FIT9137
Introduction to Computer Architecture and Networks

Week 2: Computer Architecture and Operating Systems

Dr. Muhammed Esgin

www.shutterstock.com · 325009091

Unit LOs

Week 2 Lesson - Computer Architecture and Operating Systems

Weekly Learning Outcomes:

At the end of this week, you will be able to:

- Explain the concept of Operating Systems (OS) and their major functions
- Be aware of the history of Unix
- Work with a command line in Unix and/or Windows

Today

- Major functions of an O/S - file management
- We will use ***Unix*** and/or ***Linux*** as case-studies to illustrate how operating systems work.

Activity A

FLUX: OS functions

Key OS functions are:

1. File management
2. Memory management
3. Process management
4. All of the above
5. None of the above

To participate, go to

flux.qa/JSBJLK

Major Functions of an O/S

- *File* Management
- *Memory* Management
- *Process* Management

File Management

- Operating systems typically operate on the concept of a *file* – a collection of logically related data.
- Typical tasks of the File Management System:
  - Controlling transfer of data to and from secondary storage - activity A&B
  - Controlling security of file access - activity A
  - Keeping track of storage space and maintaining file directories - activity B
  - Provide sharing mechanisms for files
  - (Possibly) provide recovery and restoration mechanisms

File types – as shown by **ls -l**

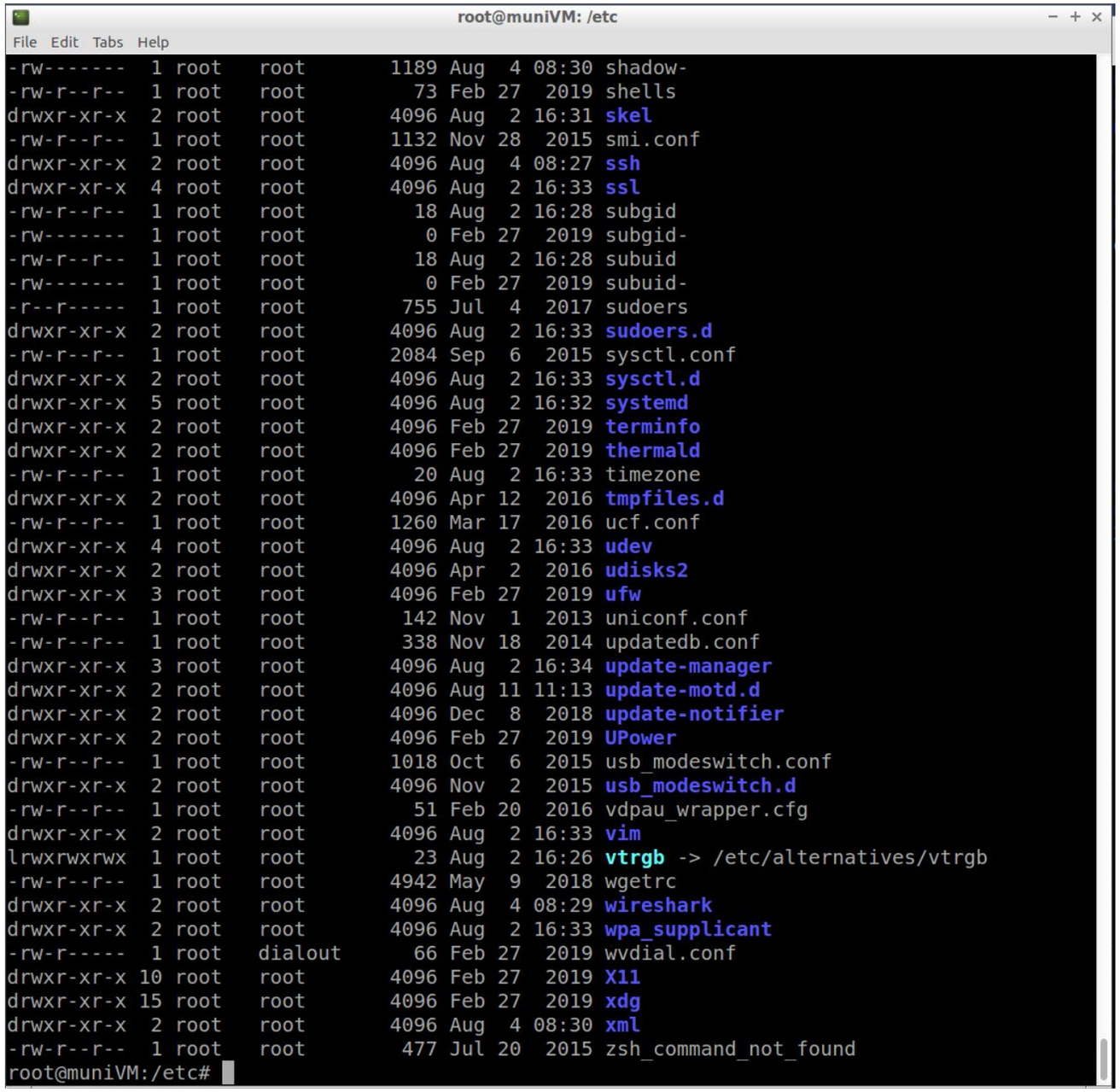
- **Character File type**
  - \- regular (ordinary) file
  - d directory
  - b buffered special file (e.g. a disk drive)
  - c unbuffered special file (e.g. a terminal)
  - l symbolic link
  - p pipe
  - s socket

The command ***ls –l*** will show things such as file types, permissions, file sizes, modification dates, etc

The ***file*** command returns the type of the content of the given file name :

$ file exercise

exercise: ascii text



File permissions

- A total of nine (**9**) binary ***bits*** representing the permissions:

- **user** **group** **others**
  r/- w/- x/- r/- w/- x/- r/- w/- x/-

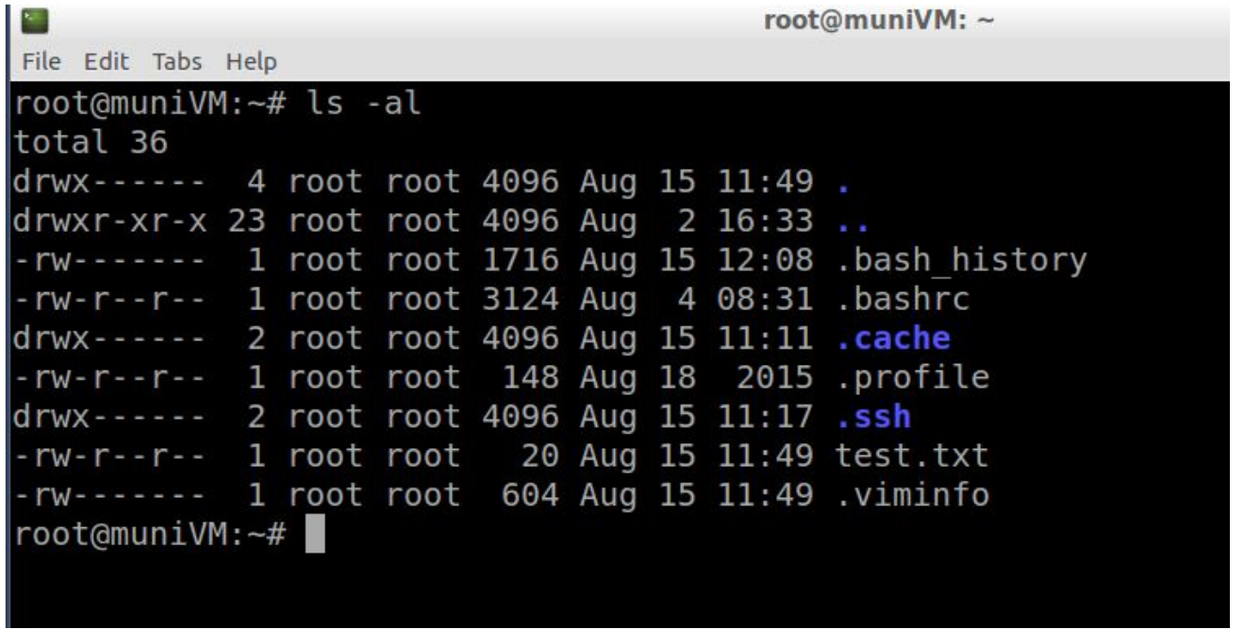
- a "**—**" indicates the permission is "off", e.g.

  rwxrw—r——

- an example to follow shortly

A user can choose to restrict access to their **files/directories**, so that other users may or may not access them.

A ***Superuser*** (the "**root**" user) **has access to all files** irrespective of permissions.



File permissions : example using *ls -al*

- # **ls -al** example

```
root@muniVM:~# ls -al
total 36
drwx------  4 root root 4096 Aug 15 11:49 .
drwxr-xr-x 23 root root 4096 Aug  2 16:33 ..
-rw-------  1 root root 1716 Aug 15 12:08 .bash_history
-rw-r--r--  1 root root 3124 Aug  4 08:31 .bashrc
drwx------  2 root root 4096 Aug 15 11:11 .cache
-rw-r--r--  1 root root  148 Aug 18  2015 .profile
drwx------  2 root root 4096 Aug 15 11:17 .ssh
-rw-r--r--  1 root root   20 Aug 15 11:49 test.txt
-rw-------  1 root root  604 Aug 15 11:49 .viminfo
root@muniVM:~#
```

<u>Output explanations:</u>

- The permission mode of this file is **read** and **write** for the **owner**, **read** for the **group,** and **read** only for **others**
- There is 1 **hard link**
- The user-id of the file's owner is ***root***
- The group-id of the file is ***root***
- The size of the file is **20** "blocks" – NB. block size can vary between systems
- The file was last modified on **Aug 15 4 11:49**
- The file name is ***test.txt***

The option "*-l*" in the command above is to request the output in ***long listing,*** format
The option "*-a*" in the command above is to request the output in ***all files (including the hidden files,*** format
There is another option, "*-h*", which will make ls display sizes in "human readable" format (eg. 8K, 555M, 4G, etc.)

Some commonly used **Octal** (base-8) values for file permissions

```
700     ==>     /* owner:  rwx------ */
400     ==>     /* owner:  r--------- */
200     ==>     /* owner:  -w-------- */
100     ==>     /* owner:  --x------- */

070     ==>     /* group:  ---rwx--- */
040     ==>     /* group:  ---r----- */
020     ==>     /* group:  ---w----- */
010     ==>     /* group:  ---x----- */

007     ==>     /* others: ------rwx */
004     ==>     /* others: ------r-- */
002     ==>     /* others: ------w-- */
001     ==>     /* others: ------x-- */

4000    ==>     /* set user id on execution */
2000    ==>     /* set group id on execution */
```

Values of 0-7 can be used to indicate if a particular bit is “on or “off”

This sort of “shortcut” is commonly used in Unix commands

a ‘1’ means the corresponding bit is “on”

a ‘0’ means the corresponding bit is “off”

Examples :

		4	0	0
chmod 400 file1	====>	r--	---	---
chmod 764 file2	====>	rwx	rw-	r--

(4 0 0 == 100 000 000)

Activity B

FLUX: Links

Which statement is true about links?

A. Hardlinks still work even if the file they pointed to does not exist

B. Symlinks can exist even if the file they pointed to does not exist!

C. All of the above

D. None of the above.

To participate, go to

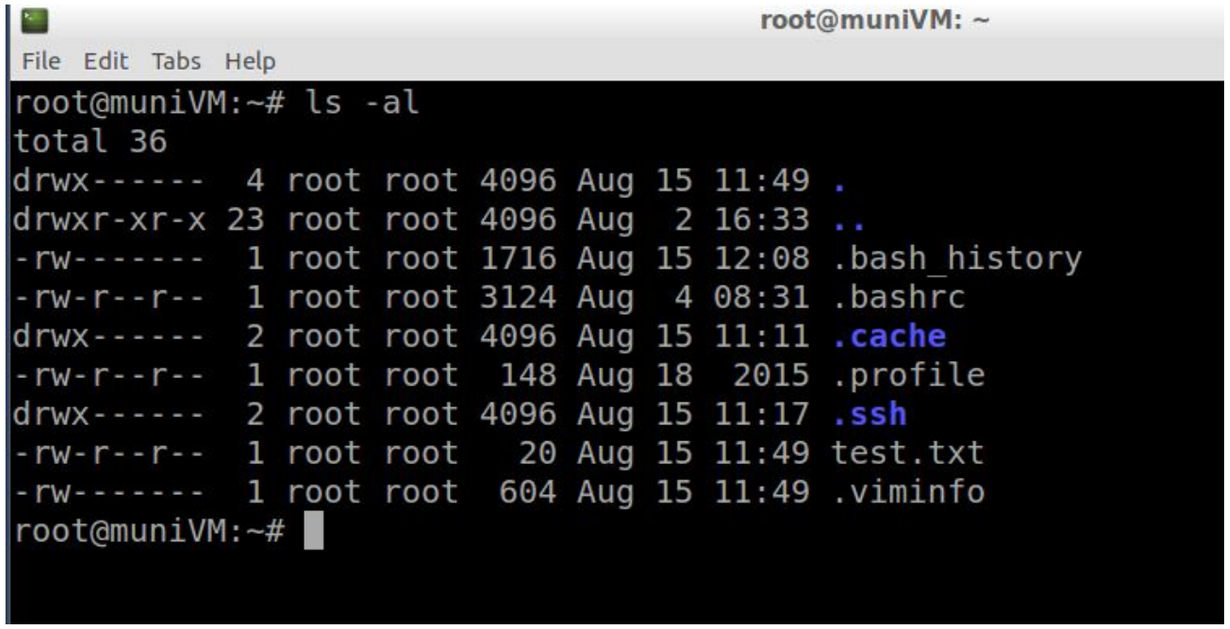
flux.qa/JSBJLK

File permissions

- **Three levels of permissions :**
  - the ***user***
  - the user's ***group*** and
  - ***others*** who have account on the system

- **Three kinds of permissions (for each level) :**
  - **read**, **write** and ***execute***
  - these have the usual meaning for ordinary files (for directory files – x means something different – more later)

Hence 9 different combinations in total



Workshop Summary

- Major functions of OS - file management
- Post-class activity