

OPERATING SYSTEM: UNIX/LINUX



Course 2

Osman SALEM

Maître de conférences - HDR

osman.salem@u-paris.fr



1

Permission levels

- "r" means "read only" permission
- "w" means "write" permission
- "x" means "execute" permission
 - In case of directory, "x" grants permission to list directory contents

2



Access Permissions

- Each of the three permissions are assigned to three defined categories of users.
- The categories are:
 - owner — The owner of the file or application.
 - group — The group that owns the file or application.
 - others — All users with access to the system.

3



File Permissions

```
jean@epita: ~/epita1
jean@epita:~/epita1$ ls -l
total 8
-rw-rw-r-- 1 jean jean 11 mars  15 23:30 file1
-rw-rw-r-- 1 jean jean 45 mars  15 23:30 file1.txt
jean@epita:~/epita1$
```

User (you)

4

File Permissions

```
jean@epita: ~/epita1
jean@epita:~/epita1$ ls -l
total 8
-rw-rw-r-- 1 jean jean 11 mars  15 23:30 file1
-rw-rw-r-- 1 jean jean 45 mars  15 23:30 file1.txt
jean@epita:~/epita1$
```

Group

5

File Permissions

```
jean@epita: ~/epita1
jean@epita:~/epita1$ ls -l
total 8
-rw-rw-r-- 1 jean jean 11 mars  15 23:30 file1
-rw-rw-r-- 1 jean jean 45 mars  15 23:30 file1.txt
jean@epita:~/epita1$
```

"The World": other

6



Command: chmod

- If you own the file, you can change its permissions with "chmod"
 - Syntax: `chmod [user/group/others/all]+[permission] [file(s)]`
 - Below we grant execute permission to all:

```
jean@epita: ~/epita1
jean@epita:~/epita1$ ls -l
total 8
-rw-rw-r-- 1 jean jean 11 mars 15 23:30 file1
-rw-rw-r-- 1 jean jean 45 mars 15 23:30 file1.txt
jean@epita:~/epita1$ chmod a+x file1
jean@epita:~/epita1$ ls -l
total 8
-rwxrwxr-x 1 jean jean 11 mars 15 23:30 file1
-rw-rw-r-- 1 jean jean 45 mars 15 23:30 file1.txt
jean@epita:~/epita1$
```

7



Permissions

- **u**: user or file creator
- **g**: group of users
- **o**: others
- **a**: all
- Permissions
 - **r**, read, 4 in octal
 - **w**, write, 2 in octal
 - **x**, execution, 1 in octal
 - ...|...|...
 - **u g o**
 - rwx r-x r-x or 755
 - `chmod permissions fichiers`
 - **u, g et o** for user, group, other, **a** for all
 - **a** (=all) all users
 - **+** = addition/suppression
 - `chmod g+w,o+r a.png`
 - `chmod g-x,o-x share`
 - `chmod a+x filename.sh`

8



Access Permission of File

- The #'s can be:
 - 0 = Nothing
 - 1 = Execute
 - 2 = Write
 - 3 = Execute & Write (2 + 1)
 - 4 = Read
 - 5 = Execute & Read (4 + 1)
 - 6 = Read & Write (4 + 2)
 - 7 = Execute & Read & Write (4 + 2 + 1)

9



Permissions

- Exemples to modify permissions:
 - `chmod ug+w fichier`
 - `chmod go-rwx fichier`
 - `chmod u=rwx,g=rw,o=r fichiers`
 - `chmod u=rwx,g=r fichiers`
 - `chmod u=rwx,g=r,o= fichiers`
 - `chmod 700 /home/rep-a-moi`
 - `chmod 764 test`

Table 7-6

Example	Result
<code>chmod 754 hello.txt</code>	All permissions for the owner, read and execute for the group, and read for all other users.
<code>chmod 777 hello.txt</code>	All users (user, group, and others) receive all permissions.

10



Permissions

Table 7-3

Example	Result
<code>chmod u+x</code>	The owner is given permission to execute the file.
<code>chmod g=rw</code>	All group members can read and write to the file.
<code>chmod u=rwx</code>	The owner receives all permissions.
<code>chmod u=rwx,g=rw,o=r</code>	All permissions for the owner, read and write for the group, and read for all other users.
<code>chmod +x</code>	All users (owner, group, others) receive executable permission (depending on umask).
<code>chmod a+x</code>	All users (owner, group, and others) receive executable permission (a for all).



Permissions

`chmod 751 myfile`

change the file permissions to rwx for owner, rx for group and x for others

`chmod go=+r myfile`

Add read permission for the group and others



Access Permission of File/Directory

- The ownership of the file or directory can be changed using the command
`chown <owner> <file/directory name>`
- The group of the file or directory can be changed using the command
`chgrp <group> <file/directory name>`

13



Change the File Ownership with chown and chgrp

- User root can use chown and chgrp as follows:
 - `chown new_user.new_group file`
 - `chown new_user file`
 - `chown .new_group file`
 - `chgrp new_group file`

14



Wildcards

- A means of matching patterns; usually in file names or when searching contents of a text file
- Sometimes referred to as regular expressions
- Examples:
 - “*” An asterisk – matches any number of letters or digits
 - “?” Question mark – matches a single letter or digit
 - “[]” Square brackets – matches range of letters or digits

15



Wildcard Examples

- list all temp tablespace data files that contain two alphanumeric characters after the word temp in the name. i.e. temp01.dbf, tempAB.dbf
ls temp?? .dbf
- list all files that start with a letter followed by a digit followed by any combination of letters or characters and ends with .log; i.e. a2test.log
ls [a-z][0-9]*.log

16



INPUT & OUTPUT

- 'which' - searches current \$PATH for executable
 - 'which cat' && 'which ls'
- 'echo \$PATH' - reveals the current \$PATH
- Redirection:
 - '<' - INPUT - Usually defaults to a source file
 - '>' - OUTPUT - clobbers target file
 - '>>' - APPEND - appends to target file if it exists and creates it if it doesn't

17



Input/Output Redirection ("piping")

- Programs can output to other programs
- Called "piping"
- "program_a | program_b"
 - program_a's output becomes program_b's input
- "program_a > file.txt"
 - program_a's output is written to a file called "file.txt"
- "program_a < input.txt"
 - program_a gets its input from a file called "input.txt"

Table 6-3

Channel	Number Assigned
Standard input (stdin)	0
Standard output (stdout)	1
Standard error output (stderr)	2

18



Standard Error

- a.k.a stderr, filehandle 2
- Default location is terminal screen
- Redirect stderr to a new file with the redirect operator “2>”
- Append to an existing file with the concatenation operator “2>>”
- Redirect stderr to the same location as stdout using “2>&1”
- Pipes typically not used with stderr

Table 6-4

Redirection Character	Description
<	Redirects standard input
>	Redirects standard output (> without a preceding number is just an abbreviation for 1>)
2>	Redirects standard error output

19



Use Piping and Redirection

- To display the output of a command on the screen as well as written to a file, use **tee**
 - ls -l | tee output-file

20



Use Piping and Redirection (continued)

Table 6-5

Link	Result
<code>command1 && command2</code>	<code>command2</code> is executed only if <code>command1</code> is completed without any errors.
<code>command1 command2</code>	<code>command2</code> is executed only if <code>command1</code> is completed with an error.

21



Linux File Management and Viewing

- File compression, backing up and restoring
- `gzip` - zip a file to a gz file.
- `gunzip` - unzip a gz file \Leftrightarrow `gzip -d filename.txt`
- `zip` - Compresses a file to a .zip file (`zip -r outfile.zip infile`)
- `unzip` - Uncompresses a file with .zip extension.
- `bzip2` - efficient zip (30% smaller than zip) with .bz2
- `bunzip2` decompresses files compressed with bzip2 and removes the suffix **`bz2`**
 - Equivalent to `bzip2 -d file`
 - `bunzip2 file.bz2`

22



Linux File Management and Viewing

- **tar** Archives files and directories. Can store files and directories on tapes.
 - Ex: `tar -zcvf <destination> <files/directories>` - Archive copy groups of files.
 - `tar -zxvf <compressed file>` to uncompress
- **tar** is the most commonly used tool for data backup
 - It requires an option, the name of the archive (or the device file) to be written to, and the name of the directory to back up
 - `tar -cvf file.tgz /home`
 - `tar -cvf /backup/etc.tar /etc`
 - `tar -czvf /backup/etc.tgz /etc`
 - `tar -cvf /dev/st0 /home -X exclude.files`
 - `tar -tvf /dev/st0`
 - `tar -cvjf file.tbz2 file1 file2`
 - j: compression through bzip2
 - tbz ou tb2 or .tar.bz2 are the same

23



Linux File Management and Viewing

- `tar -xvjf fichier.tar.bz2`

- `gzip tarball.tar`
- `gunzip tarball.tar.gz`
- `bzip2 tarball.tar`
- `bunzip2 tarball.tar.bz2`

24