Linux Basics

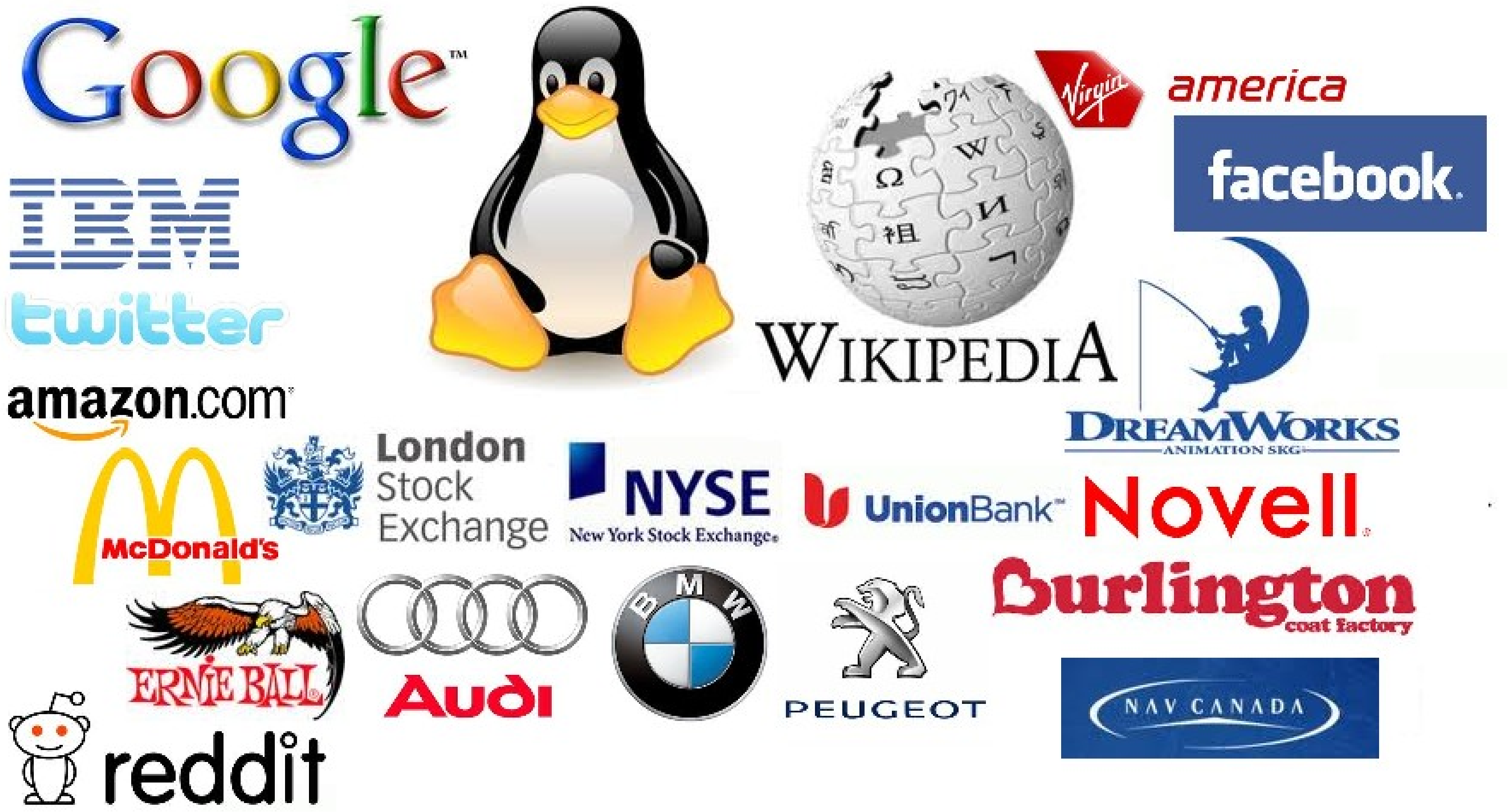
Software Testing

www.softacademy.bg

Agenda

- Who uses Linux and why?
- Distributions
- Remote Access
- Linux file system
- Basic Commands
- Other Useful Commands

Who uses Linux?



Why use Linux?

- Free, open source
- Fast, reliable, secure
- No viruses & malware
- Integration, modification, maintenance

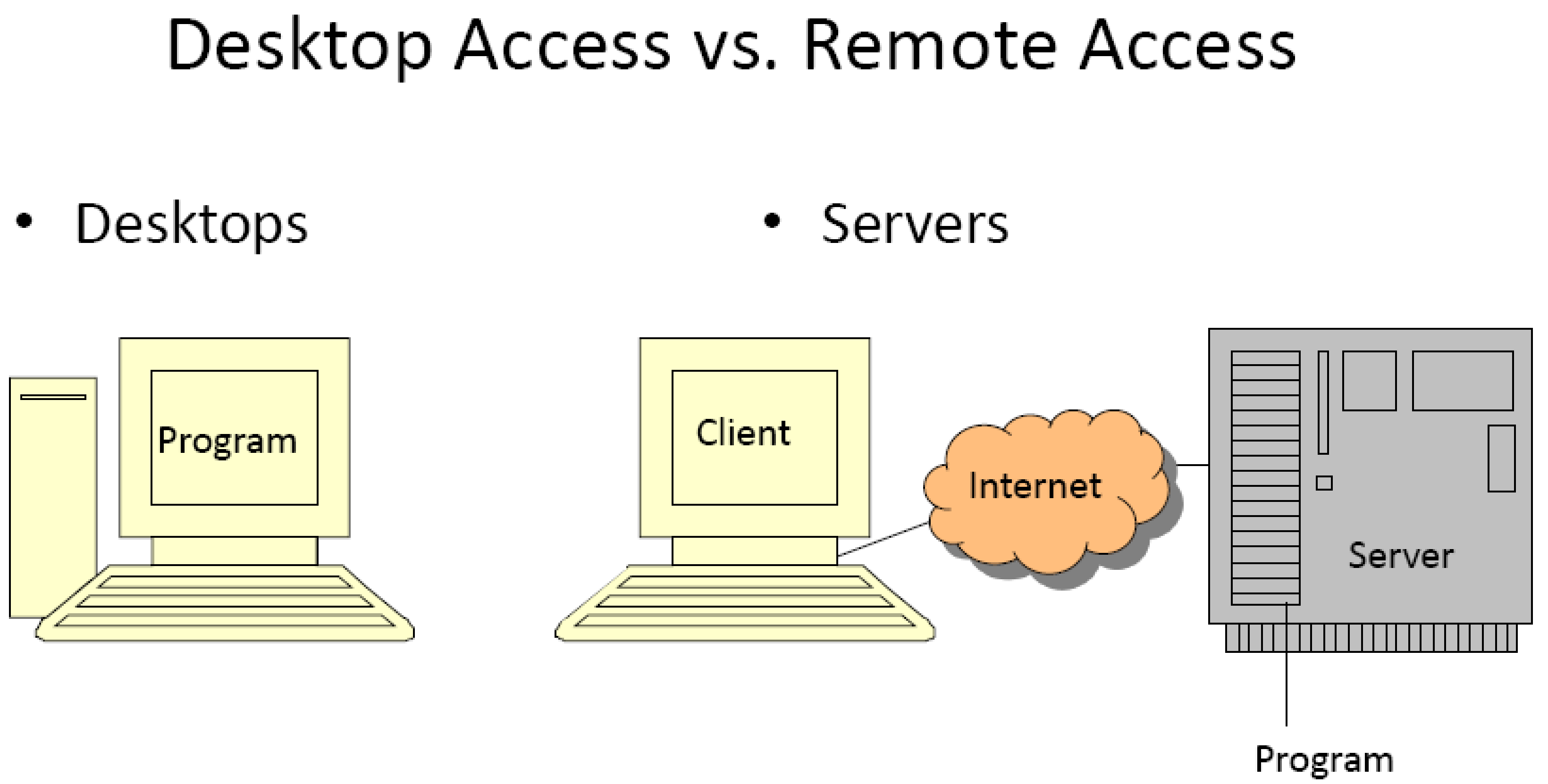
- Not convinced? Read more at http://whylinuxisbetter.net

Linux distributions

A Linux **distribution** is a collection of (usually open source) software on top of a Linux kernel. A distribution (or short, distro) can bundle server software, system management tools, documentation and many desktop applications in a **central secure software Repository**

- **Red Hat** is a billion dollar commercial Linux company that puts a lot of effort in developing Linux. They have hundreds of Linux specialists and are known for their excellent support. They give their products (Red Hat Enterprise Linux and Fedora) away for free. While **Red Hat Enterprise Linux** (RHEL) is well tested before release and supported for up to seven years after release, **Fedora** is a distro with faster updates but without support.
- **Ubuntu** - Canonical started sending out free compact discs with **Ubuntu** Linux in 2004 and quickly became popular for home users (many switching from Microsoft Windows). Canonical wants Ubuntu to be an easy to use graphical Linux desktop without need to ever see a command line. Of course they also want to make a profit by selling support for Ubuntu.
- **Debian** - There is no company behind **Debian**. Instead there are thousands of well organized developers that elect a Debian Project Leader every two years. Debian is seen as one of the most stable Linux distributions. It is also the basis of every release of Ubuntu. Debian comes in three versions: stable, testing and unstable. Every Debian release is named after a character in the movie Toy Story.
- **Other** - Distributions like CentOS, Oracle Enterprise Linux and Scientific Linux are based on Red Hat Enterprise Linux and share many of the same principles, directories and system administration techniques. Linux Mint, Edubuntu and many other *buntu named distributions are based on Ubuntu and thus share a lot with Debian. There are hundreds of other Linux distributions.

Remote access



Linux remote access

How to access Linux systems remotely?

- SSH (Secure Shell) - a protocol used to securely log onto remote systems. It is the most common way to access remote Linux and Unix-like servers.
  - From Linux (built-in) -> just run command in terminal: *$ ssh remote_host*
- From Windows (using client programs):
  - PuTTY: http://www.putty.org/
  - WinSCP: https://winscp.net/eng/download.php

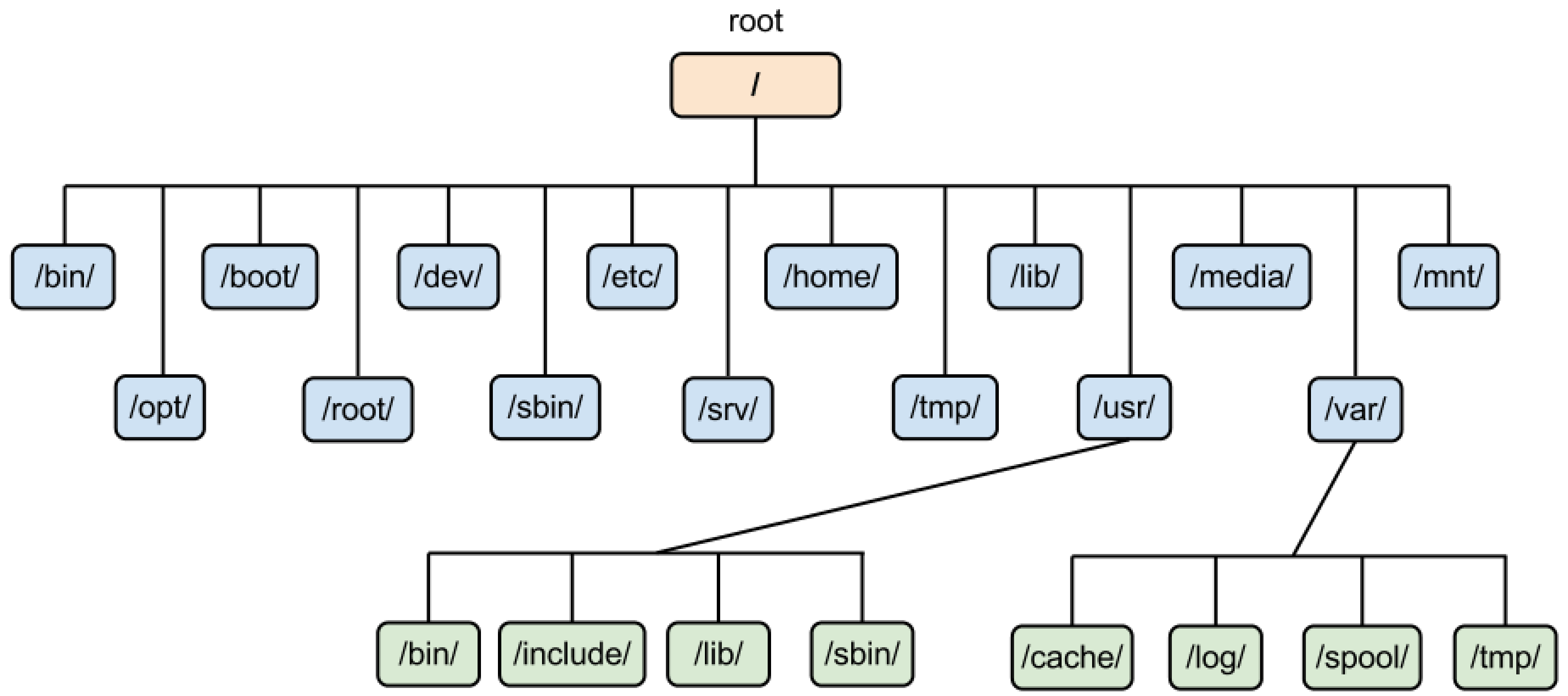
Running Linux on Windows machine - options:

- Install on a VM: Oracle VirtualBox, VMWare Workstation Player, Windows Hyper-V, etc.
- Install Linux on a USB stick: http://www.pendrivelinux.com/
- Use Cygwin: https://www.cygwin.com/

Linux file system

- A directory in Linux is similar to a “Folder” in Windows OS
- Files are organized into directories and sub-directories
- In Linux, paths begin at the root directory which is the top-level of the file system and is represented as a forward slash (/)
- Forward slash is used to separate directory and file names

Linux file system



Linux file system

Directory	Content
/bin	Common programs, shared by the system, the system administrator and the users.
/boot	The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader and is an attempt to get rid of the many different boot-loaders we know today.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows
/home	Home directories of the common users.
/initrd	(on some distributions) Information for booting. Do not remove!
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/lost+found	Every partition has a lost+found in its upper directory. Files that were saved during failures are here.
/misc	For miscellaneous purposes.
/mnt	Standard mount point for external file systems, e.g. a CD-ROM or a digital camera.
/net	Standard mount point for entire remote file systems
/opt	Typically contains extra and third party software.
/proc	A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command man proc in a terminal window. The file proc.txtdiscusses the virtual file system in detail.
/root	The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user.
/sbin	Programs for use by the system and the system administrator.
/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation etc. for all user-related programs.
/var	Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it.

Using Linux Shell commands

- **Linux Shell** - a program that takes commands from the keyboard and gives them to the operating system to perform. At the beginning, it was the only user interface available in Linux systems. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs), such as the shell.
- **Terminal** – a program called terminal emulator, that opens a window and lets you interact with the shell by typing commands
- Most commands operate like this:

command –option(s) argument(s)

Basic Commands – Navigation

- To print the name of current/working directory, use **pwd** command:

```
$ pwd
/home/username/currentdir
```

- To list all files and directories, use **ls** command

```
$ ls
$ ls /home
$ ls –l
$ ls –la ..
```

- To Change directory, use **cd** command

```
$ cd  /home/Documents
$ cd  archive
$ cd  ..
```

Basic Commands – Reading files

- To read a text file, you can use **less** command:

```
$ less readMe.txt
```

- To find the file type, use **file** command

```
$ file file_1
```

- Using **cat** command to:
  - Read files
  - Concatenate (combine) file copies into one file
  - Create new file

```
$ cat file_1
$ cat file_1 file_2 file_3
$ cat > file_4
```

Basic Commands – Manipulating files

- Copy files and directories using **cp** command

```
$ cp file_1 destinationDir
$ cp –i /user/oldPics/*.png
/home/newPics
```

- Use **mv** command to move files/directories or rename files/directories

```
$ mv file_1 destinationDir
$ mv –i file_1 file_2
```

- Removing files and directories using **rm** command

```
$ rm file_1
$ rm –r  dir1
$ rm –i file1 file2 file3
```

- Create directories using **mkdir** command

```
$ mkdir dir1
```

Basic Commands – Redirect & Pipeline

- Redirect example: redirect output from a command to file and vice-versa

```
$ ls –l > file_1
$ sort < file_1 > file_2
```

- Pipeline is used to connect two or more commands together.

```
$ ls –l | head -3
(displays the last 3 files in the directory)
```

- Filter programs are commonly used in pipelines. Filters take standard input, perform an operation upon it and send the results to standard output.

```
$ less file_1.txt | grep “keyword”
$ tail –f file_2.log | grep “error”
$ cat file_1 file_2 file_3 | pr
```

Basic Commands – Job Control

- ps – lists processes running on the system
- kill – ‘kills’ or terminates a running process

```
$ ps -l | grep “bad_program”
PID TTY STAT TIME COMMAND
2931 pts/5 SN 0:00 bad_program
$  kill 2931
```

- **bg** - putting a program to run as background process (user can still operate in the same terminal)

```
$ program1 &
```

```
$ program1
$ bg
```

Basic Commands – Helper commands

- Use **help** command to get access to the bash built-in help tool

```
$ help cp
$ help –m less
```

- Many programs support **“--help” option**

```
$ mkdir --help
```

- Most programs have a formal documentation, or manual (man page), which is accessible using **man** command

```
$ man ls
```

- Sometimes there’re multiple version of executable program installed on the server; use which to determine exact location:

```
$ which nodejs
```

Users and groups

- Linux was designed to allow more than one user to have access to the system at the same time.
- Types of Linux users:
  - **root user** – the super user that has all the rights to administer the system (UID = 0)
  - **system users** - reserved for programs and automatic processes (UID 1-499)
  - **normal users** (UID >= 500)

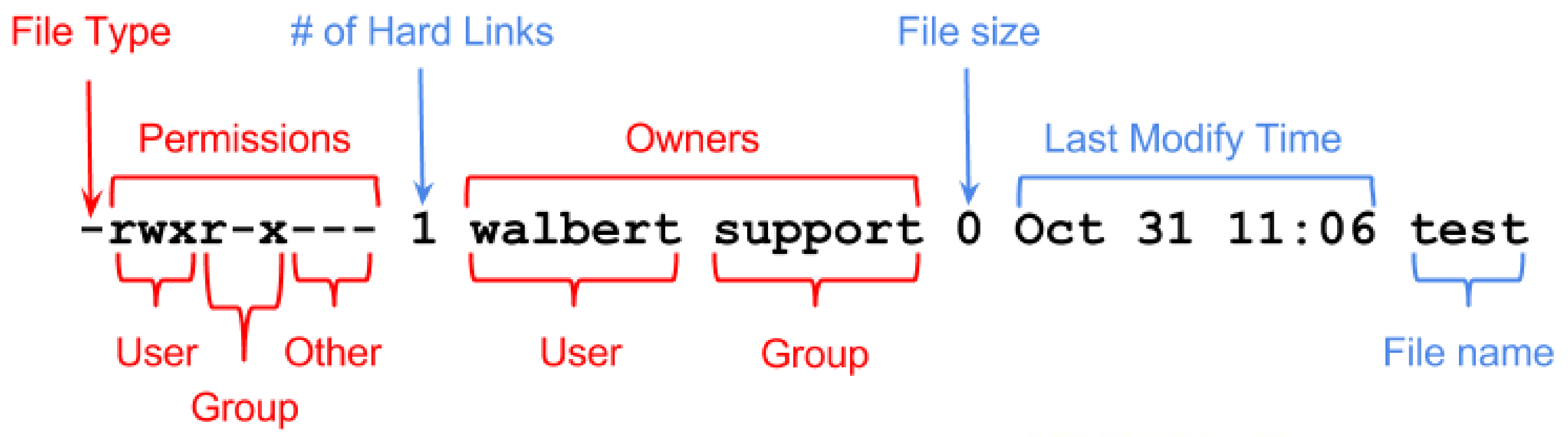
- Users are organized in groups. Each user has default/primary group, but can be member of different groups (check */etc/group* file for list of groups and associated users). When a user executes a program or creates a file, it is associated with current user’s group membership.
  - To check the login name use the command **whoami** or **echo $USER**
  - To check the groups you are a member of use the command **groups**
  - To check your user id, or group id use the command **id**

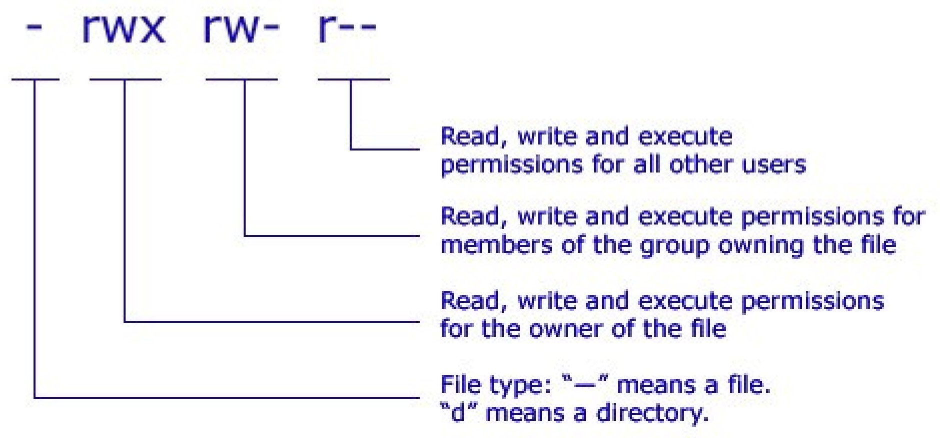
Permissions

- Permissions are the “rights” to act on a file or directory. The basic rights are read, write, and execute.
  - Read (r) - a readable permission allows the contents of the file to be viewed. A read permission on a directory allows you to list the contents of a directory.
  - Write (w) - a write permission on a file allows you to modify the contents of that file. For a directory, the write permission allows you to edit the contents of a directory (e.g. add/delete files).
  - Execute (x) - for a file, the executable permission allows you to run the file and execute a program or script. For a directory, the execute permission allows you to change to a different directory and make it your current working directory.
- Viewing file permissions

```
$ ls –l file_1.txt
rw-r--r-- 1 root root 1031 Nov 18 09:22
/etc/passwd
```

File permissions




- rwx rw- r--
Read, write and execute permissions for all other users
Read, write and execute permissions for members of the group owning the file
Read, write and execute permissions for the owner of the file
File type: "—" means a file. "d" means a directory.

Managing file permissions

Change file permissions using **chmod** command

It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them).

Here's how it works:

```
rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000
and so on...
rwx = 111 in binary = 7
rw- = 110 in binary = 6
r-x = 101 in binary = 5
r-- = 100 in binary = 4
```

```
$ ls –l file_1.txtX
-r--rw-r– 1 user group 34 Mar 9 12:29
file_1.txt
$ chmod 775 file_1.txt
-rwxrwxr-x 1 user group 34 Mar 9 12:29
file_1.txt
```

Permission string	Octal code	Meaning
`rwxrwxrwx`	777	Read, write, and execute permissions for all users.
`rwxr-xr-x`	755	Read and execute permission for all users. The file's owner also has write permission.
`rwxr-x---`	750	Read and execute permission for the owner and group. The file's owner also has write permission. Users who aren't the file's owner or members of the group have no access to the file.
`rwx------`	700	Read, write, and execute permissions for the file's owner only; all others have no access.
`rw-rw-rw-`	666	Read and write permissions for all users. No execute permissions for anybody.
`rw-rw-r--`	664	Read and write permissions for the owner and group. Read-only permission for all others.
`rw-rw----`	660	Read and write permissions for the owner and group. No world permissions.
`rw-r--r--`	644	Read and write permissions for the owner. Read-only permission for all others.
`rw-r-----`	640	Read and write permissions for the owner, and read-only permission for the group. No permission for others.
`rw-------`	600	Read and write permissions for the owner. No permission for anybody else.
`r--------`	400	Read permission for the owner. No permission for anybody else.

Managing file ownership

- By default, all files are “owned” by the user who creates them and by that user’s default group. To change the ownership of a file, use the **chown** command:

```
$ chown user2:group2 file_1.txt
```

- To change ownership of directory and all files/sub-directories in it, user **chmod –R** option:

```
$ chmod –R user2:group2
/home/user/dir1
```

- Switching users – **su** command

user1$ su user2
(will switch to user2 after prompting for user2 password)
user2$ su
(will switch to root user after prompting for root password)

Working with sudo

- Root is the super user and has the ability to do anything on a system. Therefore, in order to have protection against potential damage, **sudo** is used in place of root. Sudo allows users and groups access to commands they normally would not be able to use. Sudo will allow a user to have administration privileges without logging in as root.

- In order to provide a user with sudo ability, their name will need to be added to the **sudoers** file. This file is very important and should not be edited directly with a text editor. If the sudoers file is edited incorrectly it could result in preventing access to the system.

```
$ sudo apt-get install <package>
```

```
root$ visudo
# User privilege specification
root ALL=(ALL:ALL) ALL
user1 ALL=(ALL:ALL) ALL
user2 ALL=(ALL:ALL) ALL
```

Resources

- http://linuxcommand.org/lc3_learning_the_shell.php
- https://www.tutorialspoint.com/unix_terminal_online.php
- https://www.tecmint.com/free-online-linux-learning-guide-for-beginners/
- http://www.webminal.org/

SOFT ACADEMY