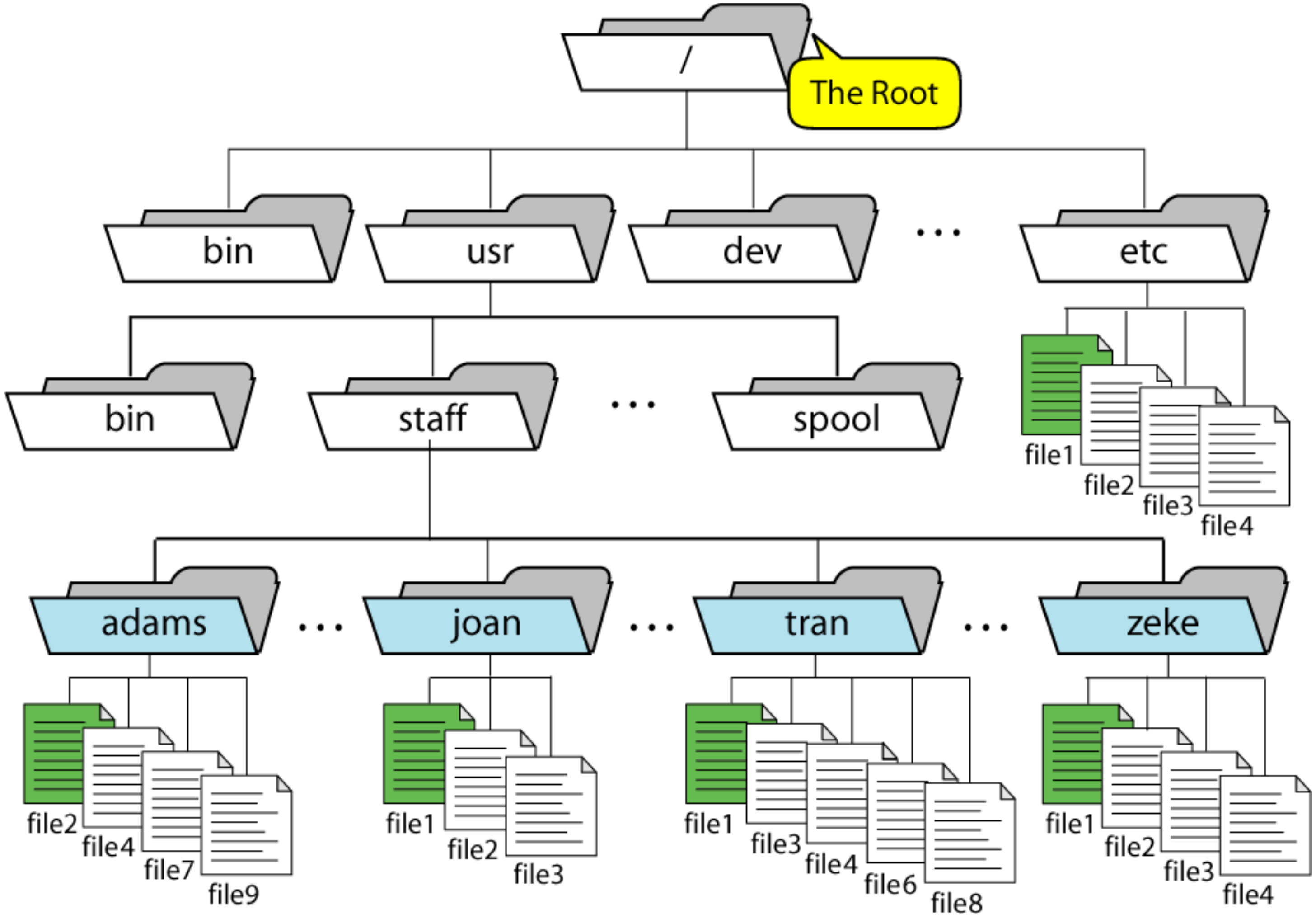
The Linux File System

File and Directory

- In linux and most of OS data are stored in files
- File
  - Contains data
  - Stored in (hard) disk
- Directory
  - Contains files
  - Stored in (hard) disk
  - Makes easy for data organizing
- Hierarchy of directories and files = file system
- Single file system for all logical disks

Figure 3-3

A Directory Hierarchy



Directory Types

- Root Directory: /
  - The first directory in any UNIX file structure
  - Always begin with the forward slash (/)
- Home Directory: $HOME or ~
  - Created by system administrator
  - This is where you are when you first log in!
  - Under $HOME, you may create your own directory structure
  - Type: cd [Return] takes you $HOME
- Current Working Directory: .
  - The Directory you are <u>currently working in</u>
  - Also called Current Working Directory (cwd)
- Parent Directory: ..
  - The directory immediately above your current working directory.

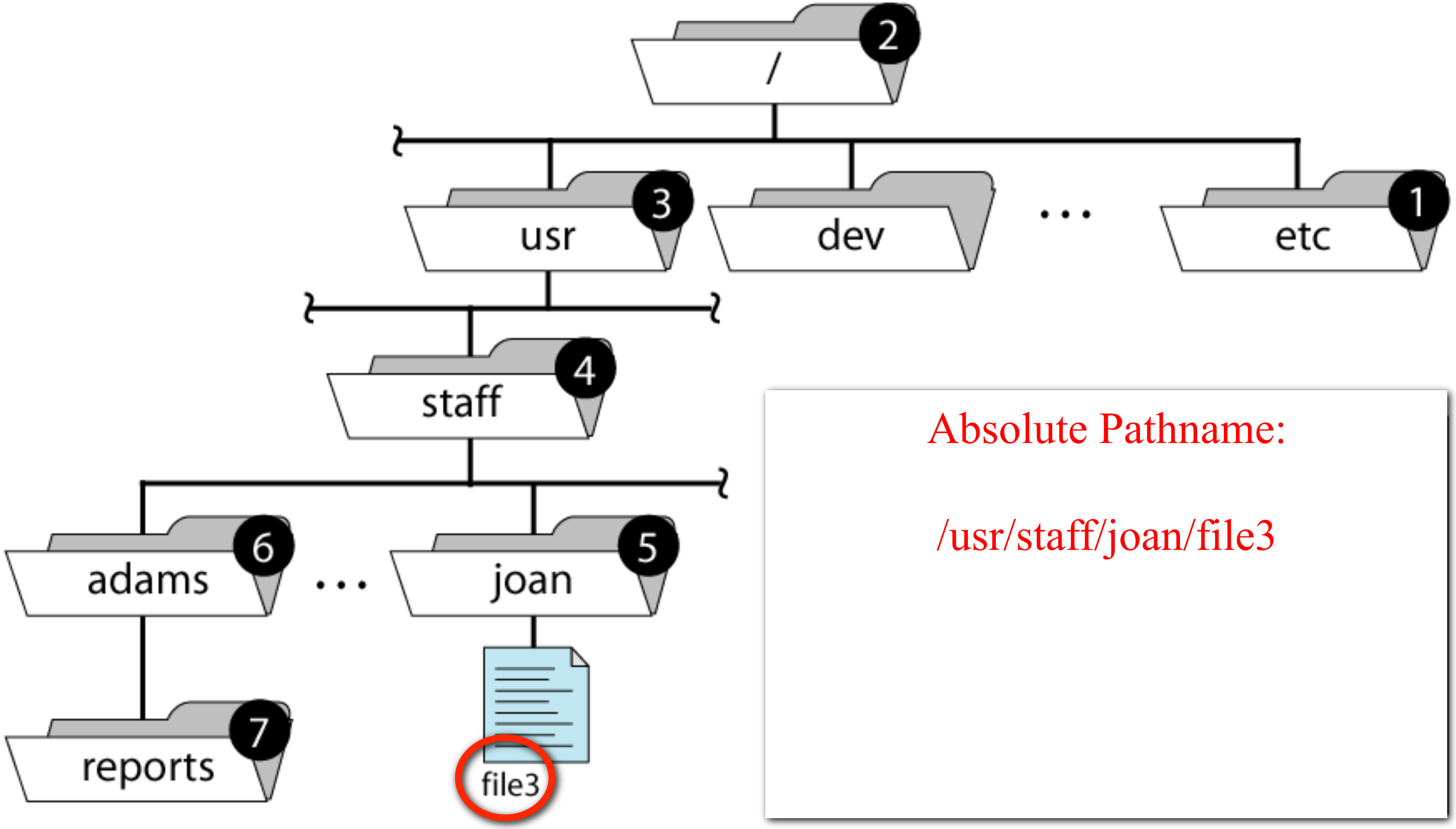
Path

Two ways of locating a file or a directory:

- By Using Absolute Path
  - Full path
  - Traces a path from root to a file or a directory
  - Always begins with the root (/) directory!
  - Example: /home/ux/krush/unix/assignments/assign1.sp04
- By Using Relative Path
  - Traces a path from the ‘cwd’ to a file or a directory
  - No initial forward slash (/)
  - Two dots (..) goes up one level on file structure
  - Dot (.) points to current working directory (cwd)
  - Example: unix/assignments/assign1.sp04
  - ../usr/stafff/joan/file1

Figure 3-4

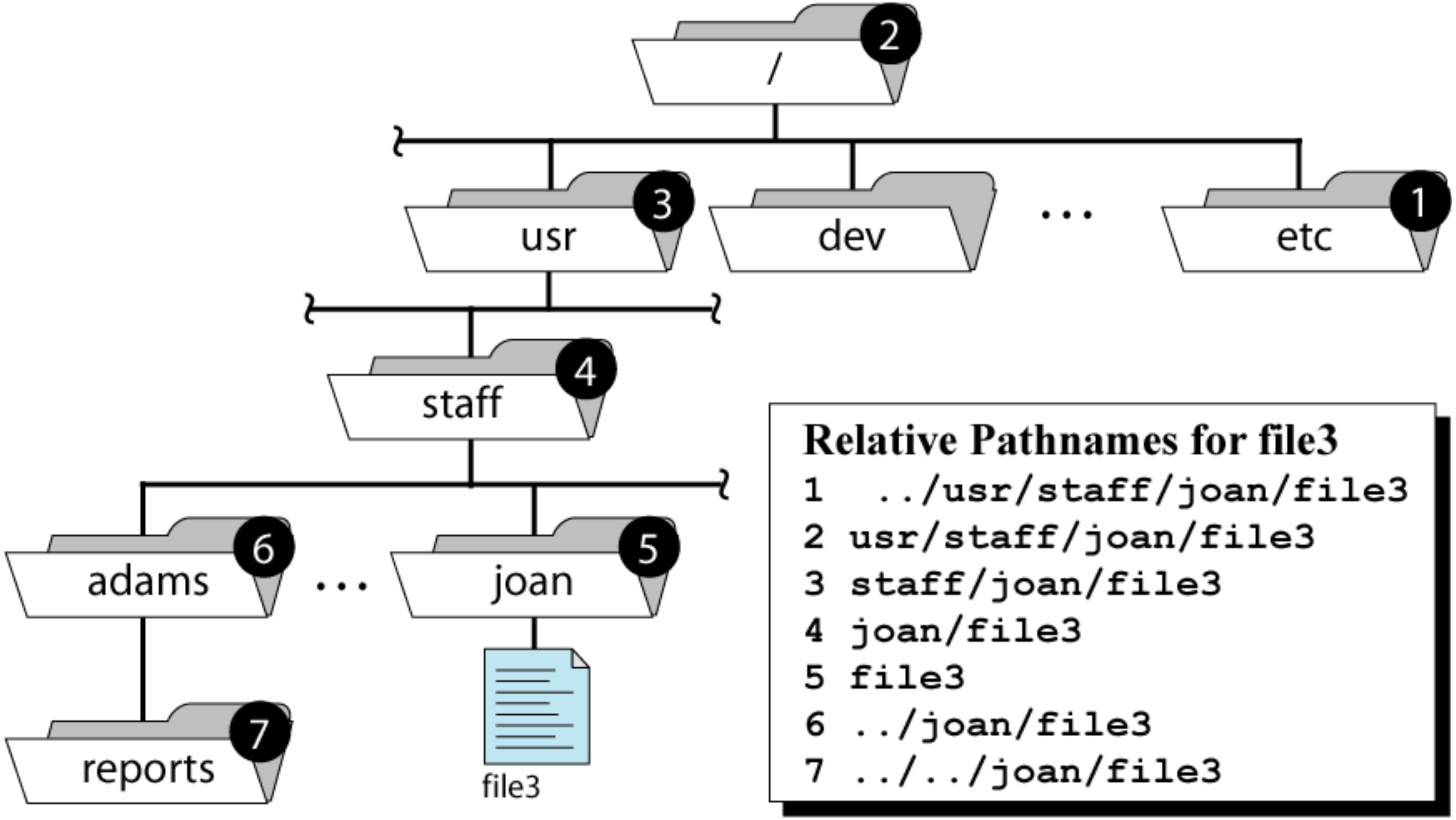
Absolute Path for file3



Absolute Pathname:

/usr/staff/joan/file3

Relative Paths for file3

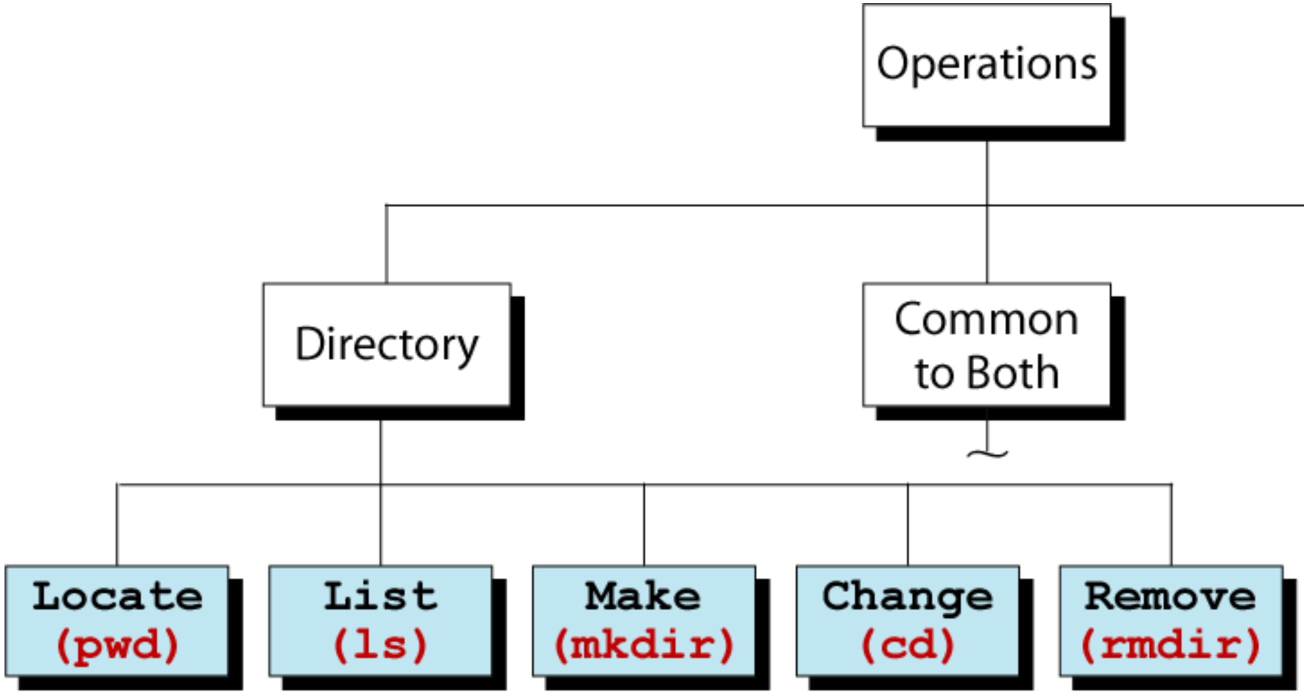


Relative Pathnames for file3

```
1   ../usr/staff/joan/file3
2  usr/staff/joan/file3
3  staff/joan/file3
4  joan/file3
5  file3
6  ../joan/file3
7  ../../joan/file3
```

Figure 3-12

Directory Operations



Display Current Directory’s Full Pathname

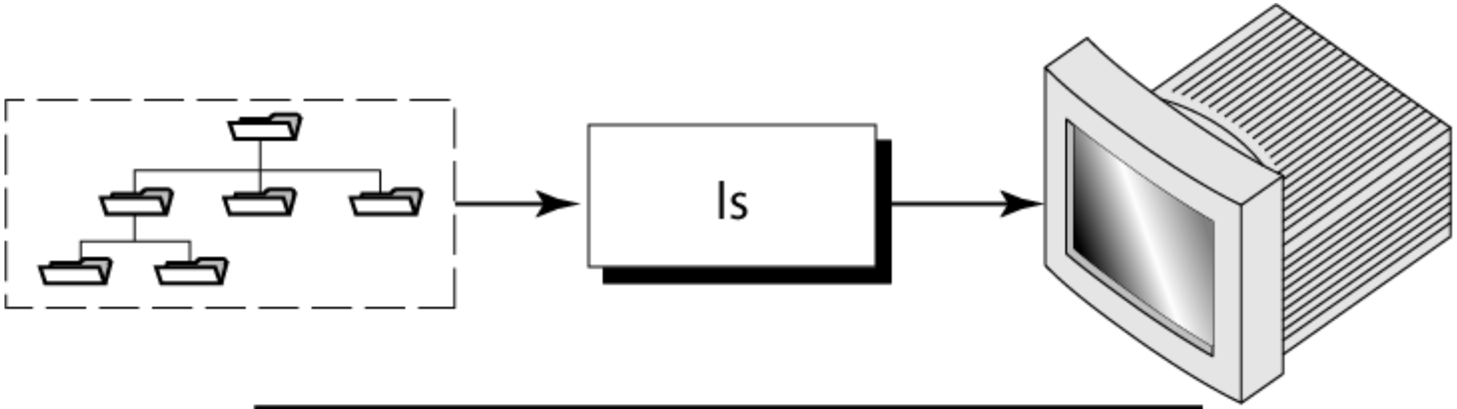
- To determine the full pathname of the current working directory, use the command named “pwd”
- pwd stands for print working directory

Example: To display the full pathname of the current working directory

```
ux% pwd
/home/ux/krush/unix
```

Figure 3-14

The ls Command



ls options pathname

-l:	long list
-d:	working directory
-n:	user/group id's
-r:	reverse order
-t:	time sequence
-u:	time—last access
-c:	time—inode date
-p:	identify directories
-R:	recursive
-1:	print one column
-i:	print inode

ls

- ls

List the content of the current directory

- ls path_name

List the content of the directory in path_name.

- ls –l

Long list

- ls –a

List all hidden files

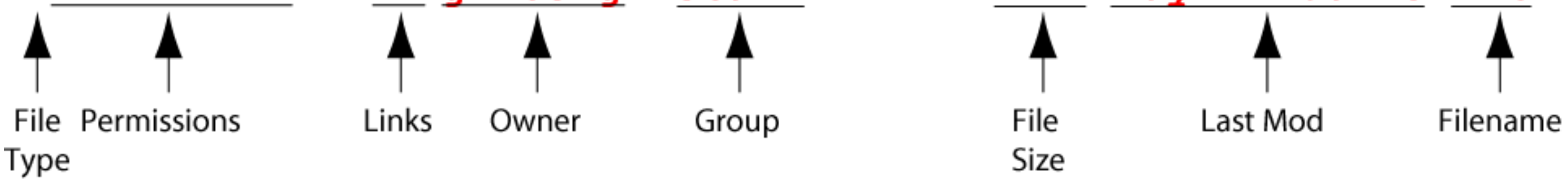
- ls -la

Combine two options –l and –a together

Figure 3-15

Long List Option

```
$ ls -l
total 2
-rw-r--r--    1 gilberg  staff          12 May 17 08:45 f-t
-rw-r--r--    1 gilberg  staff          12 May 17 08:45 f1t
```



File Type Permissions Links Owner Group File Size Last Mod Filename

List Contents of a Specific Directory

Listing contents of a subdirectory named "unix/grades"

```
ux% ls -l unix/grades
total 10
-rwxr-xr-x   3 krush    csci         72 Jan 19 19:12 330assign-graderun
-rwxr-xr-x   1 krush    csci         70 Jan 19 19:13 330exam-graderun
-rwxr-xr-x   2 krush    csci         70 Jan 19 19:12 330quiz-graderun
-r-x------   1 krush    csci        468 Feb  1 11:55 test-330grade
-r-x------   1 krush    csci        664 Feb  1 11:55 test-330grade,v
```

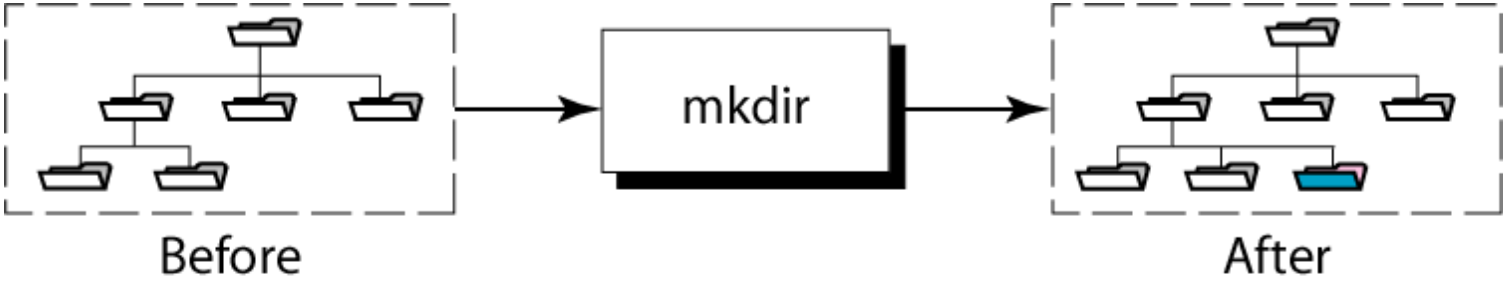
File Name Expansion & Wildcards

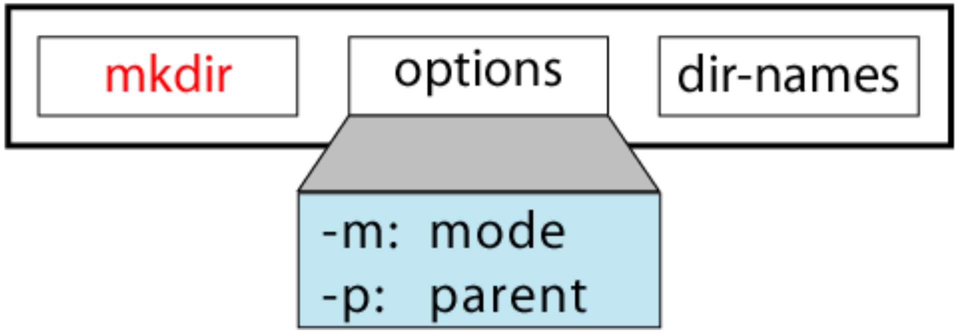
Allows you to select files that satisfy a particular name pattern (wildcards)

Character	Description	Example
*****	**Match zero or more char.**	**ls *.c**
?	**Match any single character**	**ls conf.?**
[*list*]	**Match any single character in *list***	**ls conf.[co]**
[lower-upper]	**Match any character in range**	**ls lib-id[3-7].o**
***str*{*str1*,*str2*,…}**	**Expand *str* with contents of { }**	**ls c*.{700,300}**

Figure 3-17

The mkdir Command




mkdir options dir-names

-m: mode
-p: parent

Directory Names

- Use the following characters:
  - Uppercase letters (A-Z)
  - Lowercase letters (a-z)
  - Numbers (0-9)
  - Underscore (_)
  - Period/dot (.)

Directory Names

- When naming a directory, avoid the following characters:

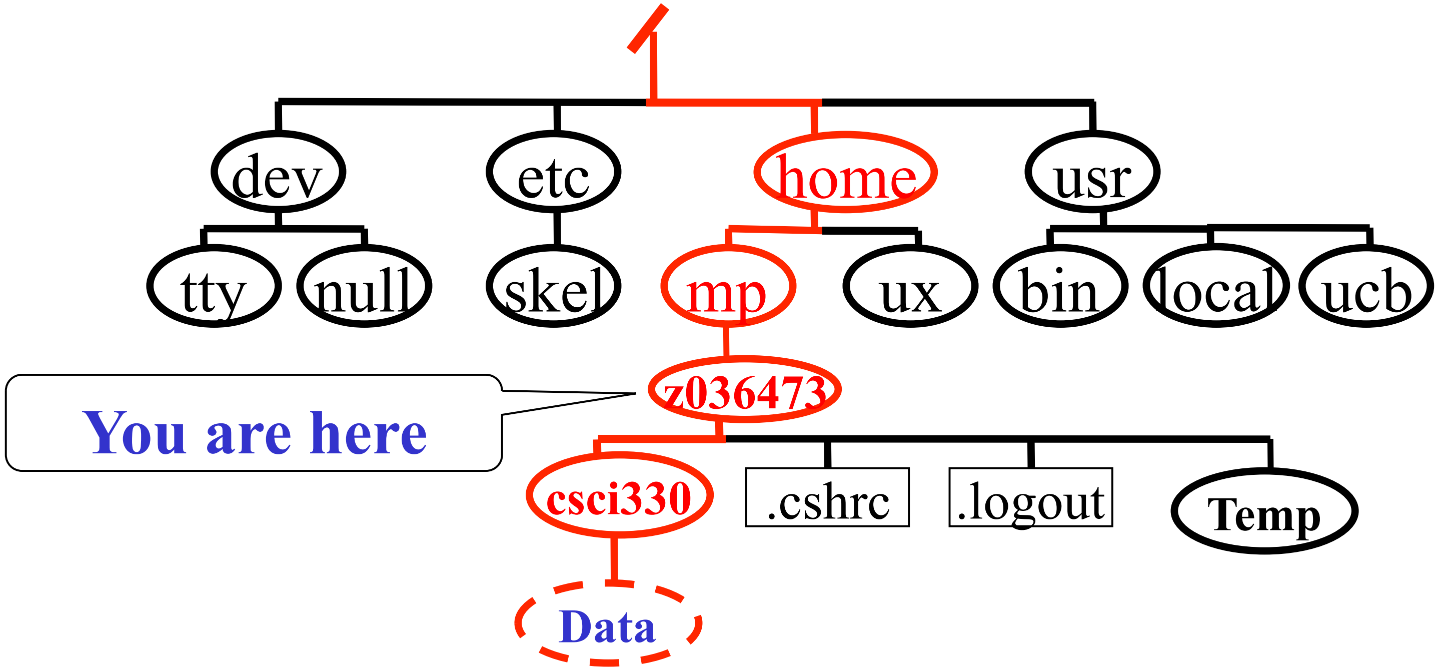
& * \ | [ ] {}

$ <> () # ? /

“ ‘ ; ^ ! ~

Space Tab

Example: Create a Directory Creation



Create a directory called Data under csci330

a) Using Absolute Pathname:

```
mkdir /home/mp/z036473/csci330/Data
```

b) Using Relative Pathname:

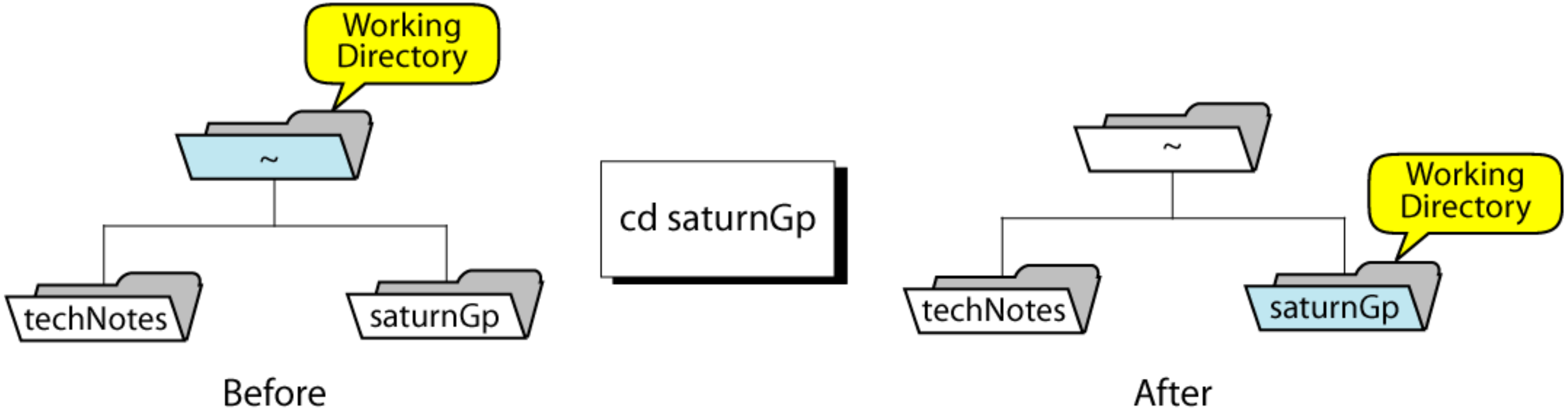
```
mkdir csci330/Data
```

c) Make also missing parent directory, directory Data does not exist

```
mkdir -p csci330/Data/subData
```

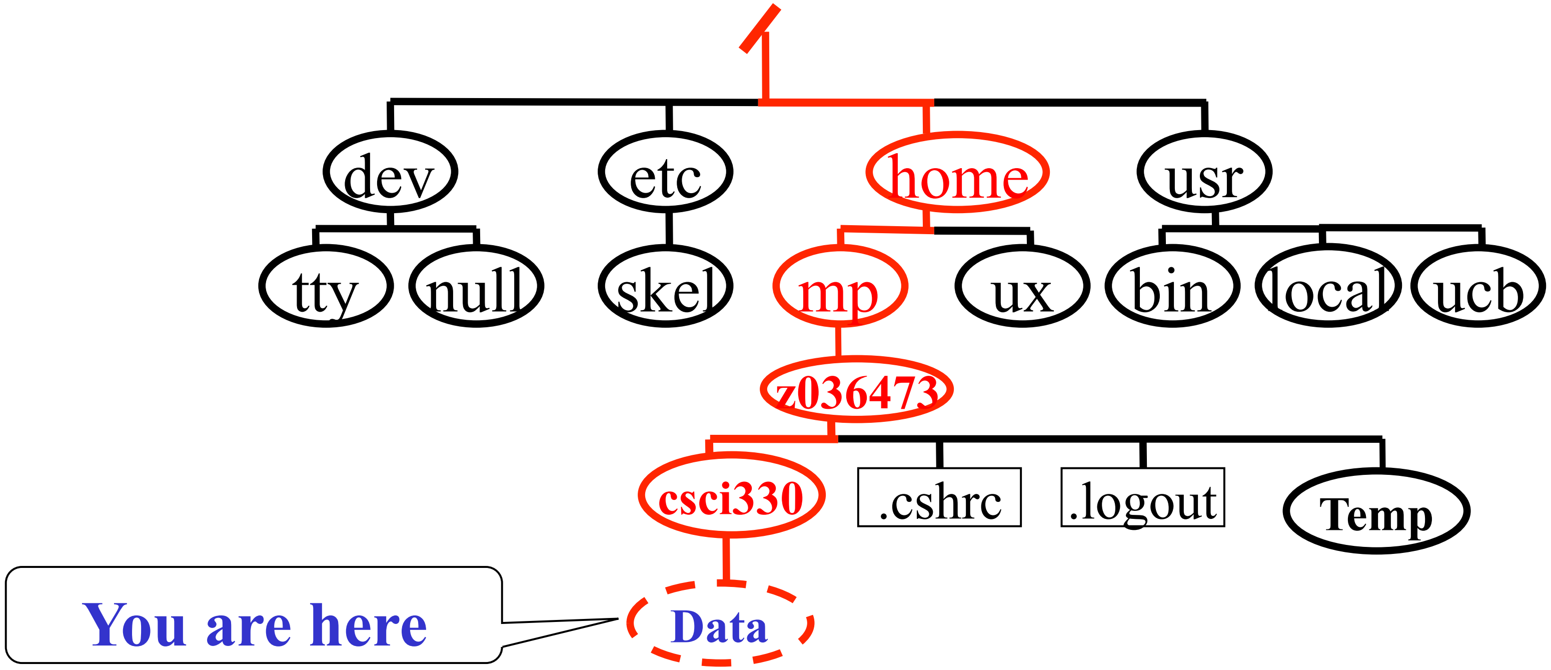
Figure 3-18

The cd Command



cd -options path name

Changing Directory



In the Data directory, go to $HOME directory

a) Using Absolute Pathname:

cd /home/mp/z036473

b) Using Relative Pathname:

cd $home   cd ../..   cd   cd ~   cd ~z036473

Remove Directories

- To remove an empty directory – a directory that does not contain user-created files, use the command named “rmdir”

<u>Example</u>: To remove a directory called “test”, which does not contain user-created files.

ux% rmdir test

- To remove a non-empty directory, use the command named “rm –r”

<u>Example</u>: To remove a non-empty directory called “old-data”

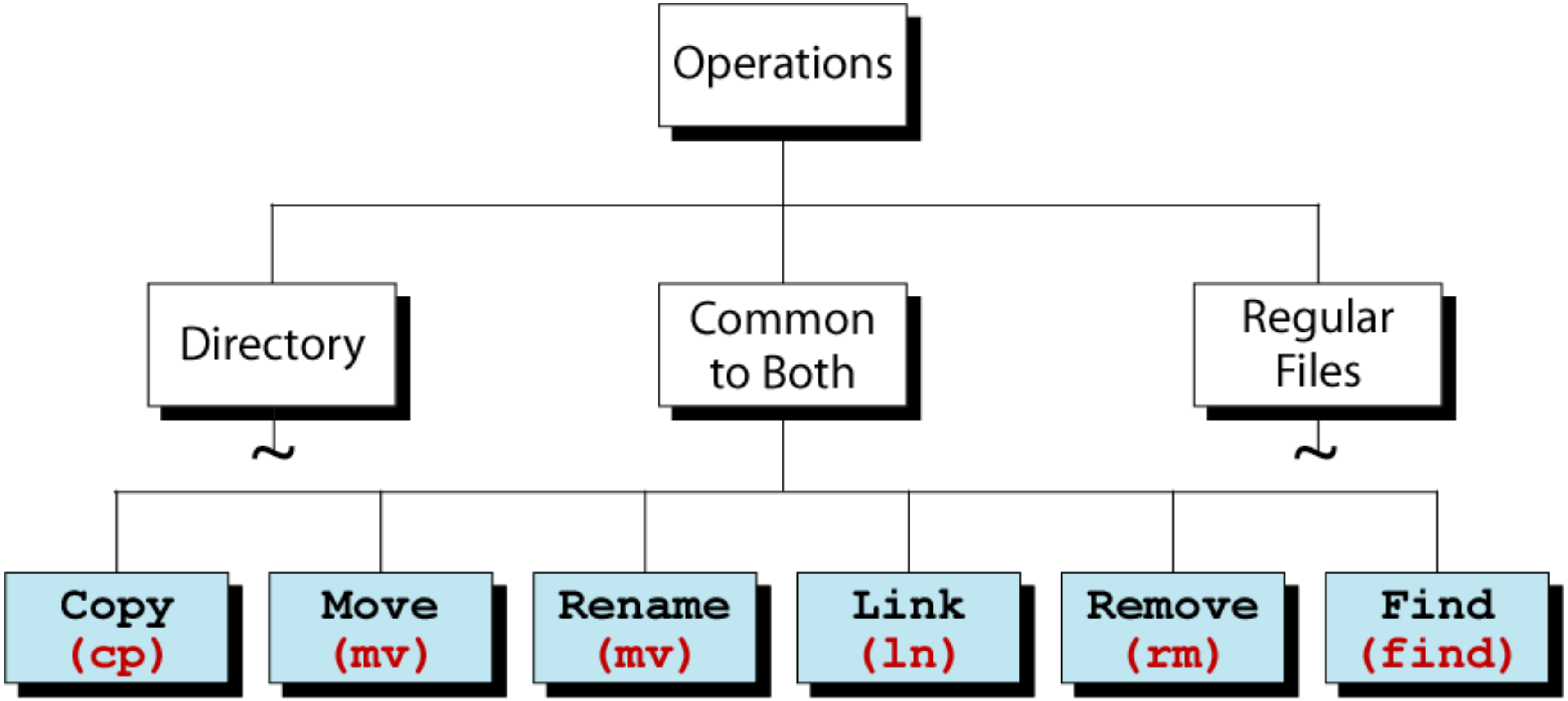
ux% rm –r old-data

Exercises

- In your home directory (usually /home/students/2017xxxx) create directory folder1 then folder2 inside folder1
- List the contents of folder1 then of folder2
- Move to folder2
- Print the current working directory
- Get back to your home directory
- Print the current working directory
- Remove folder1 and folder2
- list the content of your home directory again.

Figure 3-22

Operations Common to Directories and Regular Files



Copying Files

- To copy a file, use the command named "cp"
- Syntax: cp source-file new-file
- Commonly used options:

-i if "new-file" exists, the command cp prompts for confirmation before overwriting

-p preserve permissions and modification times

-r recursively copy files and subdirectories

Copying Files

- "source-file" must have read permission.
- The directory that contains "source-file" must have execute permission.
- The directory that contains "new-file" must have write and execute permissions.
- Note that if "new-file" exists, you do not need the write permission to the directory that contains it, but you must have the write permission to "new-file".

Moving Files

- To move files from one directory to another directory, or to re-name a file, use the command named "mv".
- The directory that contains the source file and the destination directory must have write and execute access permissions.

Moving Files

- Syntax: mv source-file destination-file
- If the destination file exists, “mv” will not overwrite exiting file.

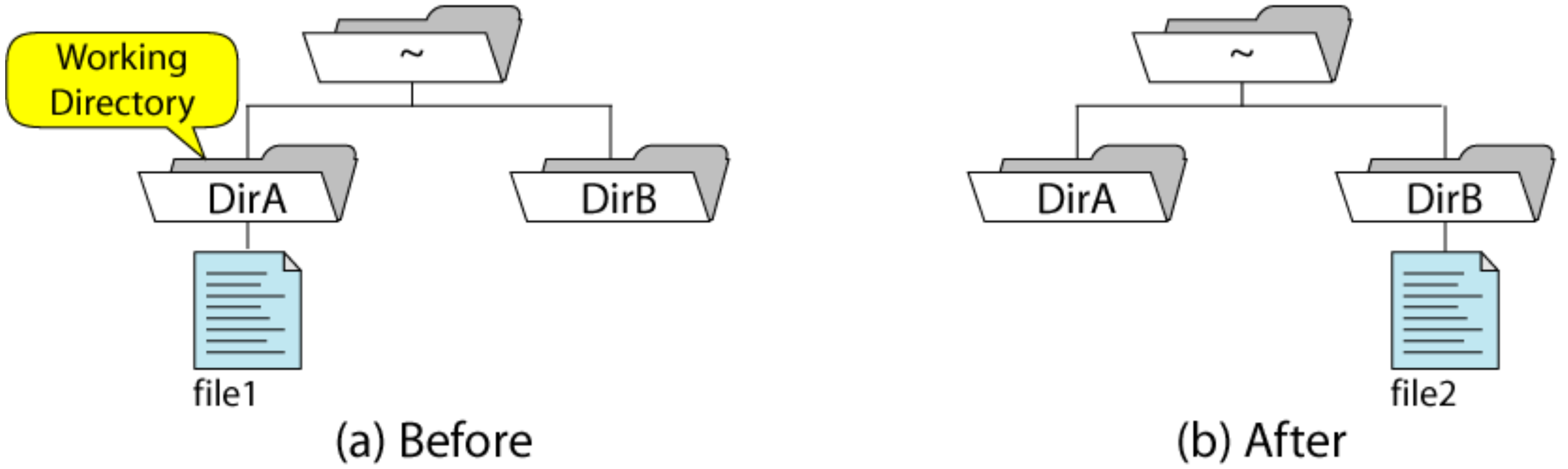
<u>Example:</u> Move “assign1.txt” a different directory and rename it to “assign1.save”

```
ux% mv assign1.txt ~/archive/assign1.save
ux% mv assign1.txt ~/archive
```

Figure 3-31

Moving a File

```
$ mv file1 ~/DirB/file2
```



Rename Directories

- To change the name of an existing directory, use the command named “mv”

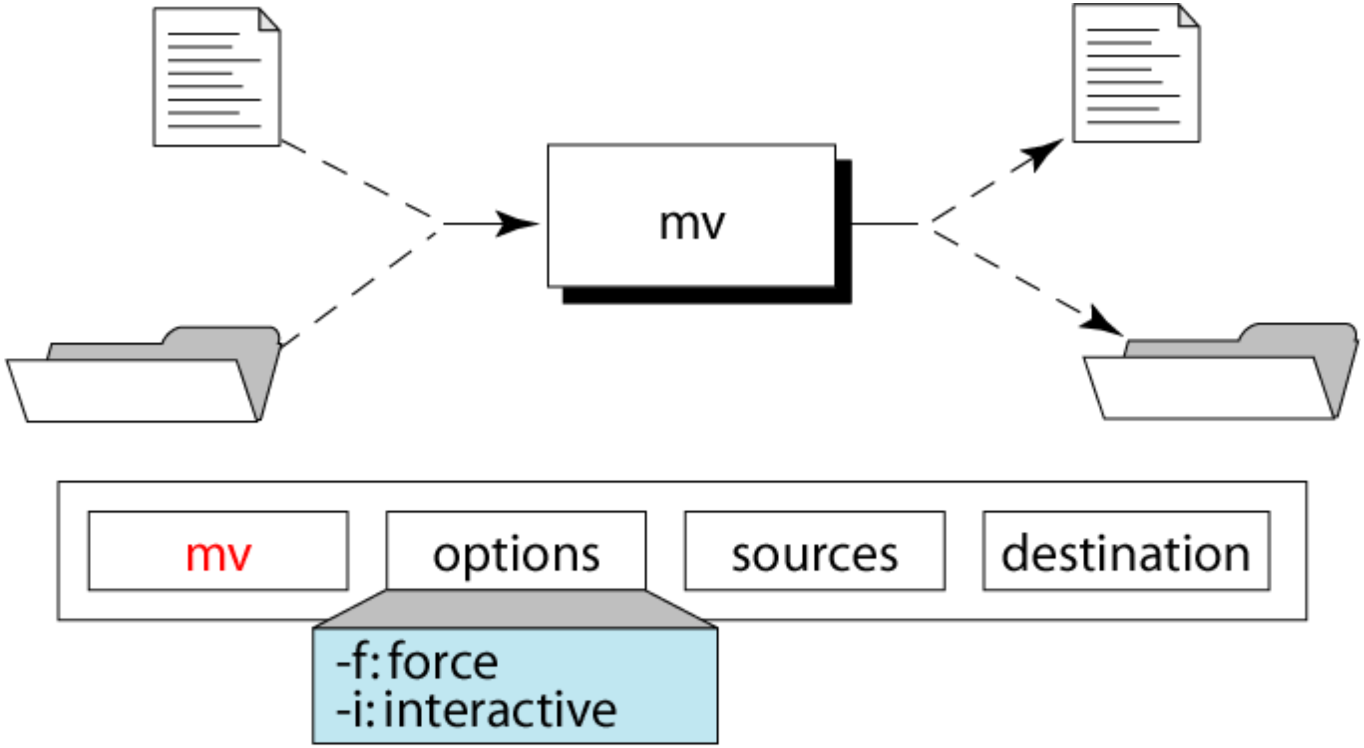
<u>Example:</u> To rename the file called “unix” to “csci330”

```
ux% mv unix csci330
```

- For the above example, what happens if “csci330” already exists in the current directory and it is the name of a directory?

Figure 3-30

The mv Command



Removing/Deleting Files

- You should remove un-needed files to free up disk space.
- To remove/delete files, use the command named "rm".
- Syntax: rm file-list
- Commonly used options:

-f force remove regardless of permissions for "file-list"

-i   prompt for confirmation before removing

-r removes everything under the indicated directory

Removing/Deleting Files

- If “file-list” contains pathname, the directory components of the pathname must have execute permission.
- The last directory that contains the file to be deleted must have execute and write permissions.

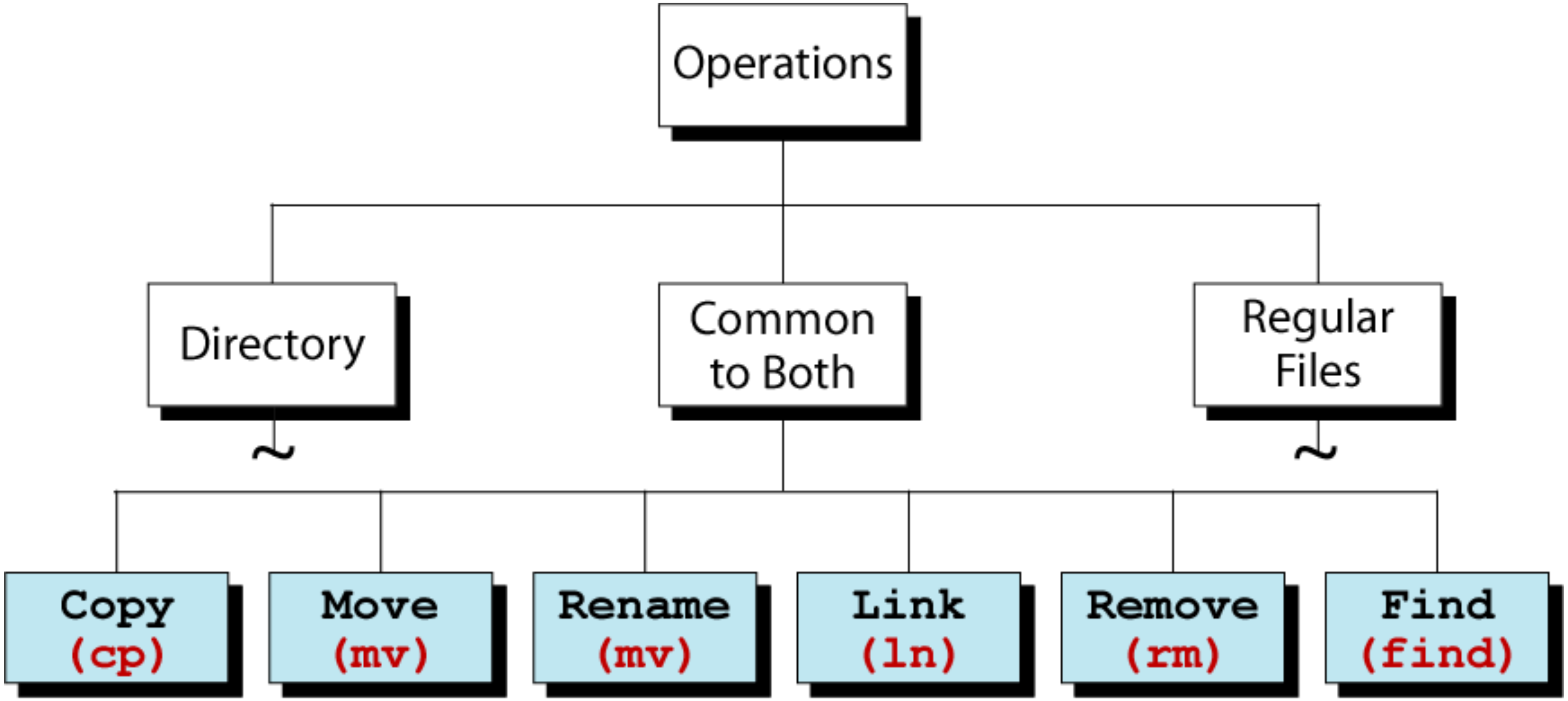
Example: Remove the file named “old-assign”

```
ux% rm unix/assign/old-assign
```

Examples

```
$ ls -l
-rw-r--r-- 1 tuananh  user1   16 Feb 10 19:12 test.txt
drwxr-xr-- 2 tuananh  user1  512 Feb 10 19:14 mydir
$ cp test.txt mydir
$ ls -l mydir
-rw-r--r-- 1 tuananh  user1   16 Feb 12 20:03 test.txt
$ rm -R mydir
$ ls -l
-rw-r--r-- 1 tuananh  user1   16 Feb 10 19:12 test.txt
$ rm test.txt
$ ls -l
$
```

Recap: Common Operations on Files



Finding Files

- The command named “find” can be used to locate a file or a directory.
- Syntax: find pathname-list expression
- “find” recursively descends through pathname-list and applies *expression* to every file.
- Expression
  - -name file_name
  - -perm permission_mod
  - -type d/f/...
  - -size N: N is the minimum number of block (512B)
  - -atime N, -mtime N, -ctime N, where is by defaut the number of day.

Finding Files

Example 1: Find all files, in your directory hierarchy, that have a name ending with “.bak”.

```
ux% find ~ -name *.bak
```

Example 2: Find all files, in your directory hierarchy, that were modified yesterday.

```
ux% find ~ –mtime –1
```

Example

- $find /usr -name toto

- $find /usr -name " *.c »

- $find / -mtime 3

- $find / -size 2000
  - All files with size more than 1 MB (= 2000 block 512 B)
  -

- $find / -type f -user olivier -perm 755

Exercise

- Write command for:
  - Search for the files with name test in your personal folder
  - Search for the files with name .c in the whole disk
  - Search for the files with name ending with .conf in /etc
  - Search for the files with size greater than 100 MB in disk
  - Search for the files with name .lib and size greater than 5MB in the whole disk

The “ln” command

- Allows file to listed in multiple directories
- 2 types:
  - Hard link
  - Symbolic link
- First: understand Unix file storage and organization

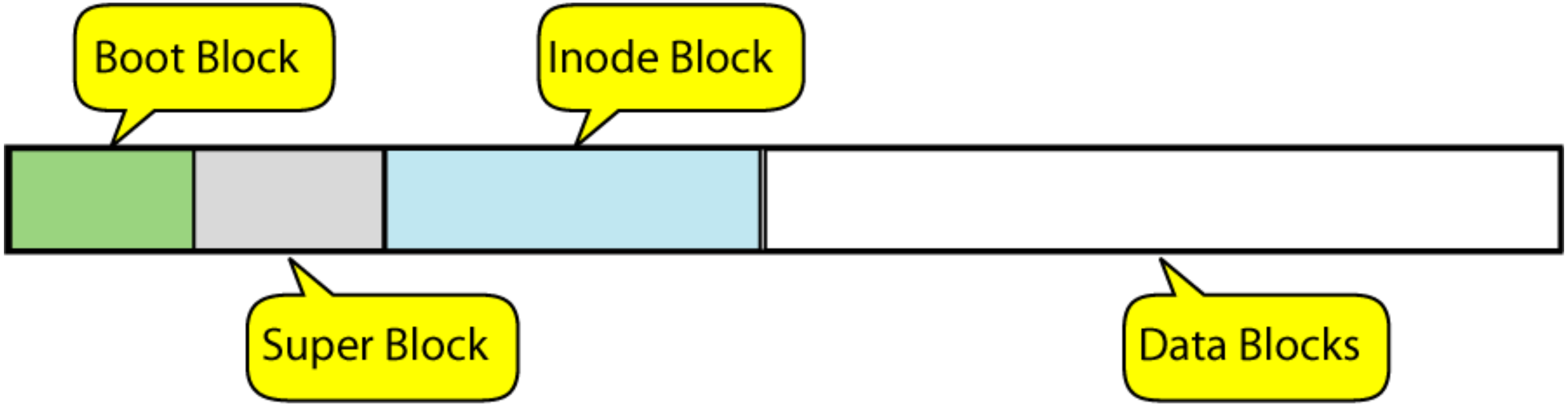
Unix file organization

- Computer has one or more physical hard drives
- Hard drive is divided into partitions

- Partition holds file system
  - File system is set of data blocks
  - Data blocks contain
    - general information
    - actual file data
    - directory information

Figure 3-5

Blocks in a file system

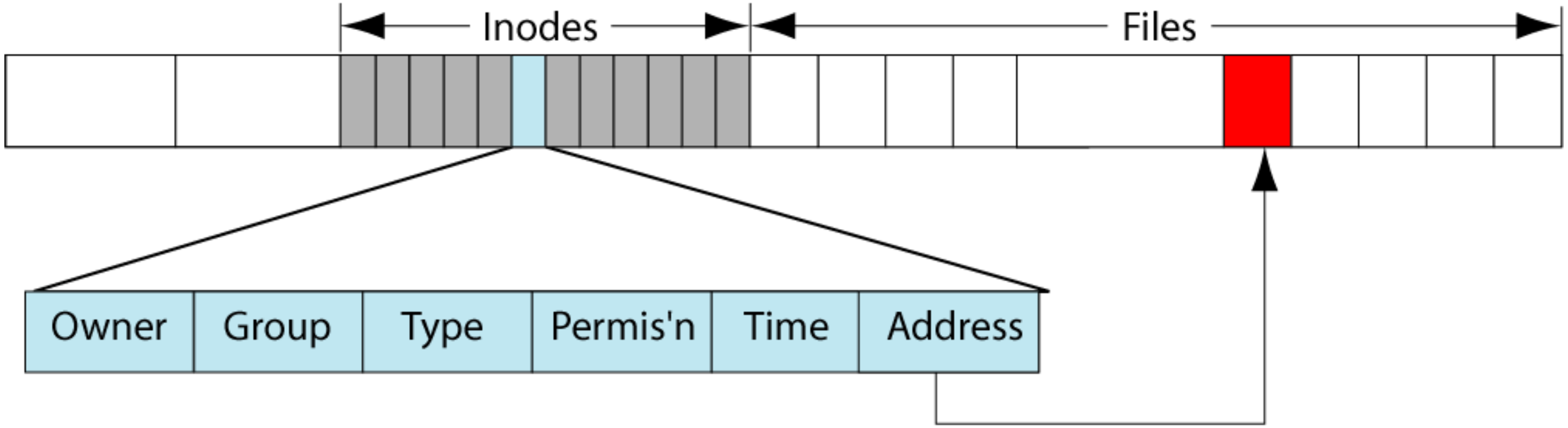


inode

- Index (or information) node: one inode per file
- Each inode has unique number
- contents:
  - File type, access permissions, link count
  - UID, GID
  - Date and time of the file’s last
    - **Data access (read and execute)**
    - **Data modification (written)**
    - **I-node modification (permission change)**
  - Data blocks assigned to the file

Figure 3-6

Inodes in a filesystem

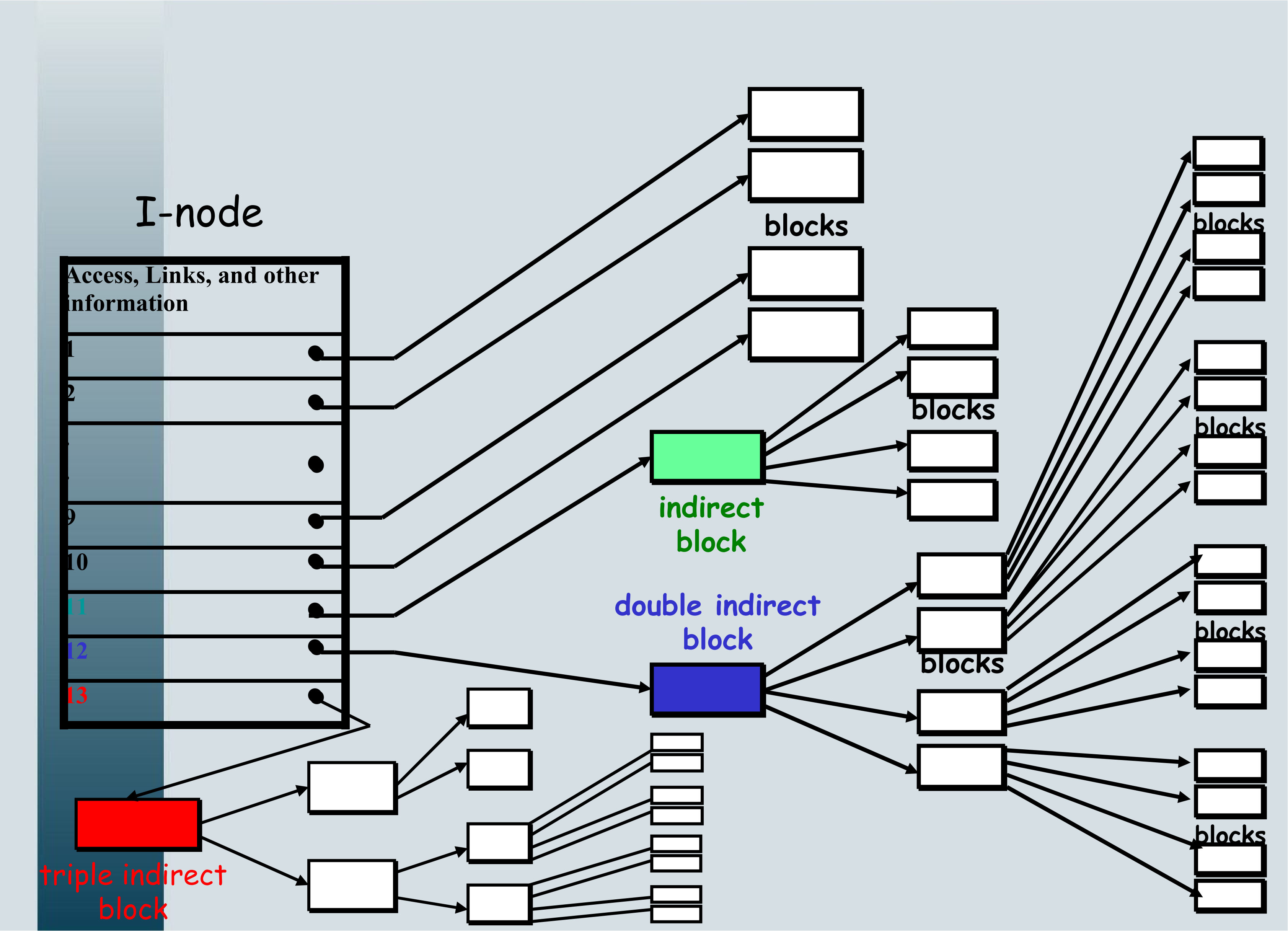


inode Contents: where is the file data ?

Inode may store:

- 10 addresses of data blocks that belong to file
- 1 address of a block that contains data block addresses
- 1 address of a block that contains addresses of blocks that contain data block addresses
- 1 address of a block that contains addresses of blocks that contain addresses of blocks that contain data block addresses

Inode structure



I-node

Access, Links, and other information

1

2

9

10

11

12

13

blocks

indirect block

double indirect block

blocks

blocks

blocks

blocks

blocks

blocks

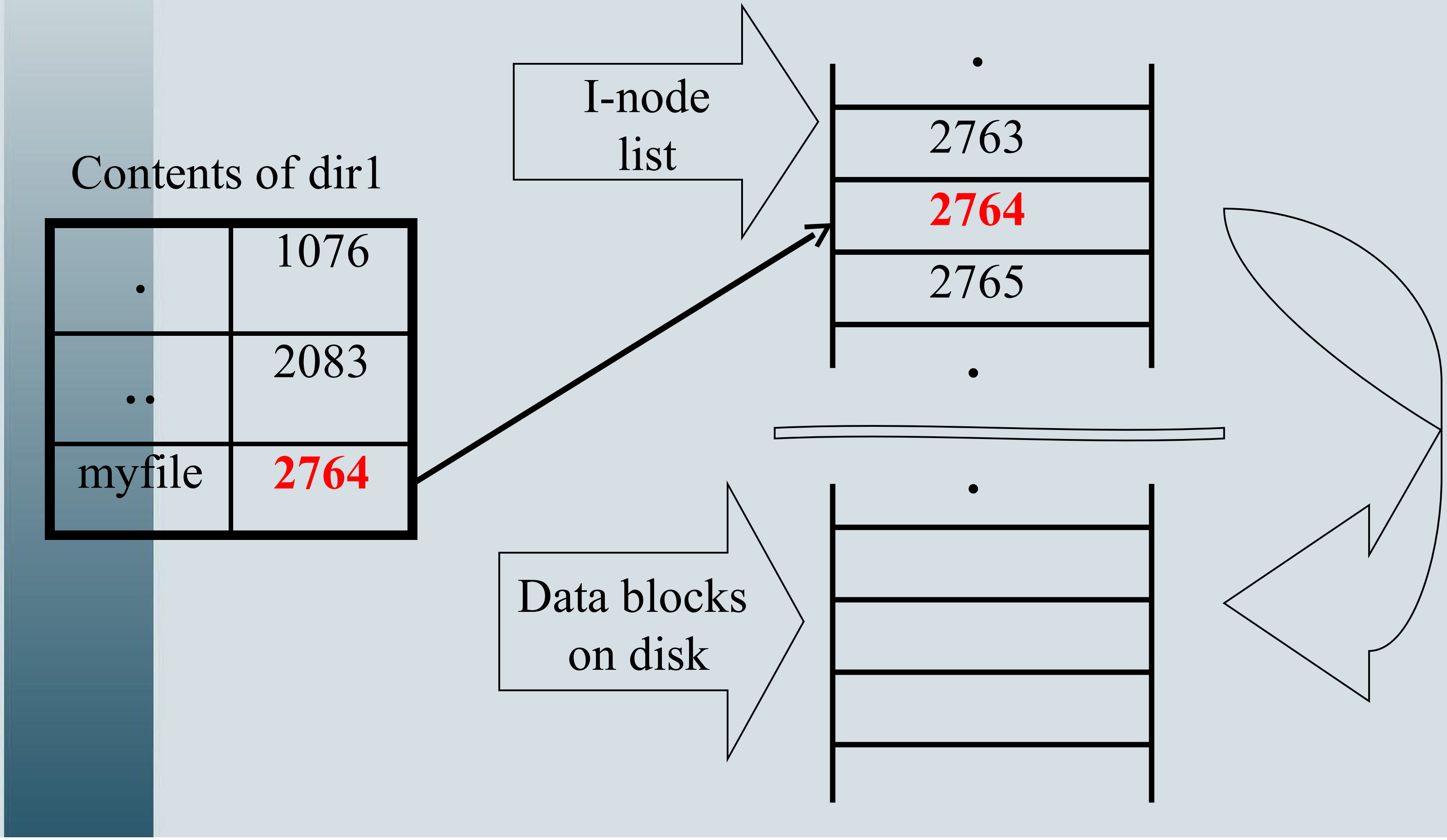
triple indirect block

Directory representation

Directory is a file:

- Has inode like regular file, but different file type
- Data blocks of directory contains simple table:

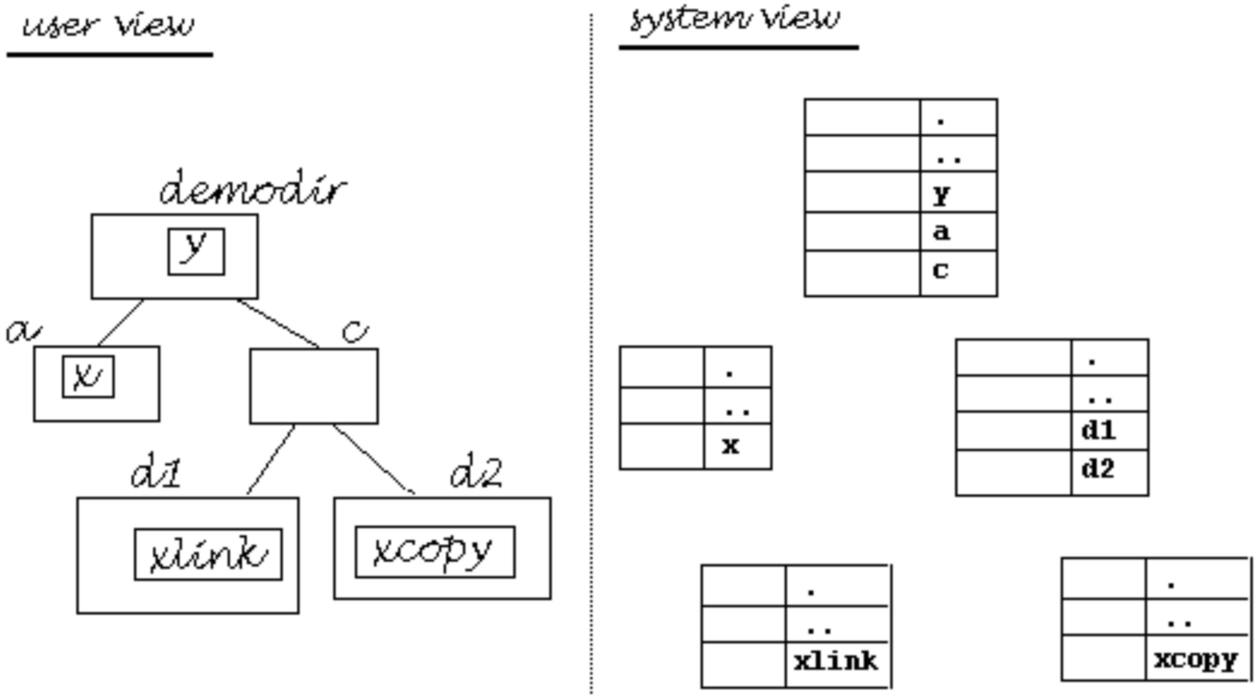
Name	Inode number



Contents of dir1

.	1076
..	2083
myfile	**2764**

Example: user view vs. system view



Output: ls -li

```
ux% ls -li crontab.cron
   118282 -rw-r--r--   1 krush    csci          80 Feb 27 12:23 crontab.cron
```

I-node

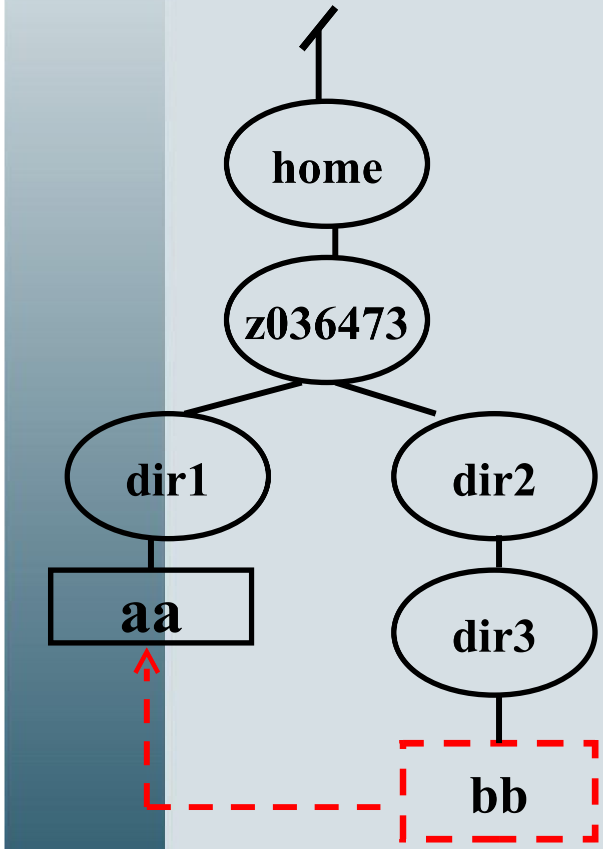
Linking Files

- To share a single file with multiple users, a link can be used.
- A link is:
  - A reference to a file stored elsewhere on the system.
  - A way to establish a connection to a file to be shared.
- Two types:
  - Hard link
  - Symbolic link (a.k.a. “soft link”)

Hard Link

Advantages	Disadvantages
Allow access to original file name via the file name or the I-node number The original file continues to exist as long as at least one directory contains its I-node Checks for the existence of the original file	**Cannot link to a file in a different file system** Prevents owner from truly deleting it, and it counts against his/her disk quota

Hard Link



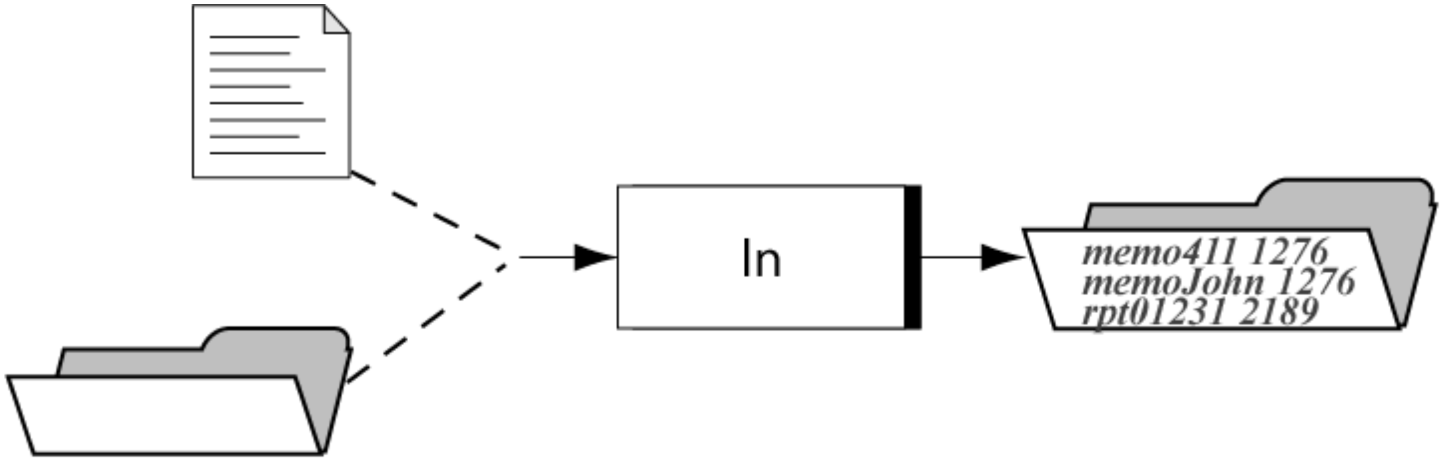
Syntax: ln shared-file link-name

From dir3, link to the file ‘aa’ in dir1 name it ‘bb’:

```
% ln /home/z036473/dir1/aa bb
```

Figure 3-32

The ln Command

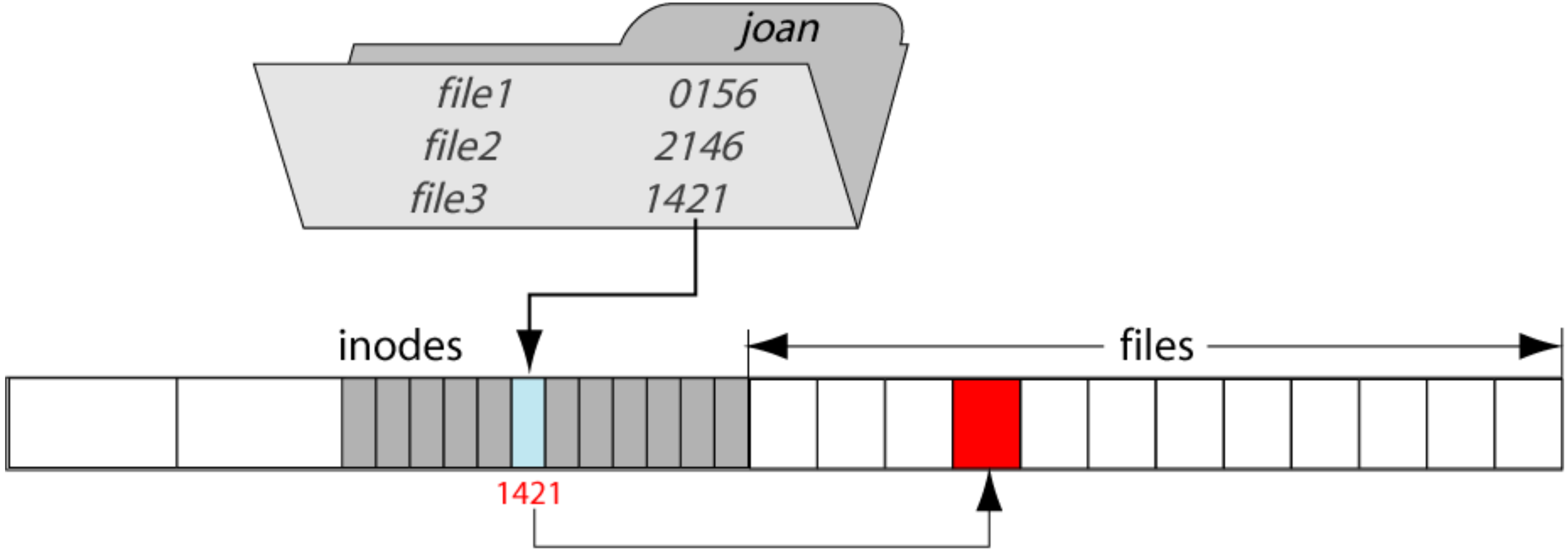


ln options sources destination

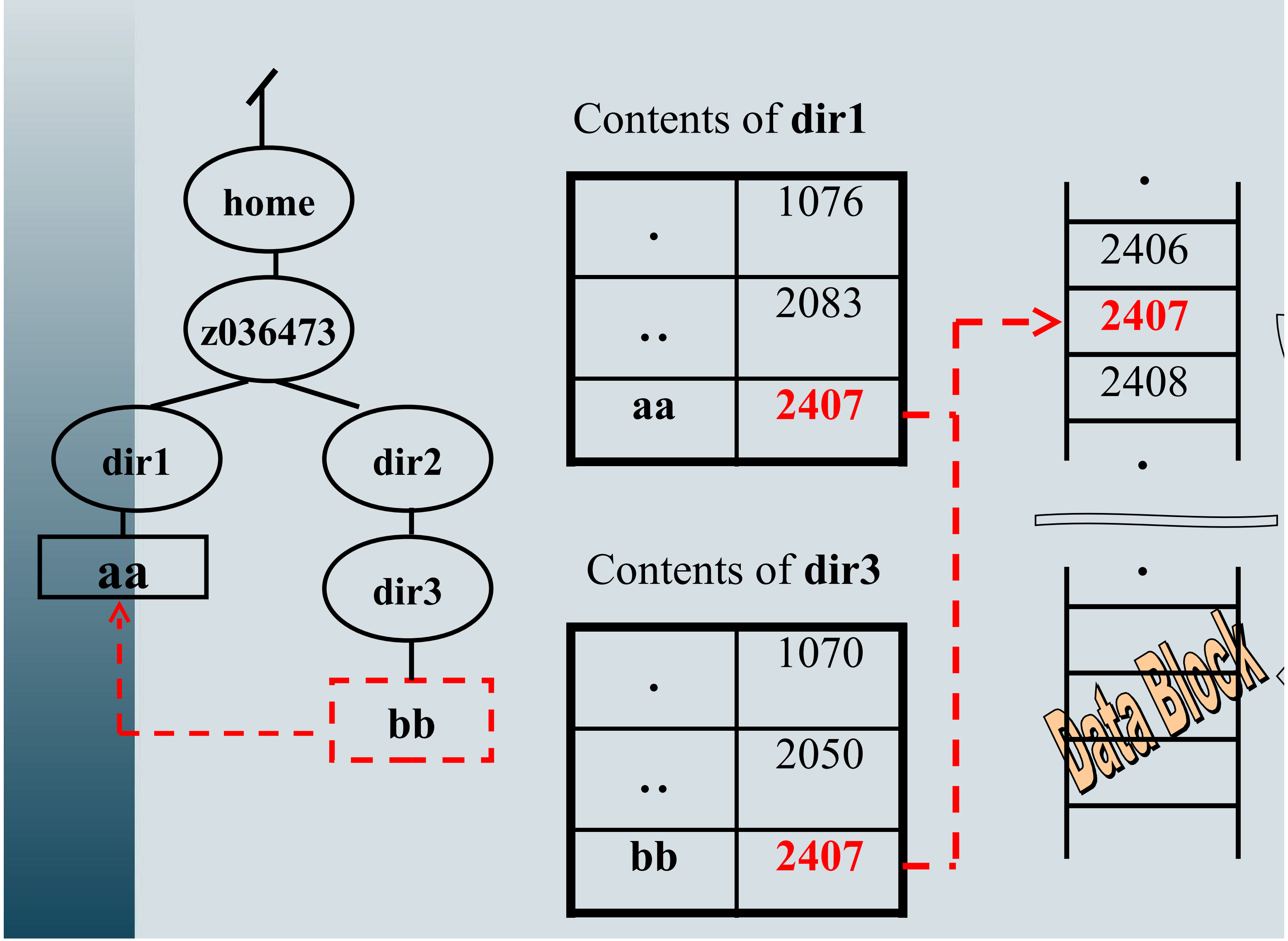
-s: symbolic
-i: interactive
-f: force

Figure 3-8

A Hard Link



A Hard Link



Contents of **dir1**

.	1076
..	2083
aa	2407

Contents of **dir3**

.	1070
..	2050
bb	2407

Exercise

- Use nano or gedit for creating a file in your personal folder, name it file1
- Make a hardlink to file1, name the new file file2
- Show the content of file2 on screen by using
  - $ cat file2
- Modify the content of file2 by using nano
- Check the content of file1 with cat
- Delete file1
- Check the content of file2.

Symbolic Link

Advantages	Disadvantages
Allow access to original file name Can use either relative or absolute path to access the original file **Can cross partition and drives** **Allows the creation of a link to a directory**	Created without checking the existence of the shared file Cannot access the shared file if its path has restricted permissions Can be circular linked to another symbolic linked file

Symbolic Link

- A hard link may not be created for a file on a different file system
- Use symbolic link
- The linked files do not share the same I-node number
- link-path: relative path to the shared-file

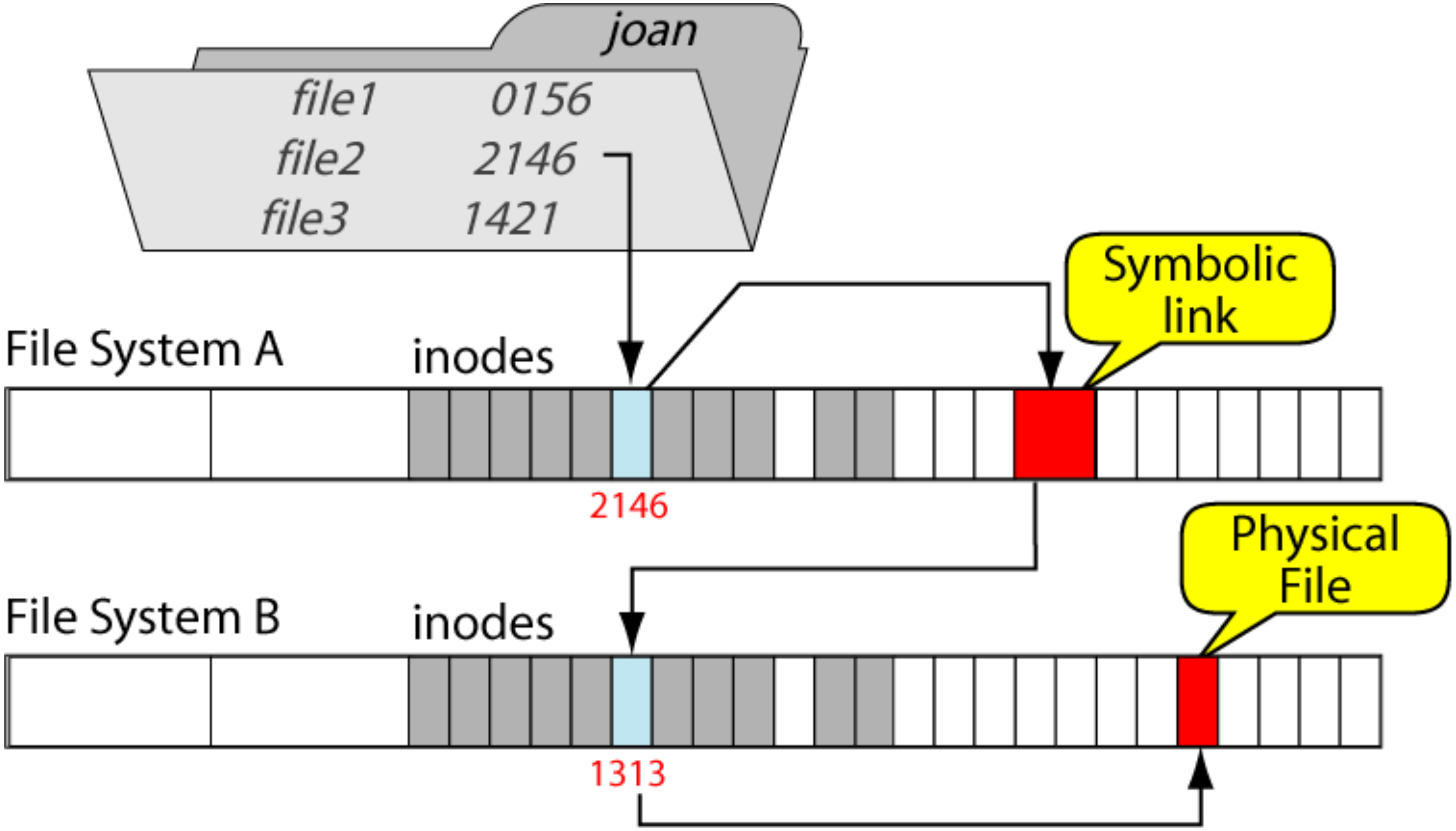
Syntax: ln –s shared-file link-path

Also called source-file

Also called target-file

Figure 3-10

Symbolic Links to Different File Systems



Examples

```
$ echo « Test content » > test.txt
$ ls -l
-rw-r--r-- 1 tuananh user1 8 Feb 10 1:12 test.txt
$ ln test.txt link1
$ ln -s test.txt link2
$ ls -l link*
-rw-r--r-- 2 tuananh user1 16 Feb 10 1:12 link1
lrw-r--r-- 1 tuananh user1 16 Feb 10 1:13 link2->test.txt
```

User’s Disk Quota

- A disk quota is set for each user account
- The command: quota –v

displays the user’s disk usage and limits

- 2 kinds of limits:
  - Soft limit: ex. 3MB
    - Maybe exceeded for one week
    - System will nag
  - Hard limit: ex. 4MB
    - Cannot be exceeded

Exercise

- Use nano or gedit for creating a file in your personal folder, name it file1
- Make a symbolic link to file1, name the new file file3
- Show the content of file3 on screen by using
  - $ cat file2
- Modify the content of file3 by using nano
- Check the content of file1 with cat
- Move file1 to another folder
- Check the content of file3.
- Move file1 back
- Check the content of file3