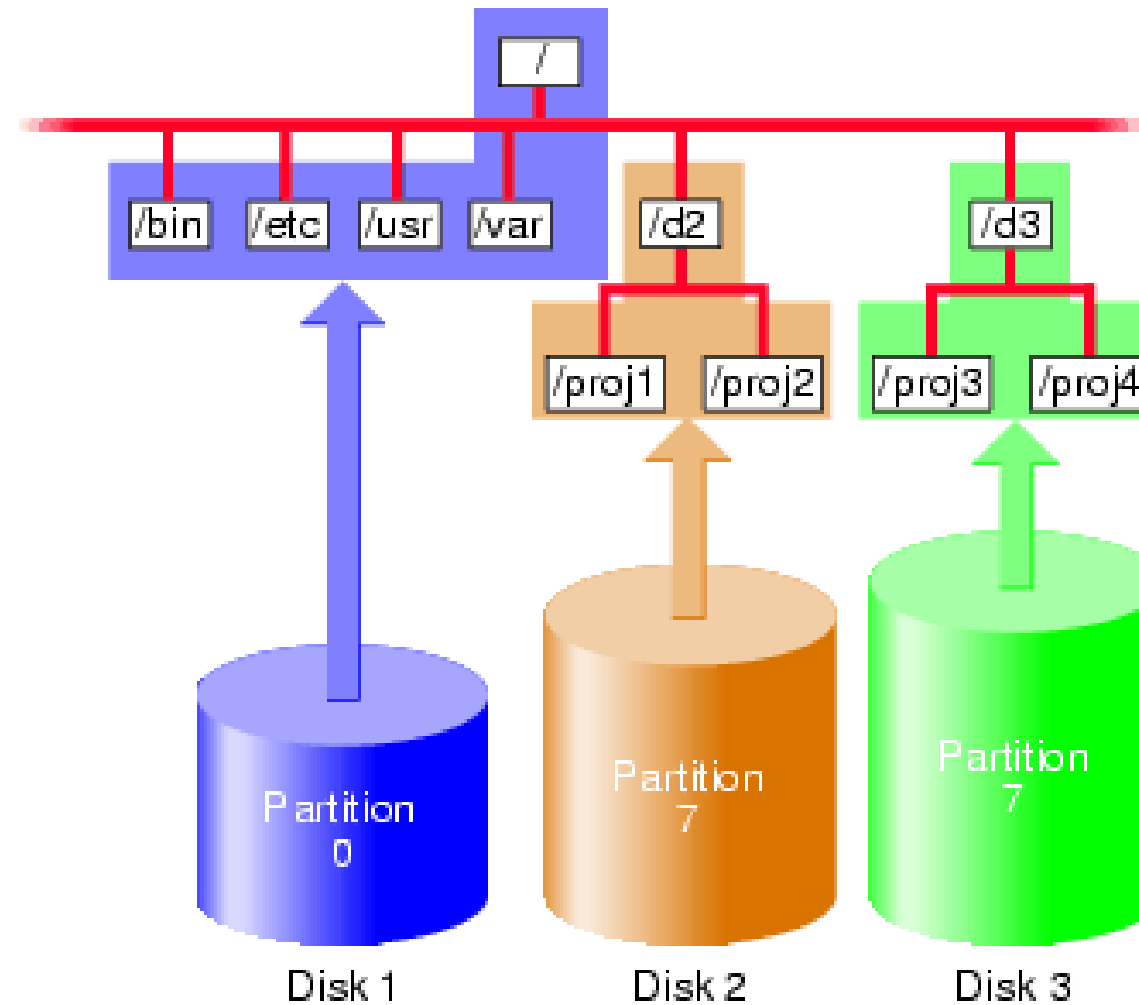
# Layout of File Systems (2)

Path Name	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\(8\)](#)
- Common types of file systems
  - 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)



# 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

E.g.

#	Device	Mountpoint	FStype	Options	Dump	Pass#
	/dev/ad0s1a	/	ufs	rw	1	1
	/dev/ad0s1b	none	swap	sw	0	0



# Mounting file system (4)

- [umount\(8\)](#)
- Unmounting file system
  - "umount" command
    - \$ umount { node | device }
    - Ex: 

umount /home	umount /dev/ad0s1e
--------------	--------------------
  - Busy file system
    - 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\(1\)](#)

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

- [lsof\(8\)](#) (/usr/ports/sysutils/lsof) – list open files

```
$ 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
```

# 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 Types (2)

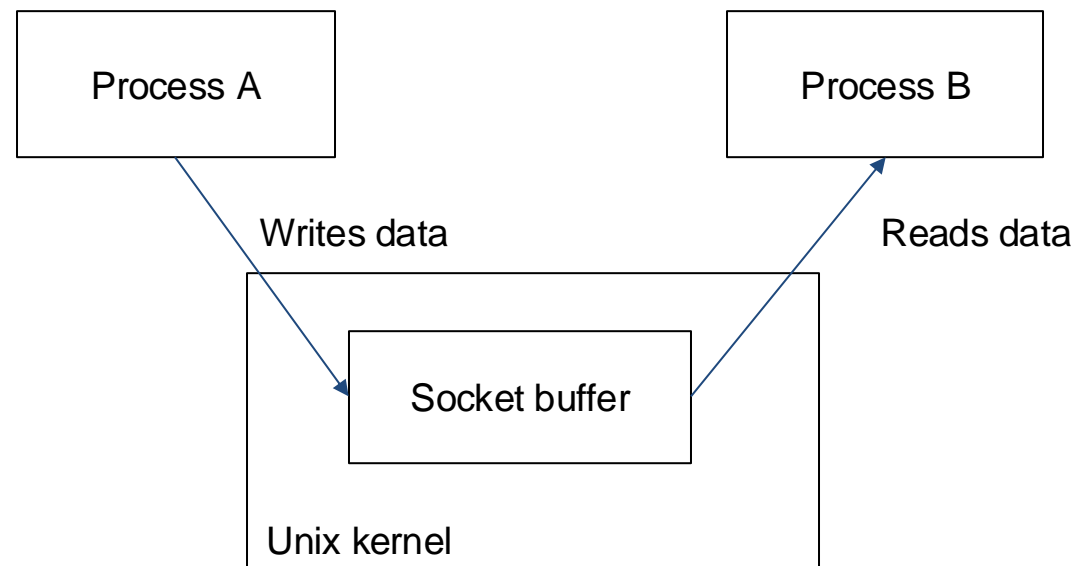
- file command
  - determine file type
    - `$ file .tcshrc`  
`.tcshrc: ASCII text`
    - `$ file /bin`  
`/bin: directory`
    - `$ file /bin/sh`  
`/bin/sh: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically linked (uses shared libs), stripped`
  - `/usr/ports/sysutils/file`

# File Types (3)

- Directory
  - . and ..
  - mkdir / rmdir

# File Types (4)

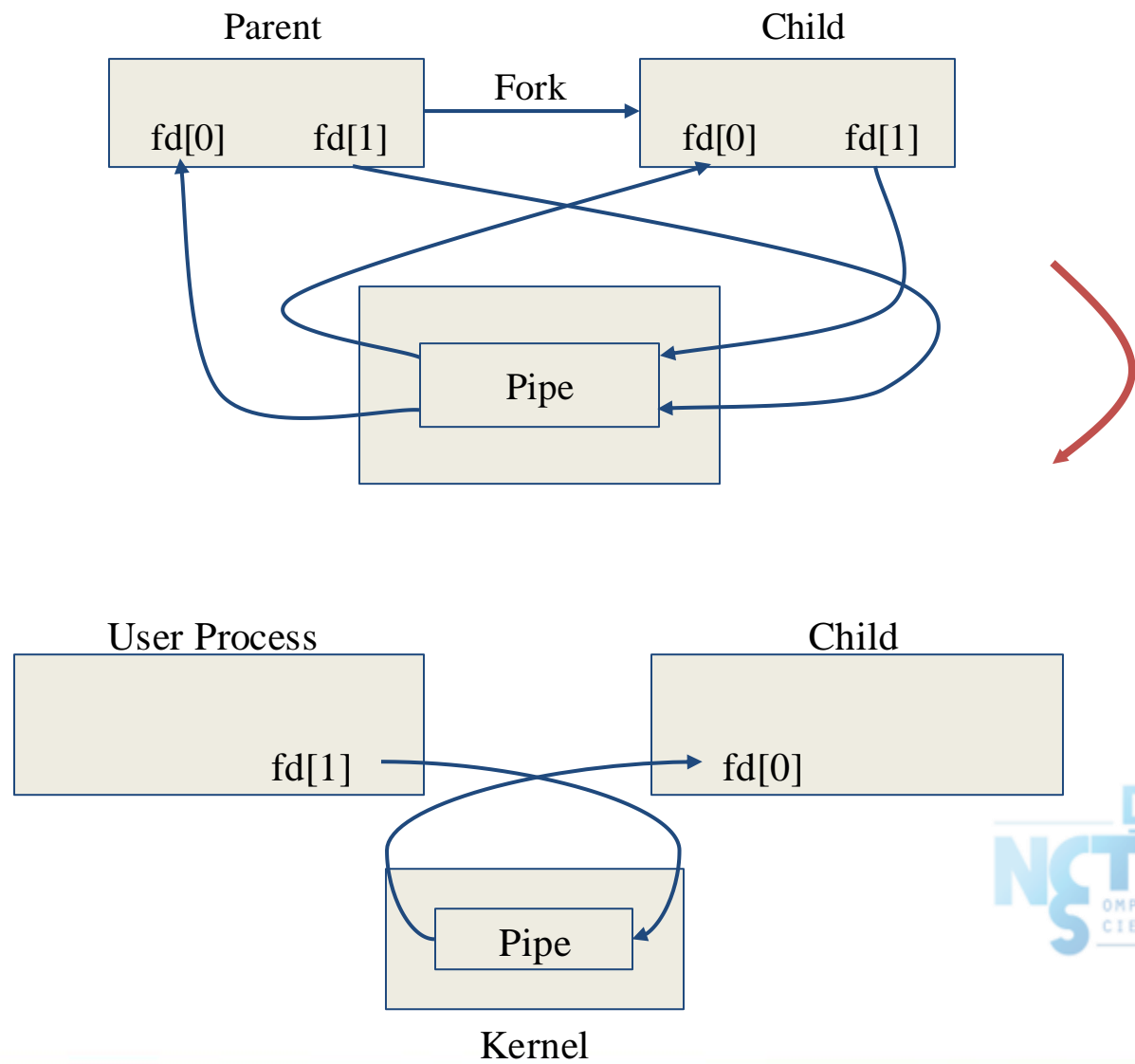
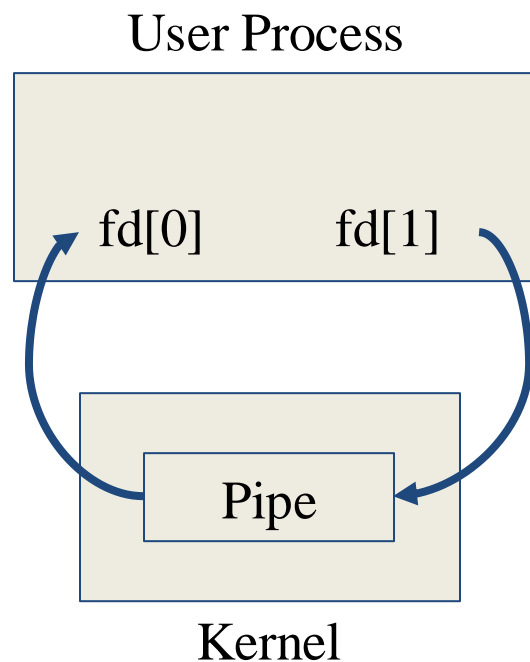
- UNIX domain socket
  - Created by `socket()`
  - Local to a particular host
  - Be referenced through a filesystem object rather than a network port



# File Types (5)

- Pipes

- Let two processes do "FIFO" communication



# File Types (6)

- Named Pipe
  - `$ mkfifo [-m mode] fifo_name ...`
  
  - `$ mkfifo pipe`
  - `$ du >> pipe`  
(another process)
  - `$ sort -n pipe`



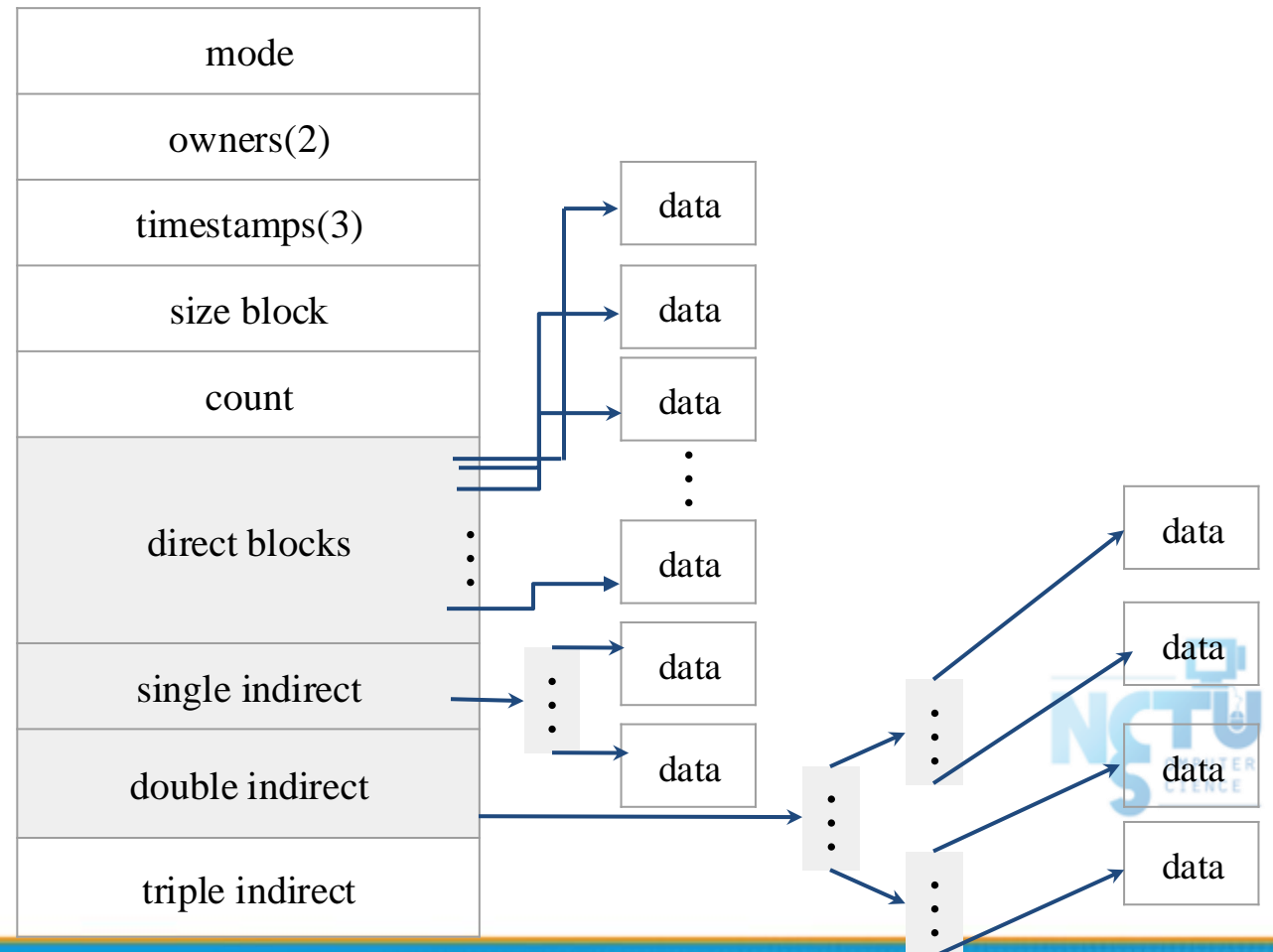
# File Types (7)

- Symbolic Link
  - A file which points to another pathname
  - `$ ln -s ori-file soft-file`
  - Like "short-cut" in Windows

# inode and file (1)

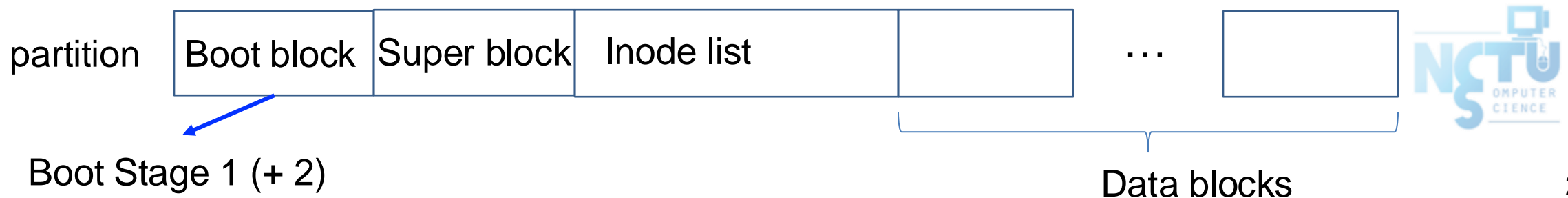
- inode
  - A structure that records information of a file
    - You can use "ls -i" to see each file's inode number

```
$ ls -i
```



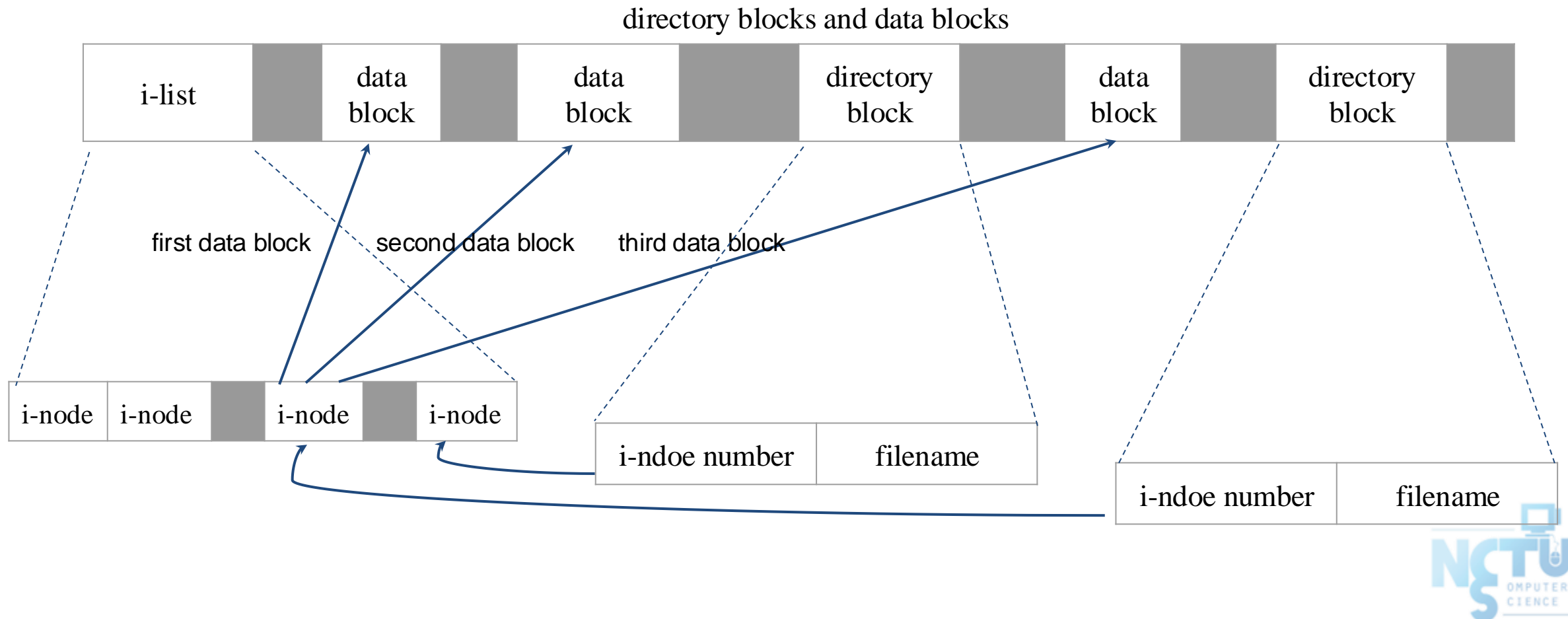
# inode and file (2)

- A Unix partition consists of four major components:
  - The **boot block** contains the bootstrap program used to load the operating system.
  - The **super block** describes the state of the file system (e.g., total size of the partition, block size, **inode number of the root directory**)
  - **Inode list** contains a linear array of inodes. While **users think of files in terms of file names, Unix thinks of files in terms of inodes.**
  - **Data blocks** containing the actual contents of files.



# inode and file (3)

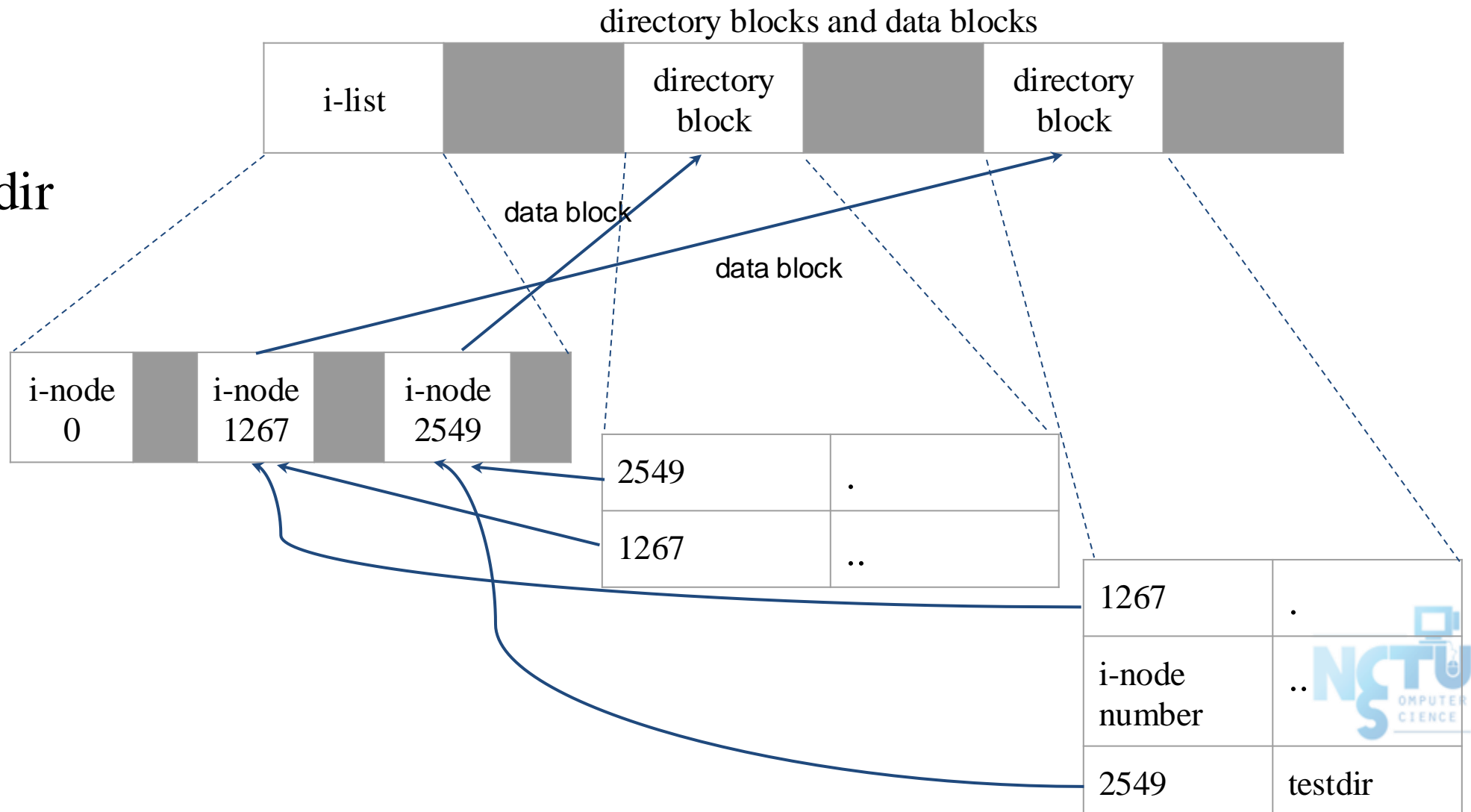
- More details of inode and data block



# inode and file (4)

- Example

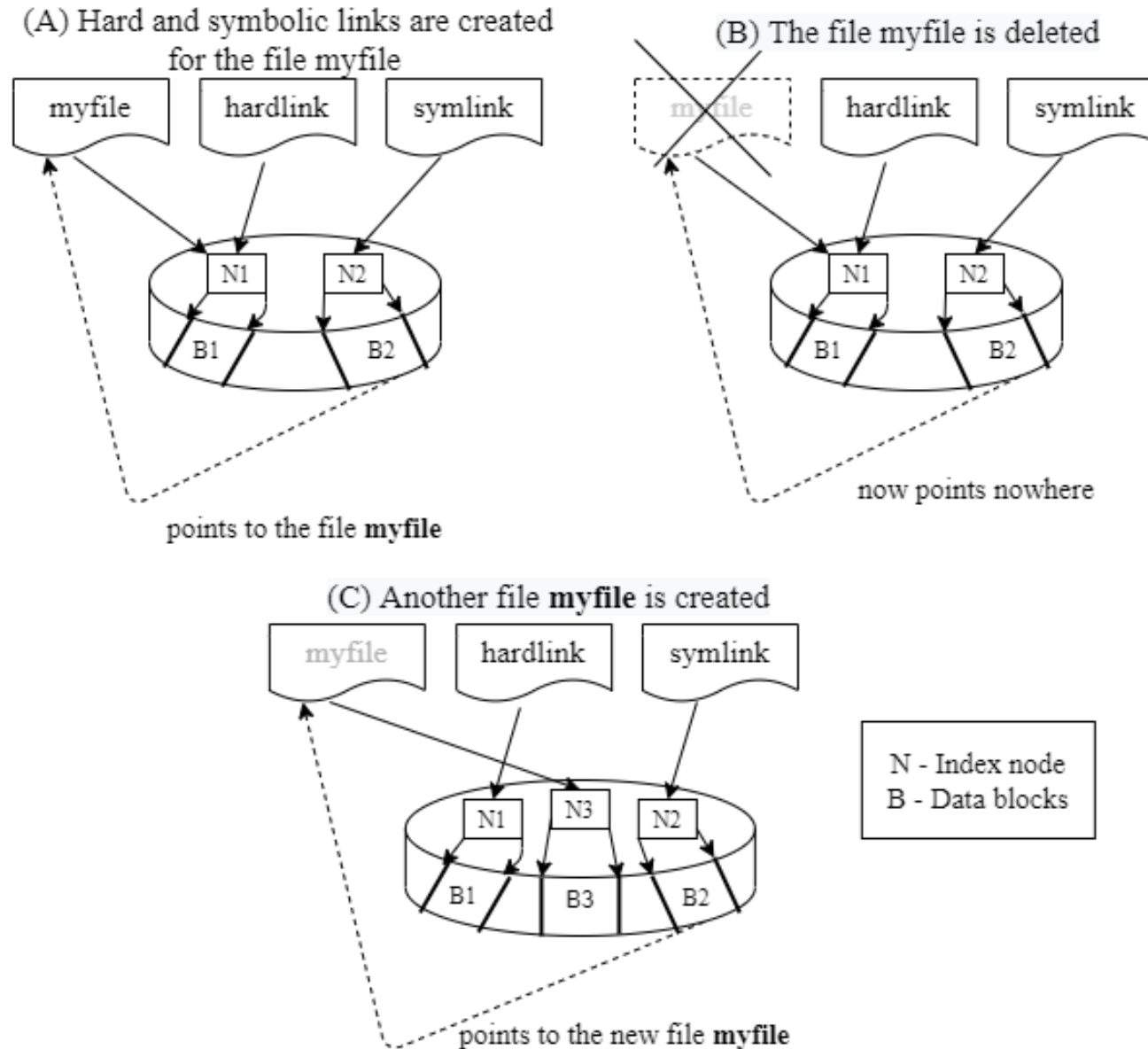
- .
- ..
- testdir



# Hard Link vs. 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 vs. Symbolic Link (2)



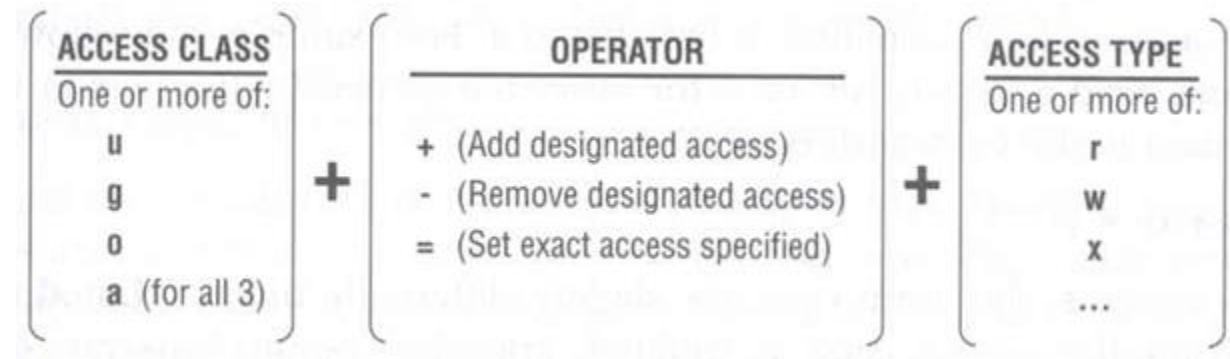
(A)  
\$ touch myfile  
\$ ln myfile hardlink  
\$ ln -s myfile symlink

(B)  
\$ rm myfile

(C)  
\$ touch myfile

# 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
      - write

# File Access Mode (3)

- 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 (4)

- Decimal argument of chmod
  - setuid: 4000
  - setgid: 2000
  - sticky : 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	444	- 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 (5)

- 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) & (!umask) value.
  - Ex:

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
277	- r-- --- ---	d r-x --- ---
777	- --- --- ---	d --- --- ---

# File Protection

Command	Minimum Access Needed	
	On file itself	On directory
<code>cd /home/test</code>		X
<code>ls /home/test</code>		r
<code>ls -s /home/test/*.c</code>		rX
<code>cat runme</code>	r	X
<code>cat &gt;&gt; runme</code>	w	X
<code>run-binary</code>	X	X
<code>run-script</code>	rX	X
<code>rm rumme</code>		wX

# Changing File Owner

- Changing File Owner
  - Commands:
    - [chown\(8\)](#) -- change user owner
    - [chgrp\(1\)](#) -- change group owner
- Change the file ownership and group ownership

```
$ chown -R tsaimh /home/tsaimh
$ chown -R tsaimh:dcs /home/tsaimh
$ chown -R :dcs /home/tsaimh
$ chgrp -R dcs /home/tsaimh
```

# FreeBSD bonus flags (1)

- [chflags\(1\)](#) 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

```
$ 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
```

# FreeBSD bonus flags (2)

```
$ ls -al > file
$ chflags uappend file
$ ls -al > file
file: Operation not permitted.
$ ls -al >> file
$ ls -ol
total 2
-rw-r--r--  1 tsaimh  dcs  uappnd 325  9  4 16:14 file
$ cat file
total 8
drwxr-xr-x  2 tsaimh  dcs  512  9  4 16:13 .
drwxr-xr-x 49 tsaimh  dcs 4608  9  4 16:13 ..
-rw-r--r--  1 tsaimh  dcs   0  9  4 16:13 file
total 10
drwxr-xr-x  2 tsaimh  dcs  512  9  4 16:13 .
drwxr-xr-x 49 tsaimh  dcs 4608  9  4 16:13 ..
-rw-r--r--  1 tsaimh  dcs  162  9  4 16:13 file
```



# Appendix

國立陽明交通大學資工系資訊中心

Information Technology Center of Department of Computer Science, NYCU

# Journaling File System

- Write operational logs to the journal first, then commit it asynchronously.
- If system crashed, check the log
  - fully committed: skip
  - partial committed: rollback or commit
  - non-committed: ignore or commit
- Reduce "fsck" time and data inconsistency
- Example
  - ext3, ext4
  - xfs
  - btrfs

# CoW (Copy on Write) File System

- If some data is copied but not modified, they will be referred to the same physical address in the storage
- Pros
  - Reduce the space used
  - Low cost snapshots
- Cons
  - Data inconsistency (for example, the reference count is not consistent)
  - Not "real" used space on file
- Example
  - ZFS deduplication

# File Attribute Extension

- Associate files with metadata not interpreted by the filesystem
- Key-value pairs, saved in the inode
- Example
  - mime\_type
  - md5/sha1 checksum
  - security attributes