Files, accounts, and permission in Linux

LECTURER: TRẦN NGUYÊN NGỌC

EMAIL: NGOCTN@SOICT.HUST.EDU.VN; NGOC.TRANNGUYEN@HUST.EDU.VN

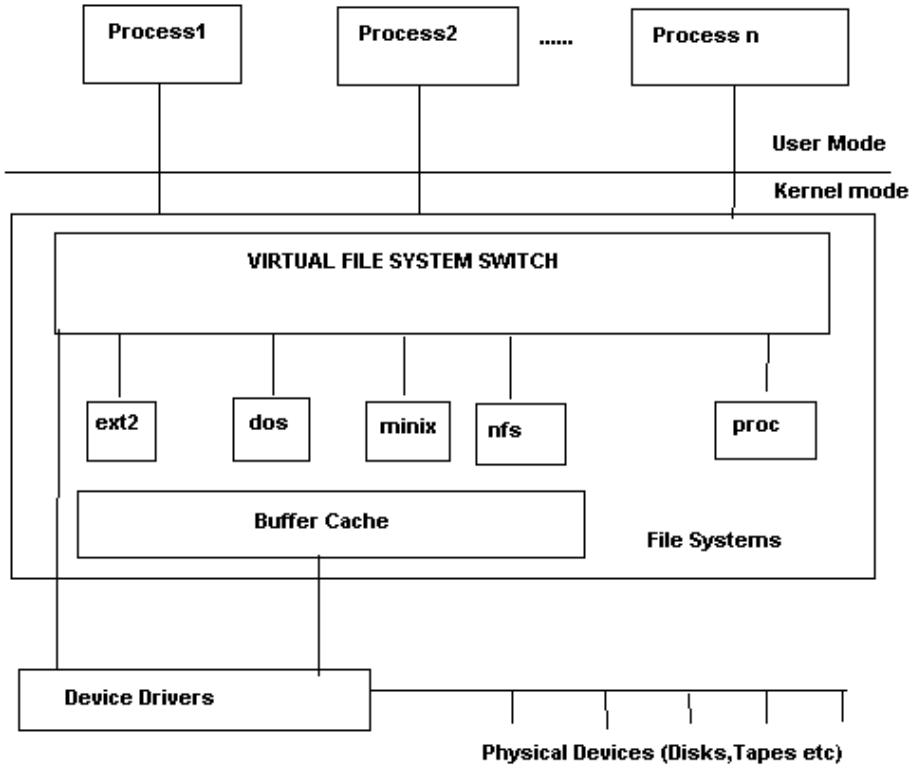
OFFICE: 405-B1 HUST

Contents

- Files in Linux
  - Concept of logic file system
  - Operations with directories
  - Operations with files
  - Inode
- Accounts and permission
  - User and group accounts
  - Manage user and group accounts
  - Access permission
  - File permission
  - Directory permission
  - Manage the access permission

FILES IN LINUX

Logic file system



Structure of file system

One/Many hierrachical tree(s) with directories and files

- File: group of bits
- Directory: group of files and directories

Root directory (/) is the root for the whole hierrachical tree

Files are leaves

Popular Linux directories

/ (root directory)

- /bin : essential user binaries
- /boot : static boot files the OS needs in order to boot
- /etc : contains all configuration files of the system or in its sub-directories
- /dev : where all your devices live: keyboard, mouse, printer, disk, partition.
- /home : contains a home folder for each user
- /lib : contains libraries needed by the essential binaries in the /bin and /sbin
- /usr : contains applications and files used by users, as opposed to applications and files used by the system
- /var : is the writable counterpart to the /usr directory
- /proc: special files that represent system and process information

Linux files vs. Windows files

Similar

- Maximum length of file names is 255
- Accept most characters to name files except some special characters such as * ? [ ] & as they are used for special purposes

Linux files only

- A single hierrachical structure for the file system, unlike Windows
- No definition of extension part of file name (character '.' is treated the same as other characters).
- No logic disks the hierrachical file system
- '/' is used instead of '\'

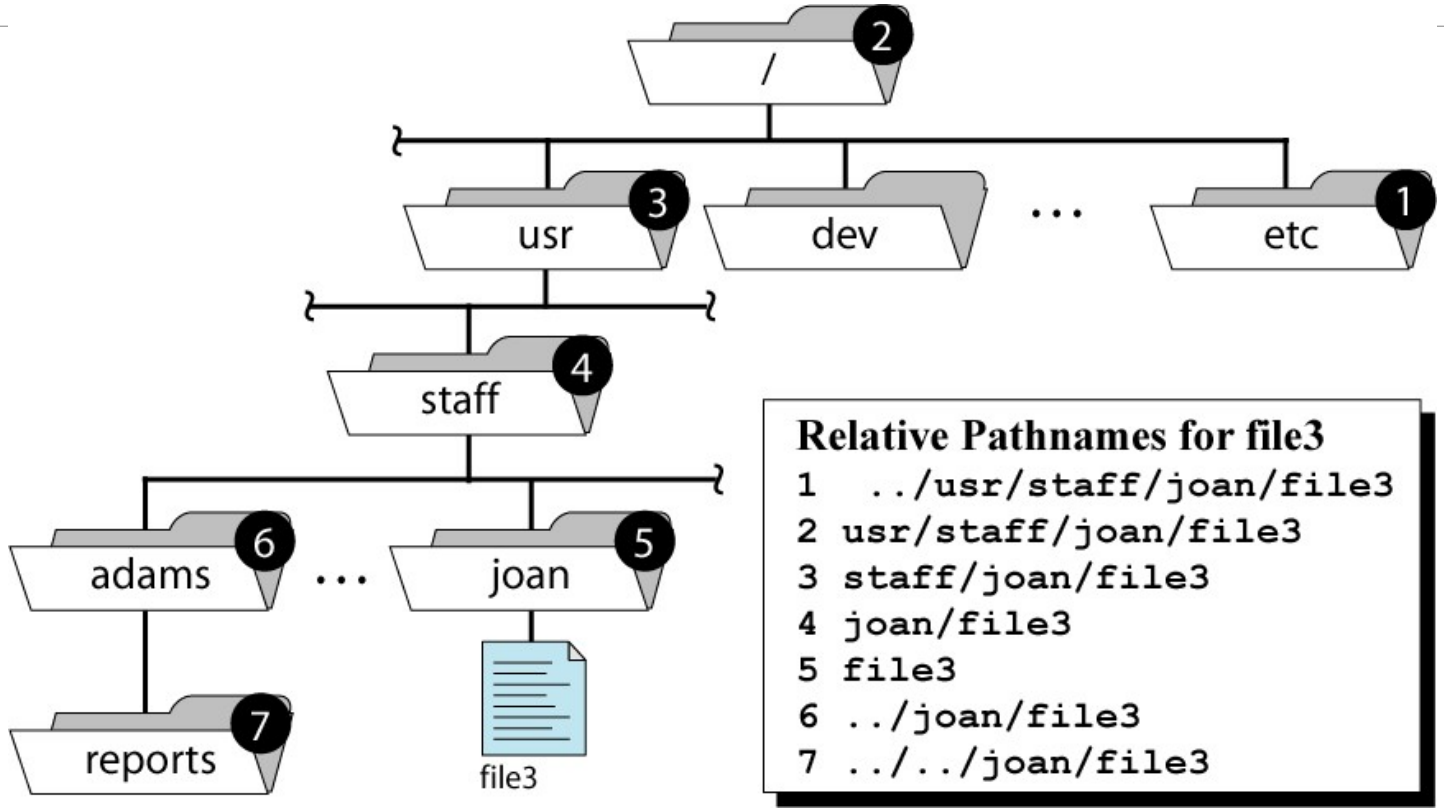
Special paths and directories

To access files and directories, we need to use paths (absolute or relative)

Path can be started from special directories

- / : root directory
- ~/ : home directory
- . : current directory
- .. : parent directory of the current one

Absolute vs relative paths



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
```

Basic commands to manage directories

pwd

cd

ls –la [new dir]

mkdir [-p] [new dir]

rmdir [empty dir]

Manage directory

pwd: absolute path of the current/working directory

cd: change the working directory

- $ cd /home/tuananh ↵
- $ cd tuananh ↵

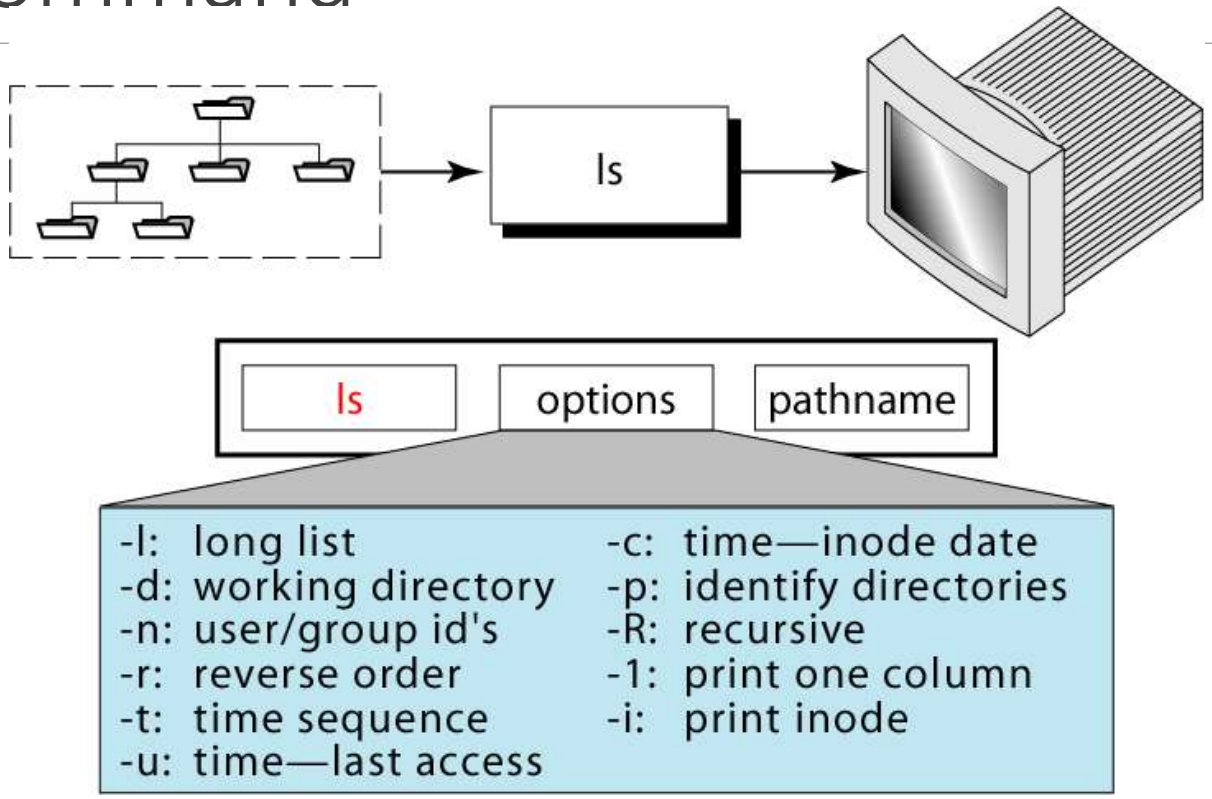
ls: list files inside a directory

- $ ls ↵
- $ ls /home/tuananh
- $ ls –la tuananh
  - Option –a to list hidden files
  - Option -l to list all information, not just names

mkdir: create a blank directory

rmdir: delete an empty directory

Ls command


ls
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

File types

Following characters represent file types of Linux

- : normal file

d : directory

b : block device file (special)

c : character device file (special)

l : symbolic link

m : share memory

p : named pipe

Special names: hidden files start as « . » (Ex: /home/tuananh/.bashrc)

Example

```
$ cd ~
$ pwd
/home/tuananh
$ ls -la
-rw-r--r-- 1 tuananh  user1 2451 Feb  7 07:30 .bashrc
-rw-r--r-- 1 tuananh  user1 4025 Feb 10 19:12 linux.ppt
drwxr-xr-- 2 tuananh  user1  512 Feb 10 19:12 linux
$ mkdir vanban
$ cd vanban
$ pwd
/home/tuananh/vanban
$ cd ..
$ pwd
$ rmdir vanban
```

Wildcard characters

- **Asterisk (*)** matches one or more occurrences of any character, including no character
- **Question mark (?)** represents or matches a single occurrence of any character
- **Bracketed characters [ ]** matches any occurrence of character enclosed in the square brackets
- **Bracketed characters with exclamation mark [!]** matches any occurrence of character not in the square brackets

Examples

```
$ ls -l *.[c,h]
-rw-r--r-- 1 tuananh  user1 2451 Feb  7 07:30 myprog.c
-rw-r--r-- 1 tuananh  user1 2451 Feb  7 07:30 myprog.h
$ ls -l *prog
drwxr-xr-- 2 tuananh  user1  512 Feb 10 19:12 c_prog
drwxr-xr-- 2 tuananh  user1  512 Feb 10 19:12 java_prog
$ ls -l .*
-rw-r--r-- 1 tuananh user1 451 Feb 7 07:30 .bashrc
-rw-r--r-- 1 tuananh user1 225 Feb 7 07:30 .bash_profile
-rw-r--r-- 1 tuananh user1 351 Feb 7 07:30 .bash_logout
```

Manage files

$cp file1 […] dir

- Copy one or more files to a directory

$mv file1 […] dir

- Move one or more files to a directory
- And/or change names

$rm file1 […]

- Remove one or more files

option -R (recursive)

- Allow to copy/move/remove a whole directory including child directories and files

Manage files (cont.)

cat: quick look of a file

more: view each line of a file

tail: view the end of a file

head: view the beginning of a file

touch: create a new file, update an old one

echo content > [file]

Example

```
$ 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 vanban
$ cp test.txt vanban
$ ls -l vanban
-rw-r--r-- 1 tuananh  user1   16 Feb 12 20:03 test.txt
$ rm –R vanban
$ ls -l
-rw-r--r-- 1 tuananh  user1   16 Feb 10 19:12 test.txt
$ rm test.txt
$ ls -l
```

inode

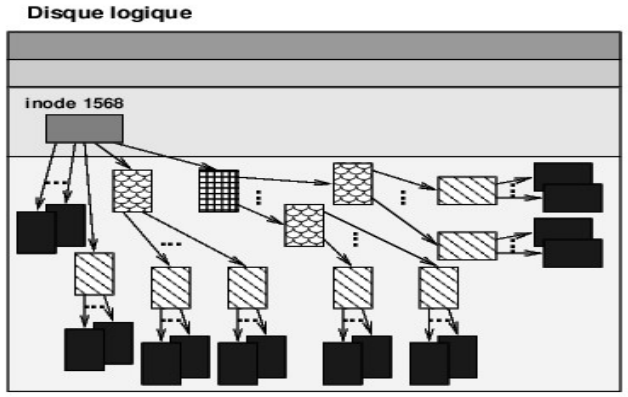
The inode (index node) is a data structure that describes a file-system object such as a file or a directory

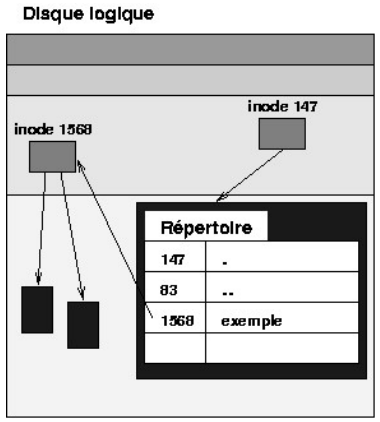
Content of a file is stored in data blocks

- Blank file = inode without data blocks

A directory is a link with the content of a reference table

- A link attachs a file name with an inode





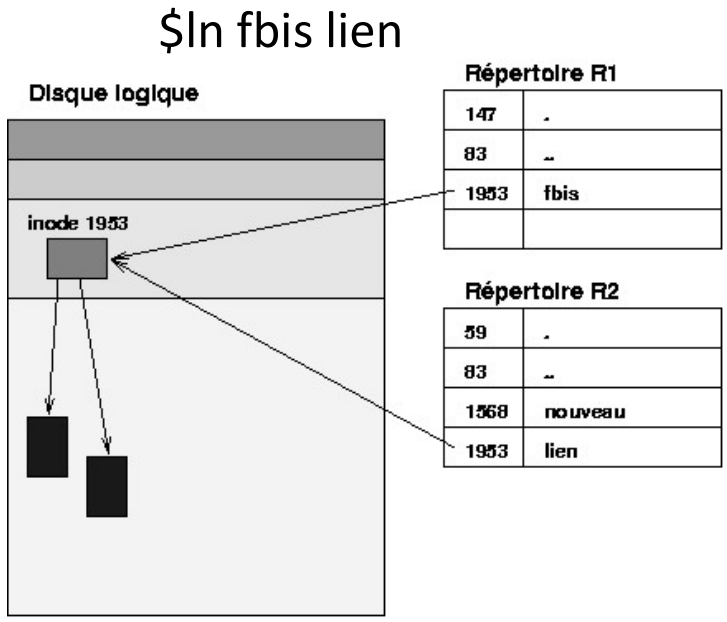
Hard link (1)

A hard link is a direct reference to a file via its inode

There might be multiple hard links to a same inode

Command ln allows to create a new hard link to an existing inode

- The new file share the same inode with the original fiel
- Syntax: ln <old_file> <new_file>



Hard link (2)

The number of hard link(s) to an inode can be show by using ls –l

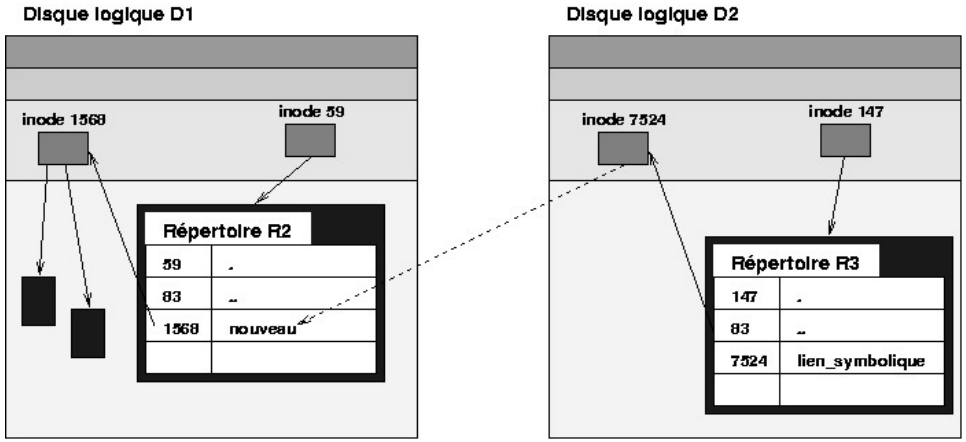
```
$ ls -l
-rw-rw-r-- 1 tuananh user1    0 Nov 12 15:19 file
drwxr-xr-x 2 tuananh user1 4096 Dec 14 17:50 dir
```

Question: Why does a directory alway have at least 2 hard links

Removing a file means deleting a hard link to the inode

- If the deleted file is the last link to the inode, the inode will be removed as well

Symbolic link



ln -s R2/nouveau R3/lien_symbolique

- While creating a symbolic link (option –s), a new inode is created
- Inode contains the path (relative or absolute) of reference object

Hard link vs symbolic link

Symbolic can be used to overcome the limitation of storage

- A hard link need more storage than a symbolic link

How could we distinguish a symbolic file and the original file of a symbolic link?

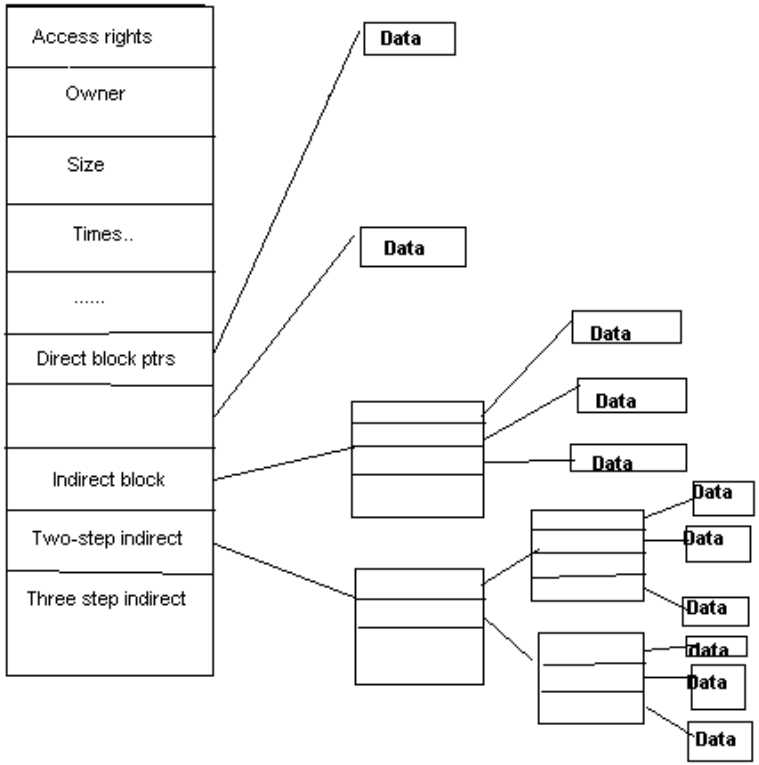
- What will happen if we delete the original file?

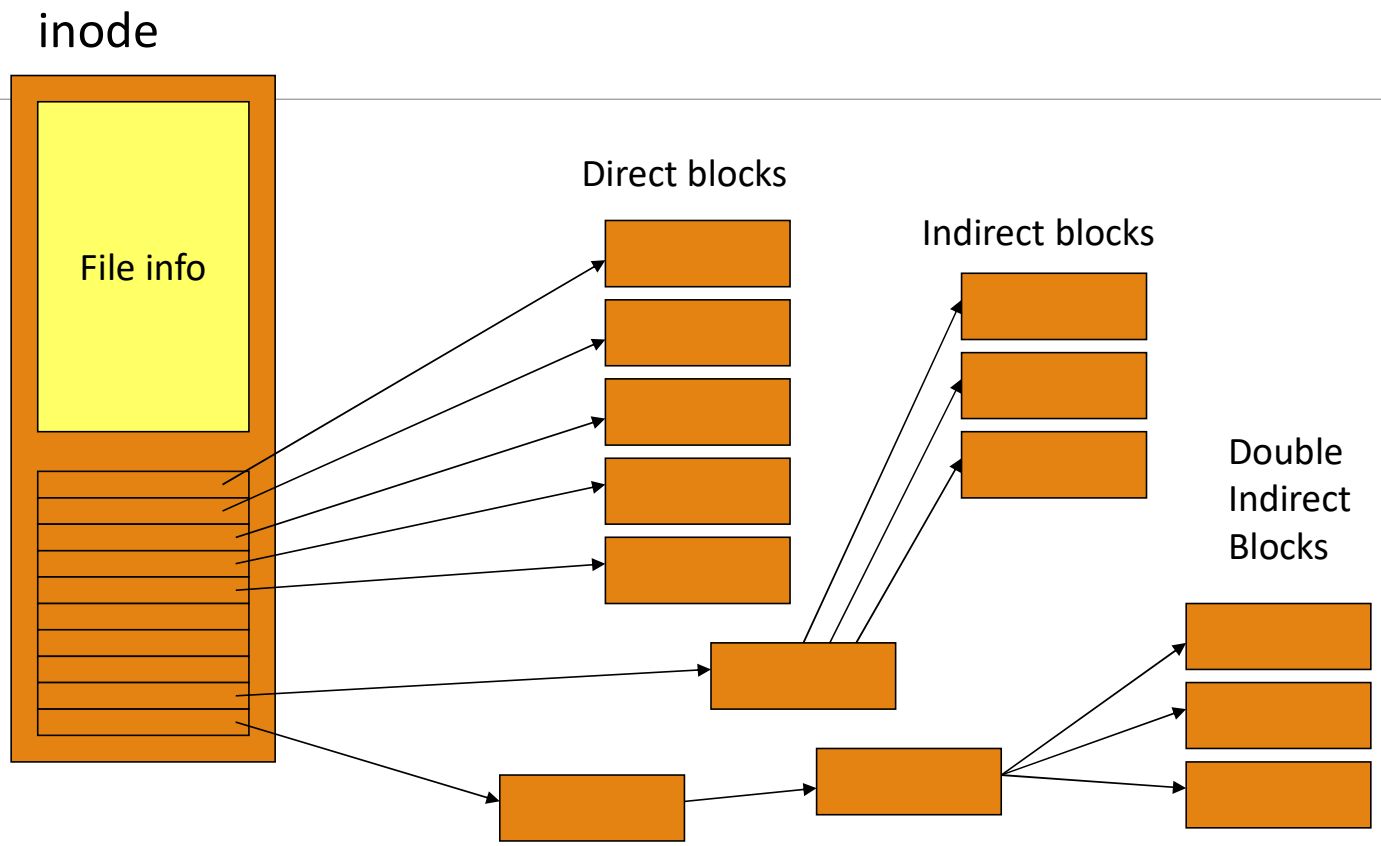
Symbolic link is similar to shortcut in Windows OS

Examples

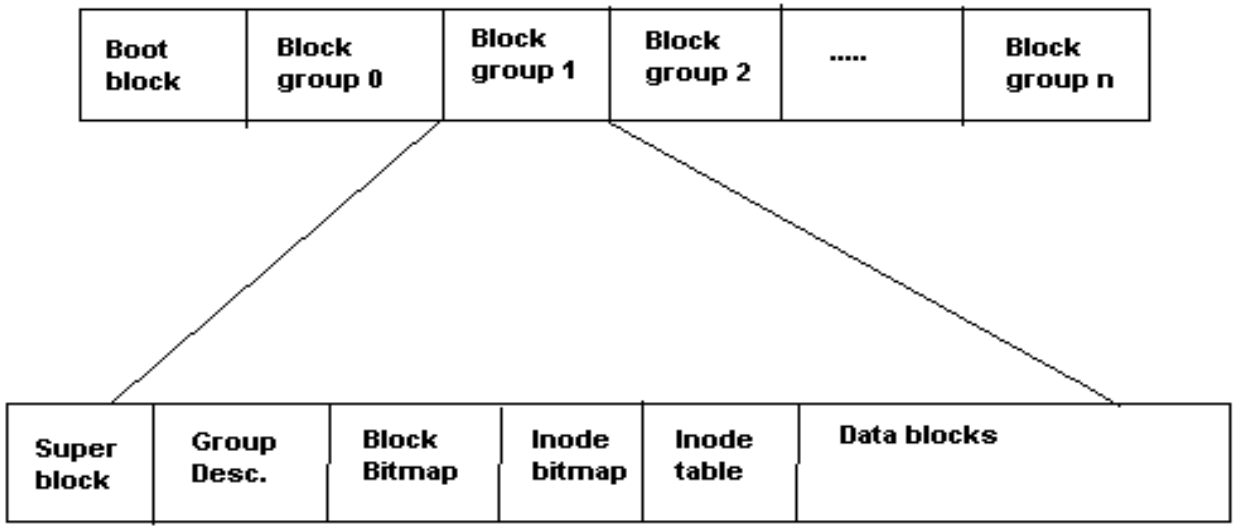
```
$ ls -l
-rw-r--r-- 1 tuananh user1 8 Feb 10 1:12 test.txt
$ ln test.txt link1
$ ln -s test.txt link2
$ ls -l link*
-rw-r--r-- 2 tuananh user1 16 Feb 10 1:12 link1
lrw-r--r-- 1 tuananh user1 16 Feb 10 1:13 link2->test.txt
```

Inode structure




inode
File info
Direct blocks
Indirect blocks
Double
Indirect
Blocks

Organisation on hard drive(s)



EXT2FS STRUCTURE

Search files

$ find <name of directory> expressions

- Allow to search files inside a directory (default is the working directory) with some conditions or commands to be executed on found files.

Conditions

- Name : -name <name>
- Permission: -perm <permission>
- Type : -type d/f/...
- Size: -size N
- Thời gian : -atime N, -mtime N, -ctime N

Executable commands on found files

- -print
- -exec command

Examples

$find /usr -name toto
- Find files named toto inside the directory /usr (including child directories of /usr)

$find /usr -name " *.c "
- List all files ending as « .c »

$find / -mtime 3
- Find all files been modified last 3 days

$find / -size 2000
- Find all files with the size of 1 MB (= 2000 block 512 B)

$find / -size +20M
- Find all files with the size over 20 MB

$find / -size -2GB
- Find all files with the size under 2GB

$find / -type f -user olivier -perm 755
- Find all files belonging to the user oliver and having the permission of 755

grep : find within lines

$grep [-options] expreg [files]

- Find the line satisfied the conditions of expressions.

- Options
  - -c : only show the total number of lines satisfied conditions
  - -I : ony show file name
  - -v : only show unsatisfied lines
  - -i : don’t care capitalised or not
  - -n : only show line number

Some special characters

- grep can use some special characters :

. Represent any character

* Repeat the previous character

^ Beginning of a line

$ End of a line

[...] list or range of finding characters

[^..] list or range of non-finding characters

- Note: to avoid confusion, we should place characters inside double quotation marks.

Examples

- $grep “^t" /etc/passwd
  - Find inside the file /etc/passwd all lines beginning with “t” character.
- $grep [^t] /etc/passwd
  - Find all lines not beginning with “t” character
- $grep “tuananh" /etc/passwd
  - Find all lines containing “tuananh”
- $ls -l /etc | grep "^d"
  - Show all child directories of /etc

Command tar – to save and backup files/directories

Example of using 'tar'

```
(1)# tar cvf file1.tar ./homework1
(2) # tar x file1.tar
(a)# tar cvfz backup.tar.gz file1 file2 file3

(c)# tar xvfz backup.tar.gz
```

Command gzip to compress and decompress

tar to save the whole directory as a single file

gzip to compress/decompress that file

Use gzip:

- gzip [options] [file]
- gzip –d: decompress

ACCOUNTS AND PERMISSION

User account

Normal users

Administrators

Group users

A user account needs

- Username, password, and home directory (/home/username)
- Group (each user need to belong to only one primary group through a user can belong to many groups)
- All user information is stored in the file: /etc/passwd

/etc/passwd

Username:password:UID:GID:Info:Home:Shell

Username: It is used when user logs in. It should be between 1 and 32 characters in length.

Password: An x character indicates that encrypted password is stored in /etc/shadow file.

User ID (UID): Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.

Group ID (GID): The primary group ID (stored in /etc/group file)

User ID Info: The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. This field use by finger command.

Home directory: The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes /

Command/shell: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. Please note that it does not have to be a shell.

/etc/shadow

User:Pwd:Last pwd change :Minimum:Maximum:Warn:Inactive :Expire

User name : It is your login name

Password: It your encrypted password. The password should be minimum 6-8 characters long including special characters/digits

Last password change (lastchanged): Days since Jan 1, 1970 that password was last changed

Minimum: The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password

Maximum: The maximum number of days the password is valid (after that user is forced to change his/her password)

Warn : The number of days before password is to expire that user is warned that his/her password must be changed

Inactive : The number of days after password expires that account is disabled

Expire : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used

Group users

Each user can belong to one or many groups

- One group = group name + member list
- Can share files among members of a group.
- The list of groups is stored in /etc/group
- root can create additional groups, aside default groups that the Linux OS created

/etc/group

group_name:Password:Group ID (GID): Group List

group_name: It is the name of group. If you run ls -l command, you will see this name printed in the group field.

Password: Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups. X means passwd is stored in /etc/gshadow

Group ID (GID): Each user must be assigned a group ID. You can see this number in your /etc/passwd file.

Group List: It is a list of user names of users who are members of the group. The user names, must be separated by commas.

/etc/gshadow

Group name — The name of the group. Used by various utility programs as a human-readable identifier for the group.

Encrypted password — The encrypted password for the group. If set, non-members of the group can join the group by typing the password for that group using the newgrp command. If the value of this field is !, then no user is allowed to access the group using the newgrp command. A value of !! is treated the same as a value of ! — however, it also indicates that a password has never been set before. If the value is null, only group members can log into the group.

Group administrators — Group members listed here (in a comma delimited list) can add or remove group members using the gpasswd command.

Group members — Group members listed here (in a comma delimited list) are regular, non-administrative members of the group.

Tools to manage user/group

useradd/mod/del

passwd

groupadd/mod/del

gpasswd

sg/newgrp

su

users/groups

id

Ownership

Each file belongs to a user and a group

- The file/directory creator will be the owner and the primary group of the owner will be the group owner.

The ownership allows to determine the permission of a user to files or directories.

Access permission

r : read

- Allow to read the content of a file or directory

w : write

- Allow to change the content of a file
- Allow to add or remove files inside a directory

x : execute

- Allow to execute a binary file
- Allow to change to a directory

Permission groups

There are three user based permission groups each file/directory :

- u (owner) : the owner of the file
- g (group) : users belong to the primary group of the owner
- o (others) : other users.

Each permission group has a set of permission (r, w, x).

Examples

```
$ ls -l
----rw-rw- 1 tuananh  user1   16 Feb 10 19:12 test1.txt
-rw-rw-rw- 1 tuananh  user1   16 Feb 10 19:12 test2.txt
drw-r--r-- 2 tuananh  user1  512 Feb 10 19:14 vanban
$ whoami
tuananh
$ cat test1.txt
cat: test1.txt: Permission denied
$ cat test2.txt
Un fichier de test
$ cp test2.txt vanban
cp: vanban: Permission denied
```

Some notes

To add more files to a directory, you need « w » permission

To delete or move a file, a user needs « w » permission of that file

- The file deletion depends on the permission of the directory that that file belongs to

To secure data, the owner can even remove the « r » permission to other users.

To limit access to file system, user can remove « x » permission to the root of the file system (/).

Change permission (1)

$chmod <mode> <files>

```
set_uid set-gid sticky user group other
                       rwx   --x   --x
   1       1      0    111   001   001
           6            7     1     1
```

```
$ chmod  6711  test
$ ls -l test
-rws--s--x  1  tuananh  user1  Mar 10 10:20  test
$ chmod  711  test
$ ls -l test
-rwx--x--x  1  tuananh  user1  Mar 10 10:20  test
```

Change permission (2)

```
$chmod <ugoa><+-=><rwsx> <files>
```

u | g | o | a (all)

Operation

- + (add one or some permission)
- - (remove one or some permission)
- = (assign one or some permission)

Permission = r | w | x | s

Examples

```
$ ls -l test.txt
-rw-rw-r-- 1 tuananh user1  150 Mar 19 19:12 test.txt
$ chmod o+w test.txt
$ ls -l test.txt
-rw-rw-rw- 1 tuananh user1  150 Mar 19 19:12 test.txt
$ chmod a-rw test.txt
$ ls -l test.txt
---------- 1 tuananh user1  150 Mar 19 19:12 test.txt
$ cat test.txt
cat: test.txt: Permission denied
```

Default permission while creating a file

You can view or change the default permission of a file while creating by using command umask

```
$umask
```

```
022
```

- 0 means no limitation (rwx)
- 2 mean no writing permission (w) but can read or execute (r-w).

```
$umask 022
```

Change the owner and group

$chown [-R] <utilisateur> <files>

- Change the ownber

$chgrp <group> <files>

- Change the group
- Use option –R to change the ownership of child files/folders of a folder

Those commands can only be executed by the root user

Some special permission with binary files

set-uid: -rw**s** --- ---

- Program will be run under the permission of the owner

set-gid: - --- rw**s** ---

- Program will be run under the permission of the group owner

bit sticky

- Program can only have one instance in memory

Examples

```
$ ls -l /etc/passwd
-rw-rw---- 1 root  root  568 Feb 10 19:12 passwd
$ ls -l /bin/passwd
-rwsrws--x 1 root  root 3634 Feb 10 19:12 passwd
```

When a user change their password (/bin/passwd), that user « borrow » the root permission to change the password of their account