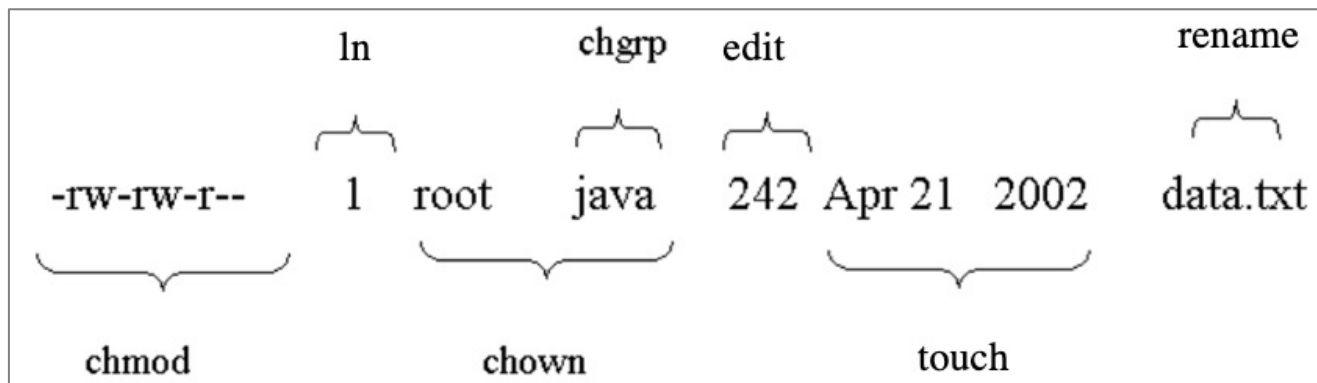

Operating Systems

Practice Work 04. Linux Permissions. SUID/SGID/Sticky Bits.

Changing a File's Parameters

- **inodes** contain a lot of information about a file
 - mode and type of file
 - number of links to the file
 - owner's UID
 - owners GID
 - number of bytes in file
 - last accessed, modified, inode changed times
 - physical disk addresses (direct and indirect blocks)
 - number of blocks
 - shadow inode with extend access information (ACL)
- **Test all inode change commands**: ls -al , ln, rename, editor, chown, chgrp, chmod, touch, chattr, lsattr



File's & Directory Permissions

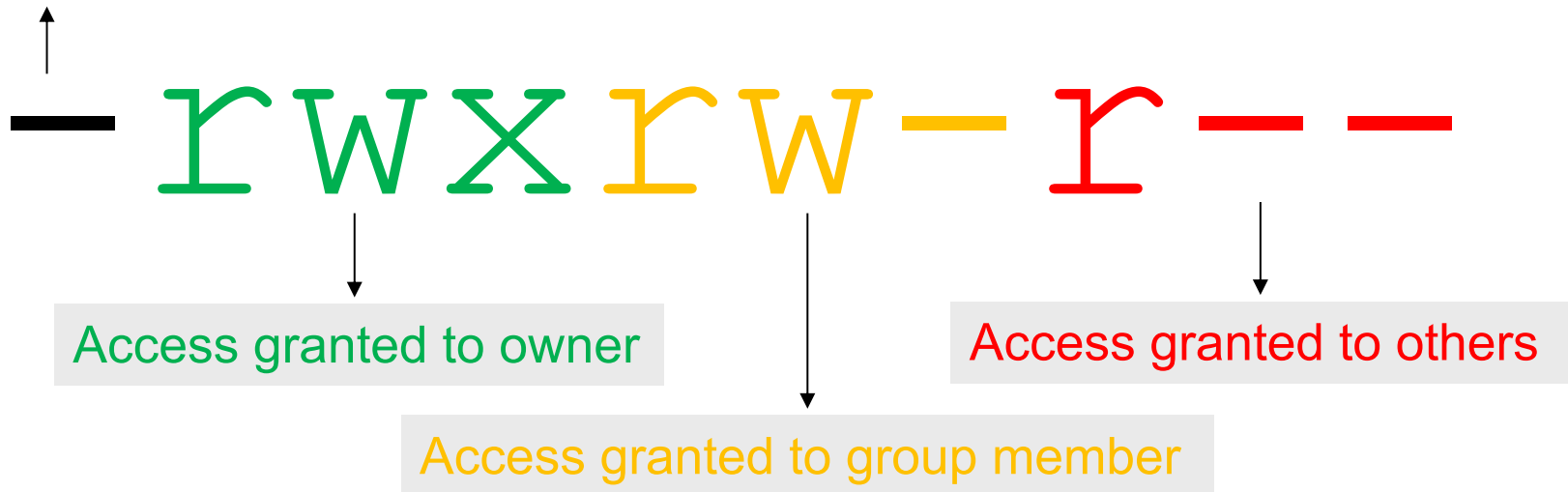
File type

- : plain file
d : directory
c : character device (tty, printer)
b : block device (disk, CD-ROM)
l : symbolic link
s : socket
=, p : pipe (FIFO)

How to read a `-rwxrw-r-` permission

r : read / w : write / x : execute

user	group	other
rwx	rw-	r--
4+2+1	4+2+0	4+0+0
7	6	4



chmod: Changing a File's Permissions

- When you create a file, its initial permissions depend on your **umask** value (discussed later).
- You can change a file's permissions with the **chmod** command or the **chmod()** system call.
- If you are logged in as the **superuser**, you can change the permissions of **any file**.
- You can change a file's permissions only if you are the file's **owner**.
- The **symbolic form** of the chmod command:

chmod [-R] [agou] [+-=] [rwxst] *filelist*

- This command changes the permissions of filelist (a single file or a group of files).
- The letters **agou** specify **whose** privileges are being modified. You may provide one or more.
- The symbols **+-=** specify **what** is to be done with the privilege. You must type only 1 symbol
- The last letters **rwxst** specify **which** privilege will be modified: read, write, execute bits and suid, sgid, sticky bits.
- The **-R option** causes the chmod command to run **recursively**. If you specify a directory in filelist, that directory's permissions change, as do all of the files contained in that directory. If the directory contains any subdirectories, the process is repeated.

chmod: Changing a File's Permissions

Letter Meaning

a	Modifies privileges for all users
g	Modifies group privileges
o	Modifies others' privileges
u	Modifies the owner's privileges

Symbol Meaning

+	Adds to the current privilege
-	Removes from the current privilege
=	Replaces the current privilege

Letter Meaning

r	Read access
w	Write access
x	Execute access
s	SUID or SGID
t	Sticky bit ^[9]

Examples:

```
$ chmod a+rwx file
$ chmod u=rw file
$ chmod ag-r,o+wx f1 f2
$ chmod u=s file
$ chmod ug+wx,o-t file
$ chmod a+rwx,go-wx f1
$ chmod a-rwx,u=rwx f1
```

```
% ls -l notes
-rw-r--r-- 1 sian      biochem    4320 Feb  9 13:20 notes
% chmod g+w notes
% ls -l notes
-rw-rw-r-- 1 sian      biochem    4320 Feb  9 13:20 notes
%
```

chmod: Changing a File's Permissions

- This format is called the **absolute (or numeric, or octal) form** of the chmod command:

chmod [-R] mode filelist

- The mode to which you wish to set the file, expressed as an **octal value** with 3 octal numerals. Every octal numeral is interpreted as 3 binary bits (rwx)

Task 1. Find binary and symbolic permissions:

\$ chmod 000 file	000 000 000	-----
\$ chmod 640 file	110 100 000	-rw-r-----
\$ chmod 123 file	001 010 011	---x-w--wx
\$ chmod 777 file	111 111 111	-rwxrwxrwx
\$ chmod 752 file	?	?
\$ chmod 246 file	?	?
\$ chmod 412 file	?	?
\$ chmod 206 file	?	?
\$ chmod 213 file	?	?
\$ chmod 357 file	?	?
\$ chmod 234 file	?	?
\$ chmod 325 file	?	?
\$ chmod 567 file	?	?
\$ chmod -R 341 dir fl a*.txt	?	?

umask

- The umask (Unix shorthand for "**user file-creation mode mask**") is a 3 or 4 octal number that Unix uses to determine the file permission **for newly created files and directory**.

\$ umask NNN

- Every process has its own umask, inherited from its parent process.
- **By default**, Linux/Unix specify an octal standard mode of **666** (any user can read or write the file) when they create new files.
- **By default**, Linux/Unix specify an octal standard mode of **777** (any user can read, write, or look the directory) when they create new directory.
- For Result Permissions using bitwise AND with the **default permissions** and the complement of the umask value (**the bits that are not set in the umask**).
- Normally, you or your system administrator set the umask in your `.login`, `.cshrc`, or `.profile` files, or in the system `/etc/profile` or `/etc/cshrc` file. For example, you may have a line that looks like this in one of your startup files:

```
# Set the user's umask
umask 033
```
- The most common umask values are 022, 027, and 077. A umask value of 022 lets the owner both read and write all newly created files, but everybody else can only read them.

umask

Rules.

File Result Permissions Bits = NOT(umask Bits) AND (File Standard Permissions Bits)

Directory Result Permissions Bits = NOT(umask Bits) AND (Directory Standard Permissions Bits)

Example.

After umask 174

174 (001 111 100)	Umask
- 603 (110 000 011)	NOT(Umask)
* 666 (110 110 110)	Default file-creation mode
602 (110 000 010)	Result mode for new file
174 (001 111 100)	Umask
- 603 (110 000 011)	NOT(Umask)
* 777 (111 111 111)	Default directory-creation mode
603 (110 000 011)	Result mode for new directory

Set and test umask value.

```
$ umask          # current umask
0002
$ umask 174      # new umask
$ mkdir dir1     # create new directory
$ touch file1    # create new file
$ ls -l          # list permissions
drw-----wx 2 std std 512 Sep 1 20:59 dir1
-rw-----w- 1 dave dave 0 Sep 1 20:59 file1
```

Task 2.

Find result permissions for new files and directory after command:

Command	Directory	File
\$ umask 022	755	644
\$ umask 077	700	600
\$ umask 123	?	?
\$ umask 325	?	?
\$ umask 547	?	?
\$ umask 406	?	?
\$ umask 737	?	?
\$ umask 100	?	?
\$ umask 372	?	?
\$ umask 077	?	?
\$ umask 345	?	?
\$ umask 704	?	?

SUID/SGID/sticky bits

- SUID (set uid)
 - Processes are granted access to system resources based on user who owns the file.
- SGID (set gid)
 - (For file) Same with SUID except group is affected.
 - (For directory) Files created in that directory will have their group set to the directory's group.
- sticky bit
 - If set on a directory, then a user may only delete files that he owns or for which he has explicit write permission granted, even when he has write access to the directory. (e.g. /tmp)
- The access permission status that is displayed using the “ls -l” command does not have a section for special permissions
- However, since special permissions required “execute”, they mask the execute permission when displayed using the “ls -l” command.

r	w	x	r	w	x	r	w	x
↓		↓		↓		↓		↓
r	w	S	r	w	S	r	w	t
		SUID			SGID			STICKY BIT

SUID/SGID/sticky bits

Use the chmod command with 4 numerals octal mode

suid	sgid	stb	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1
7			7			7			7		
Special			user			group			others		

Task 3. Find all of the SUID and SGID files for Your UbuntuMini.

```
# find / \( -perm -004000 -o -perm -002000 \) -type f -print
```

SUID/SGID/sticky bits

Task 4. Find binary and symbolic permissions value after command:

\$ chmod 7000 file	111 000 000 000	---S--S--T
\$ chmod 5740 file	101 111 100 000	-rwsr----T
\$ chmod 1123 file	001 001 010 011	---s-w--wx
\$ chmod 0777 file	000 111 111 111	-rwxrwxrwx
\$ chmod 2752 file	?	?
\$ chmod 3246 file	?	?
\$ chmod 4357 file	?	?
\$ chmod 4457 file	?	?
\$ chmod 2143 file	?	?
\$ chmod 5475 file	?	?
\$ chmod 5234 file	?	?
\$ chmod 6567 file	?	?
\$ chmod 3210 file	?	?
\$ chmod -R 5640 dir fl a*.txt	?	?

SUID/SGID/sticky bits

Task 5. Find binary and absolute permissions value:

---S--S--T	111 000 000 000	7000
-rwsr---T	101 111 100 000	5740
---s-w--wx	001 001 010 011	1123
-rwxrwxrwx	000 111 111 111	0777
-r-xrwS-wt	?	?
-rw---sr-T	?	?
----rwx-wx	?	?
--wS-wx-wx	?	?
-r-x-w-r-t	?	?
--w-r-S-w-	?	?
----r-s--x	?	?
-rwsr-x--T	?	?
-r-S-ws-wx	?	?
drw-rwsr-t	?	?

The End