Hands-on Lab Description

CS-SYS-00001 – A Short Linux Tutorial

Category:

```
CS-SYS: Computer System
```

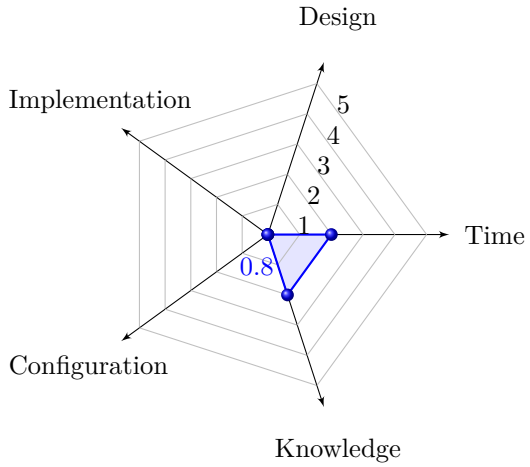
Objectives:

```
1 Learn Linux basis
2 Learn how to manage Linux based applications
3 Learn how to use Linux commandline and run Linux command
```

Estimated Lab Duration:

```
1 Expert: 20 minutes
2 Novice: 100 minutes
```

Difficulty Diagram:



Difficulty Table.

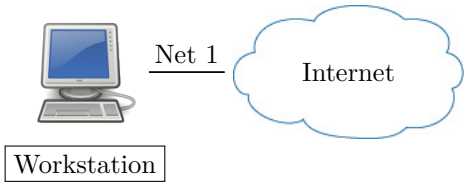
Measurements	Values (0-5)
Time	2
Design	0
Implementation	0
Configuration	0
Knowledge	2
Score (Average)	0.8

Required OS:

```
Linux: Ubuntu 18.04 LTS
```

Lab Running Environment:

```
VirtualBox https://www.virtualbox.org/ (Reference Labs: CS-SYS-00101)
```




```
1  Workstation: Linux (Ubuntu 18.04 LTS)
2  Network Setup:
   Internet: Accessible
```

Lab Preparations:

```
  Initial setup: basic Ubuntu 18.04 LTS is required for this lab
```

Task 1 Understand Linux Operating System

Linux is an operating system or a kernel. It is distributed under an open source license. Its functionality list is quite like UNIX. Linux is originally designed by *Linus Torvalds* when he was a computer science student in 1991. He used to work on the UNIX OS (proprietary software) and thought that it is needed to be improved. Compared with UNIX, there is not much difference between UNIX and Linux. Linux is a clone of UNIX, and at the core, they are essentially the same. As results, the commands used on both the operating systems are usually the same.

Linux is open-source, free to use kernel, and Linux operating system are called Distributions. It is used by programmers, organizations, profit and non-profit companies around the world to create Operating systems to suit their individual requirements. There are hundreds of Linux operating systems or Distributions available these days. Ubuntu is just one of many Linux distributions.

In Linux, files are ordered in a tree structure starting with the root directory, which is presented in Figure CS-SYS-00001.2.

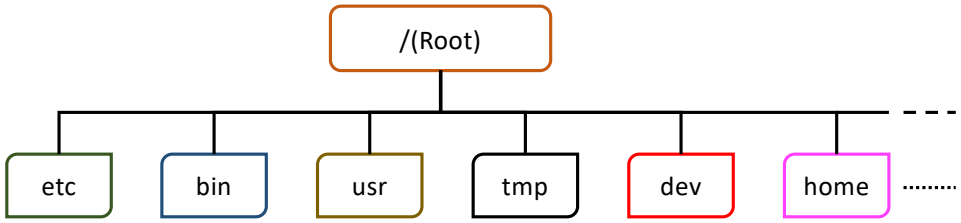


Figure CS-SYS-00001.2
Linux file system.

This root directory can be considered as the start of the file system, and it further branches out various other sub-directories. The root is denoted with a forward slash"/".

There 3 types of users in Linux:

```
1  Regular: A regular user account is created for you when you install Ubuntu on your
       system. All your files and folders are stored in /home/ which is your home
       directory. As a regular user, you do not have access to directories of other users
       .
2  Administrative (root): The root account is a superuser who can access restricted
       files, install software and has administrative privileges. Whenever you want to
       install software, make changes to system files or perform any administrative task
       on Linux; you need to log in as a root user. Otherwise, for general tasks like
       playing music and browsing the internet, you can use your regular account.
3  Service: Services such as Apache, Squid, email, etc. have their own individual
       service accounts. Having service accounts increases the security of your computer.
        Linux can allow or deny access to various resources depending on the service.
```

In Linux to escalate the user's privilege, you can issue a *sudo* command, such as:

```
$    sudo linux_command
$    sudo -i
```

The first time issuing a *sudo* command, the Linux system will ask you to input the root password. The command *sudo -i* will escalate the privilege to the root without needing to issue *sudo* before a Linux command.

It is interesting to compare between Linux and Windows operation systems for those who are familiar with Windows. Here is a list of key difference:

```
1  Linux is an open source operating system so user can change source code as per
       requirement whereas Windows OS is a commercial operating system so user does not
       have access to source code.
2  Linux is very well secure as it is easy to detect bugs and fix whereas Windows has a
       huge user base, so it becomes a target of hackers to attack windows system.
3  Linux runs faster even with older hardware whereas windows are slower compared to
       Linux.
4  Linux peripherals like hard drives, CD-ROMs, printers are considered files whereas
       Windows, hard drives, CD-ROMs, printers are considered as devices
5  Linux files are ordered in a tree structure starting with the root directory whereas
       in Windows, files are stored in folders on different data drives like C: D: E:
6  In Linux you can have 2 files with the same name in the same directory while in
       Windows, you cannot have 2 files with the same name in the same folder.
7  In Linux you would find the system and program files in different directories whereas
        in Windows, system and program files are usually saved in C: drive.
```

Task 2 Understand Linux Command Line Interface (CLI)

To work in Linux, you can either use a *Terminal* by using the Command Line Interface - CLI, or Graphical User Interface - GUI provided by various applications. CLI is a frequently used user interface that provide fast access, esp., when running simple commands or scripts to access Linux system components, and it can derive prompt feedback (usually short) to users. There are two ways to launch a terminal:

```
1  Go to the Dash and type terminal
2  Press CTRL + Alt + T to launch the Terminal
```

At the prompt of a terminal window, it may show as follows (an example):

```
  dhuang8@ubuntu:/home/dhuang8/$
```

It can be explained as follows:

```
1  The first part of this line is the name of the user (dhuang8, ubuntu, ...)

3  The second part is the computer name or the host name. The hostname helps identify a
       computer over the network. In a server environment, host-name becomes important.

5  The ':' is a simple separator

7  /home/dhuang8/$ presents the current working directory. Note that if you escalate
       your privilege to root, the $ sign will be changed to ~ #
```

Special Notice: In the command line of Linux, the user mode should be started with a "$" sign. In order to execute the command in an escalated *root* privilege, you can either use ***sudo linux_command*** to escalate the privilege temporarily, or use ***sudo -i*** to escalate the privilege permanently. In the root privilege, you should see a "#" sign starting at the command line prompt. Without a special notice, in the lab description, we do not differentiate the difference between "$" and "#". You just simply add a *sudo* in front of your command if the Linux system requires the root privilege to execute the it.

Several frequently-used directories changing CLI commands are presented as follows:

```
$    cd              % go to the home directory such as /home/dhuang8
$    cd /          % go to the root directory
$    cd /usr/sbin  % go to a specific folder such as /usr/sbin
$    cd ..            % go to the parent directory
```

Task 3 Basic Linux Commands with Examples

In this task, we present several most frequently used Linux CLI commands. To learn details for each command and options, you can simply issue its *man* page as follows:

```
$    man linux_command
```

List files and directories:

```
$    pwd     % show current residing directory
$    ls      % show files in the current directory
$    ls -R  % show all files not only in current directory but also subdirectories
$    ls -l   % give detailed information of files
$    ls -a   % view hidden files, a hidden file is a file_name started with '.",
       e.g., .file_name)
$    ls -d */  % list directories for the current directory
```

File and directory operations:

```
$    cat  file_name     % display a text file's content
$    rm   file_name      % delete a file
$    mv   file1_name file2_name  % change file1_name to file2_name, also, we can use
       it to change directory name
$    mkdir folder_name  % create a new foler with the name folder_name
$    rmdir folder_name  % delete the foler folder_name or you can use rm with -R option
```

Show hardware configuration of the Linux system:

```
$    lshw    % extract detailed information on the hardware configuration
$    lshw -short % have device tree output showing hardware paths- a short summarylshw
       -businfo
$    lshw -businfo % fetch SCSI, USB, IDE and PCI device info
$    lshw -html % display information in HTML format
$    lshw -xml % display information in XML format
$    lshw -sanitize % no sensitive information
```

Show command-line history:

```
$    history                  % show the whole history of issued commands in CLI
$    history | more       % show history page by page
$    history | grep linux_command % show specific Linux command in history
$    clear                % clear current CLI window
```

In GUI, use the following command to change display resolution:

```
$    xrandr --current  % show existing available display resolution
$    xrandr --size 4:3 % change the ratio of the display
$    xrandr --size 1280x800 % change the display resolution based on an existing
       availabe setup
```

For CLI only, you need to modify the grub file to change the resolution setting and restart the VM.

```
$    vim /etc/default/grub  % open the grub file and change the set up, e.g.,
       GRUB_GFXPAYLOAD_LINUX=1024x768. Avaliable resolutions such as 640x480 800x600
       1024x768 1280x960 1600x1200 1920x1200
$    sudo update-grub
$    reboot
```

Task 4 Understanding file permissions

Linux is a clone of UNIX, the multi-user operating system which can be accessed by many users simultaneously. For effective security, Linux divides authorization into 2 levels.

```
1  Ownership
2  Permission
```

Every file and directory on your Linux system is assigned 3 types of owner, given below.

- *User/Owner* : A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.
- *Group*: A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.
- *Other*: Any other user who has access to a file. This person has neither created the file, nor he belongs to a user group who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

Every file and directory in your Linux system has following 3 permissions defined for all the 3 owners discussed above.

- *Read*: This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- *Write*: The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.

- *Execute*: In Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

You can use *ls -l* command to display file attributes, such as

```
$    ls -l
```

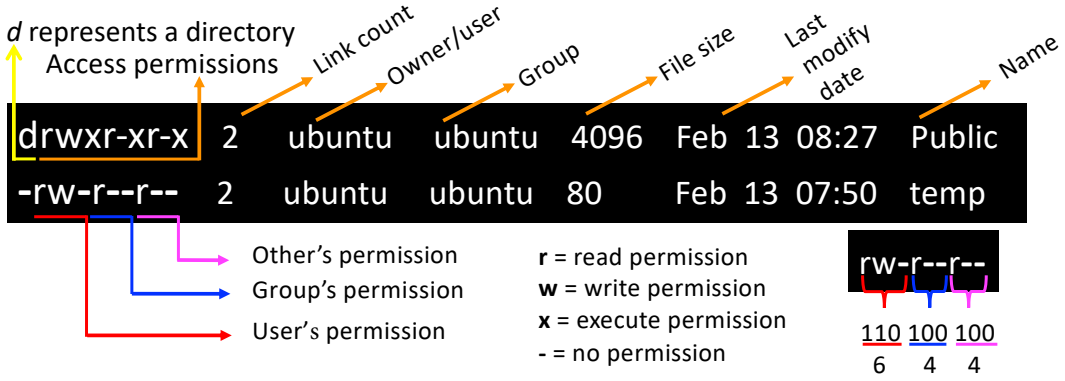


Figure CS-SYS-00001.3
Linux file and directory permission.

The results can be viewed as an output example shown in Figure CS-SYS-00001.3. In this example, the file *temp* is a text file with permission 644 represented as *rw-r–r–*. The individual value 6, 4, and 4 can be represented as binary values 110, 100, and 100, respectively. Where the value 1 means *permit* and 0 means *do not permit*. Three positions represent *read write execute* in sequence, and three decimal digit value represents permissions assigned for the *user*, *group*, and other, respectively. In this example, 644 represents the user can *read*, *write*, but not *execute* the file *temp*. Both the *group* and *other* can *read* but not be able to *write* and *execute* the file.

In Linux, a *shell* is special user program which provide an interface to user to use operating system services. Common shells in Linux include, *bash*, *csh*, *ksh*, etc. A shell accepts human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.

Usually shells are interactive that mean, they accept command as input from users and execute them. However, some time we want to execute a bunch of commands routinely, so we have type in all commands each time in terminal. As shell can also take commands as input from file we can write these commands in a file and can execute them in shell to avoid this repetitive work. These files are called Shell Scripts or Shell Programs. Shell scripts are similar to the batch file in MS-DOS. Each shell script is usually saved with *.sh* file extension, e.g., *myscript.sh*. In order, make the script executable, we then need to change a text file that include a sequence of commands to an executable file, in which an executable file is listed with green color in Linux terminal. In the above example, the file *temp* is not an executable file. To make change, we need to change the third bit from 0 to 1 by using the follow command:

```
$    chmod temp 755
$    ls -l temp
```

The list results of changed permission on file *temp* is presented in Figure CS-SYS-00001.4. After changing the permission, the file *temp* is then executable.

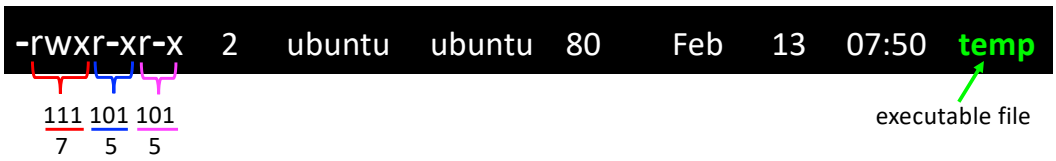


Figure CS-SYS-00001.4
Change Linux file's permission.

Task 5 Install a Software in Linux

Ubuntu's package management system is derived from the same system used by the Debian GNU/Linux distribution. The package files contain all of the necessary files, meta-data, and instructions to implement a particular functionality or software application on your Ubuntu computer. Debian package files typically have the extension '*.deb*', and usually exist in repositories which are collections of packages found on various media, such as CD-ROM discs, or online. Packages are normally in a pre-compiled binary format; thus, installation is quick, and requires no compiling of software.

There are mainly installation tools to install software in addition to compiling source code directly in Ubuntu:

- *dpkg* is a package manager for Debian-based systems. It can install, remove, and build packages, but unlike other package management systems, it cannot automatically download and install packages or their dependencies.
- The *apt* command is a powerful command-line tool, which works with Ubuntu's Advanced Packaging Tool (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system. Most of the following examples are presented using the *apt* command.
- The *aptitude* command is best suited for use in a non-graphical terminal environment to ensure proper functioning of the command keys. It can give you a menu-driven, text-based front-end to the *apt* system.

In Linux, installation files are distributed as packages. But the package contains only the program itself. Any dependent components will have to be installed separately which are usually available as packages themselves. You can use the *apt* commands to install or remove a package. Note that *apt* was introduced into the Ubuntu 16.04 LTS, and thus we can usually use both commands ***apt*** and ***apt-get***, and we do not differentiate them in the presented examples. Before installing a software, we usually first update all the installed packages in our system using command:

```
$    sudo apt-get update
```

This command updates the list of available packages and their versions, but it does not install or upgrade any packages. Another command is:

```
$    sudo apt-get upgrade
```

This command installs newer versions of the packages that we have. After updating the lists, the package manager knows about available updates for the software you have installed. This is why you first want to update.

Configuration of the Advanced Packaging Tool (APT) system repositories is stored in the */etc/apt/-sources.list* file and the */etc/apt/sources.list.d* directory. An example of this file is referenced here, along with information on adding or removing repository references from the file. You may edit the file to enable repositories or disable them by simply commenting out the appropriate line.

In addition to the officially supported package repositories available for Ubuntu, there exist additional

community-maintained repositories, which add thousands more packages for potential installation. Two of the most popular are the Universe and Multiverse repositories. These repositories are not officially supported by Ubuntu, but because they are maintained by the community they generally provide packages which are safe for use with your Ubuntu computer. Note that

```
Packages in the Multiverse repository often have licensing issues that prevent them
    from being distributed with a free operating system, and they may be illegal in
    your locality.
Be advised that neither the Universe or Multiverse repositories contain officially
    supported packages. In particular, there may not be security updates for these
    packages.
```

By default, the Universe and Multiverse repositories are enabled but if you would like to disable them edit */etc/apt/sources.list* and comment the following lines:

```
  deb http://archive.ubuntu.com/ubuntu bionic universe multiverse
  deb-src http://archive.ubuntu.com/ubuntu bionic universe multiverse
  deb http://us.archive.ubuntu.com/ubuntu/ bionic universe
  deb-src http://us.archive.ubuntu.com/ubuntu/ bionic universe
  deb http://us.archive.ubuntu.com/ubuntu/ bionic-updates universe
  deb-src http://us.archive.ubuntu.com/ubuntu/ bionic-updates universe
  deb http://us.archive.ubuntu.com/ubuntu/ bionic multiverse
  deb-src http://us.archive.ubuntu.com/ubuntu/ bionic multiverse
  deb http://us.archive.ubuntu.com/ubuntu/ bionic-updates multiverse
  deb-src http://us.archive.ubuntu.com/ubuntu/ bionic-updates multiverse
  deb http://security.ubuntu.com/ubuntu bionic-security universe
  deb-src http://security.ubuntu.com/ubuntu bionic-security universe
  deb http://security.ubuntu.com/ubuntu bionic-security multiverse
  deb-src http://security.ubuntu.com/ubuntu bionic-security multiverse
```

To install a software package:

```
$    sudo apt-get install package_name
```

Now, you can use the following command to list all installed software packages:

```
$    dpkg --list   % a full list
$    dpkg --list | more  % display listed packages page by page
$    dpkg --list | grep package_name % list specific package_name
$    sudo apt-get remove package_name % remove package_name
$    sudo apt-get --purge remove package_name % remove package_name and delete all
       related configuration files
$    sudo apt-get autoremove % remove unused packages
```

Related Information and Resource

```
Linux commands for beginners:
    https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
Linux Tutorials: https://www.linux.org/forums/linux-beginner-tutorials.123/
The Linux Command Line: http://linuxcommand.org/tlcl.php
```