Linux Command Line

(Cover All Essential Linux Commands)

A BEGINNER'S GUIDE

Ray Yao

Linux Command Line

(Cover All Essential Linux Commands)

A Beginner's Guide

By Ray Yao

Ray Yao

About the Author

Ray Yao:

Certified PHP engineer by Zend, USA
Certified JAVA programmer by Sun, USA
Certified SCWCD developer by Oracle, USA
Certified A+ professional by CompTIA, USA

Certified ASP. NET expert by Microsoft, USA Certified MCP professional by Microsoft, USA Certified TECHNOLOGY specialist by Microsoft, USA Certified NETWORK+ professional by CompTIA, USA

Preface

This book is a beginner's guide for fast learning Linux commands which are frequently used by Linux administrators or beginners. The book covers all essential Linux commands as well as their operations, examples and explanations. It also includes Linux Helping commands, symbols, shortcut keys, run levels and Vi commands. From this book, you can easily learn:
How to run all essential Linux commands.
How to copy, move, and delete files and directories.
How to create, remove, and manage users and groups.
How to access Linux server, and use SSH commands.

How to operate the run levels and change the run levels How to navigate at the command line by helping commands.

How to compare files, find out a file, manipulate file contents How to start a job, stop a job and schedule a job.

How to manage permissions, ownership of files, directories How to connect across network, communicate with network.

How to transfer files over network, send network messages And much more skill……
There is a long table containing all common Linux commands in this book, which can give you a great help in your job or study. You can learn all essential Linux commands quickly.

Table of Contents

Chapter 1
Introduction to Linux

About Linux Operating System

Linux is a Unix-like and mostly POSIX-compliant computer operating system assembled under the model of free and open-source software development and distribution.
Linux usually works as a server, because of its stability and security's feature.
Linux programs are extremely advantageous:

> Linux programs are free, you'll see, most of Linux programs are.
> They are frequently updated and for Zero charge!
> Some of them are better than those in windows. And other doesn't even exist in windows!

If Linux is free and almost all their softwares are free, it is for a reason; to understand we have to go back to 1984.

1984

So we are back in 1984, computer science was not very developed. Microsoft has just launched its first os: MS-DOS, but this one is far away from being done.

But, was MS-DOS the only one then?

No! There was other operating system but less known by the public.
The one that was called the best was "Unix". It was a lot powerful than MS-DOS but a lot complicated, what explains that only the professionals could use it.
Graphically UNIX looked a lot like MS-DOS they were both seen like a black screen with some white text in it. We must say that computers back then were not capable of doing better.

GNU Project

It is just in 1984, that Richard Stallman created GNU project.
The GNU Project is free software, mass collaboration project, announced on 27 September 1983, by Richard Stallman at MIT. Its aim is to give computer users freedom and control in their use of their computers and computing devices, by collaboratively developing and providing software that is based on the following

freedom rights: users are free to run the software, share it (copy, distribute), study it and modify it. GNU software guarantees these freedom-rights legally (via its license), and is therefore free software; the use of the word "free" always being taken to refer to freedom.
Richard Stallman was a researcher in Artificial intelligence in MIT. He wanted to create an operating system based on UNIX (the commands still the same).
But why would he create a copy of “UNIX”?
Because UNIX was not free and it was getting more expensive! Richard Stallman wanted to react by creating a free alternative: the project GNU was born.

GNU is an open operating system

GNU should not only be a free OS; it also had to be "open"
What is the difference?
A free program is a program where you can have the source code, that is to say, the "batch recipe." In contrast, Windows is a proprietary OS whose source code is stored by Microsoft. Imagine it's like Coca-Cola: nobody knows the recipe (there are many people who try to imitate it, but hey ...). So we cannot change it or see how it works inside.
An open program is mostly a free program, it is also a program that has the right to copy, modify, redistribute. It's a real ideology in computer science: people think it is better to give the source code of the programs that we create because it allows knowledge sharing and helps the computer to evolve faster. The slogan of the Free World might be: "Unity is strength."
They say whenever the program is "open source" because its source code is open; everyone can see it. There are some slight differences between "open source" program and a "free" program, but we will not go into details here.

Linus Torvalds is doing his hobby

In 1991, Linus Torvalds, a student at the University of Helsinki (Finland), began creating his free own operating system. This system became known as Linux, referring to the name of its creator (Linux is a contraction of Linus and UNIX).

Linus Torvalds, creator of Linux

What relationship with GNU? Well it turns out that these two projects were complementary: while Richard Stallman created the basic programs (program

file copy, delete, file, text editor), Linus had embarked on the creation of the "heart" an operating system kernel.
The GNU (free programs) and Linux (OS kernel) project merged to create GNU / Linux.
Theoretically, we should talk about GNU / Linux. But it is a bit difficult to write and pronounce, and by abuse of language, we often say just "Linux". This is why I continue to speak of "Linux" in the rest of the book, even though the politically correct name is "GNU / Linux" because it is the merger of two complementary projects.

Original operating systems

You should now have a better idea of the origin of the three major operating systems that exist today: Mac OS, Linux and Windows.
Thus, Mac OS and Linux are both based on UNIX, the ancestor of operating systems, while Windows, from MS-DOS is a separate branch. Overall, this is all you need to remember.
It is said that Mac OS and Linux are based on UNIX because they have "copied" its operation. It's not pejorative, it’s quite the opposite: it's been an honor to UNIX.
Linux programs do not use all the same source code as UNIX (it was also the owner, so private). They have been completely rewritten but work the same way.
If I told you all this is because I believe that knowing the origin of Linux is important. This will help you understand many things thereafter.

Linux distributions

A Linux distribution (often called distro for short) is an operating system made as a collection of software based around the Linux kernel and often around a package management system. The most well known distributions are RedHat, SUSE, Debian, Mandriva, Slackware and Ubuntu. You can find much different software and there are hundreds of different ways to install it.
To make life easier for users and allow them to make a choice, different Linux distributions were created. This is a concept that does not really exist in Windows. It's like the difference between Windows 7 Home and Windows 7 Professional, but it goes much further than that.

Here's what can differ from one distribution to another:

Installation: it can be greatly simplified as very complicated;

Installing management programs: If it is done well and centralized, it

can make the installation of new software simpler than Windows, as discussed further!
The preinstalled programs on your computer (e.g. Windows is bundled with Internet Explorer and Windows Media Player).

In fact, distribution is somehow packing Linux. The heart itself remains the same for all distributions.
Whichever distribution you install, you get a Linux compatible with others. Some distributions are just more or less easy to handle. :-)

Various existing distributions

There are many different Linux distributions. Hard to choose , you will say: indeed, the first time it is unclear what to choose ... especially since all are free! Do not worry; I'll help you make your choice.
I will not list all existing distributions, but here at least the main ones:
Slackware: one of the oldest Linux distributions. It still exists today!
Mandriva: published by a French company, it is simple to use;
RedHat: published by an American company, this distribution is known and widespread, especially on servers;
SUSE: Novell published by the company;
Debian: Debian distribution alone which is managed by independent developers rather than a business. This is one of the most popular distributions.
As I have said, whatever the distro (short for distribution) you choose, you will have a Linux. Basically, "just" a screen on first boot and various software preinstalled (I'm simplifying a bit much, but the idea is there).

Summary

Windows, Mac OS and Linux are the most popular operating systems.
Linux usually works as a server, because of its stability and security’s feature.
Linux has the distinction of being free, that is to say that its source code (the manufacturing recipe) is open: anyone can view it. In contrast, the source code that was used to design Windows and Mac OS is closed; we say that these are proprietary operating systems.
There are many variants of Linux, called distributions.

Chapter 2
Enter First Commands

Dear friends, the big day has finally arrived! You will get the chance to write your first command in console!
Okay, not too stressed?

I assure you, we will start with simple things to become familiar with the console. We'll really see the ABC, the basic survival guide of kits.

What is the Linux shell?

A Linux shell is a command-line interpreter or shell that provides a traditional user interface for the Linux operating system and for Linux-like systems.
The shell understands a plenty of shell commands and its option which change their action. The typical syntax of sell command looks like this: **command –option argument**
or
command parameter
(Usually –option argument means parameter)
Command such as: ls, cat, pwd, cp, mv, date……
Parameter such as: -a, -l, -s, --all, --help……

Example:
ls –a
Explanation:
ls is a command meaning list the contents in current directory -a is a parameter meaning “all”.
Result: list all contents in current directory.

Shell commands can be run at a prompt in text interface mode or in a shell terminal window.
The command prompts with shell command show something like this:
username@hostname: ~ $ command parameter
root@hostname:~# command parameter
user> command parameter

Explanation:

What you see here is called the command prompt. It is a message that prompts you to enter a command by giving you at the same time a lot of information. This command prompt is displayed before each command you type.

“username@hostname:~$ **”**is a command prompt.
“root@hostname:~#” is a command prompt too.
“user>” is a command prompt as well.

About the “username@hostname:~$ ls”

Example:
user2014@user-linux:~$ ls

Explanation:

user2014: the first element is your nickname. This is the user name under which you are logged and you. Indeed, remember: you can create multiple user accounts on Linux. It is generally advisable to generate a person who’s likely to use the computer.
@: This symbol indicates nothing special. It's the symbol "at"
user-linux: that's the name of the computer on which you are working. In my case it is called user-linux, but I could give it any name during installation.
: Again, this symbol does not mean anything special, it is a separator.
~: That's the folder where you currently are. You can navigate from folder to folder in the console and it is very useful that you always be reminded where you are before each command.
For information, the symbol ~ means that you are in your home directory, so-called "home" under Linux; this is equivalent to the "My Documents" folder on Windows. We will study in detail the operation of the files in Linux in the next chapter.
$: That symbol is very important; it shows your authorization level on the machine.
ls: ls is a command, show the contents of current directory.

More detail about $

$: Means you are currently using a user "normal" account with limited rights (he cannot change the most important system files). My account user2014 is a normal account with limited rights; As you can see, once we speak the same language as the command prompt, you understand what it means!
"Welcome, you are user2014 at user-linux machine. You are currently in your home directory and have limited user rights. You are using a command “ls” to list the contents in current directory."

About the “root@hostname:~#whoami”

Example:
root@user-linux:~# whoami

Explanation:
root: means user work as a super user **@**: This symbol indicates nothing special. It's the symbol "at"
user-linux: that's the name of the computer on which you are working. In my case it is called user-linux, but I could give it any name during installation.
: Again, this symbol does not mean anything special, it is a separator.
~: That's the folder where you currently are. You can navigate from folder to folder in the console and it is very useful that you always be reminded where you are before each command.
For information, the symbol ~ means that you are in your home directory, so-called "home" under Linux; this is equivalent to the "My Documents" folder on Windows. We will study in detail the operation of the files in Linux in the next chapter.
#: means you are working in super user mode **Whoami:** whoami is a command, show the current user name **More detail about #**

#: Means you are in super user mode, that is to say that you are connected under the pseudonym "root." The root is the master who has the right to do everything on his computer user (even to destroy it!). We'll see how root in more detail later; yet we remain in a limited user account, so we do not risk doing bad things.
"Welcome, you are super user at user-linux machine. You are currently in your

home directory and have super user rights. You are using a command “whoami” to show the current user name."

About “user>pwd”

Example:
user> pwd

Explanation:
user> : is a customized command prompt.
pwd: pwd is a command, print working directory.
You can customize the command prompt like user>. Of course after you are familiar with Linux command programming, you will be able to customize the shell prompt.
At this moment, the Linux machine will say hallo to you: "Welcome, you are a user at user-linux machine. You are currently in your home directory and have limited user rights. You are using the “pwd” command to print working directory."

As a bit of everything on Linux, the command prompt is fully configurable. You can shorten it if you find it is too long, or lengthen it if it does not give enough information. You can theoretically put really everything you want in the prompt, such as the current time.

Working in the console by typing commands, the latter being numerous, you can never know all of them ... and it is not the goal: the goal is that you know by heart to serve the most "common" ones and for the less common you are able to learn to use them by reading their manual.

About Linux commands

The typical syntax of sell command looks like this: **command –option argument**
or
command parameter

Let’s see some example of commands and parameters: **Example:**
Type “**date**” and press the Enter key.
user2014 @ user-linux: ~ $ date
Monday, September 20, 2010, 3:39:51 p.m. (UTC-0200) Explanation:
The first line contains the command prompt followed command I typed. In here, “**date**” is a command.
The second line is the computer response to this command: we asked about the date and time!

About Parameters

Parameters are options that are written after the command. The command and parameters are separated by a space, like this: **user2014 @ user-linux: ~ $ command parameters** The parameters themselves can contain spaces, letters, numbers ... a bit of everything, really. There is no real rule on how the settings, but fortunately programmers have adopted a sort of "agreement" so that we can recognize the different types of parameters.

Short parameters (one letter)

The most common parameters are constituted by a single letter preceded by a dash.
For instance:
-d
-l
-a
If we have to give several parameters, you can do it like this: -d -a -U -h
Or shorter:
-daUh

BEWARE! For short parameters: a parameter in different command has different meanings.
Example:
ls -t (-t means “list by timestamps”.)
eject -t (-t means “tray close”.)
chfn -f (-f means ”change information by finger name”) cut -f (-f means “cut

text by a field number")
ps -f (-f means "show process status in full information") BEWARE! Parameter is case sensitive (upper / lower case). If you write -u, this has generally not the same sense as –U.
Does a test with the **ls** command, and write it the parameter "-a" (lower case), -a means "all": **Example:**
user2014@user-linux: ~ $ ls -a.
.gconfd .mozilla-thunderbird .. gimp-2.2 .nautilus .bash_history .gksu.lock .profile .bash_logout .gnome .recently-used .bashrc .gnome2 .recently-used.xbel .config .gnome2_private .ssh Desktop .gstreamer- .sudo_as_admin_successful .dmrc .gtkrc 0.10-1.2-gnome2 .themes .esd_auth .ICEauthority .thumbnails .evolution .icons .Trash Examples .lesshst tutorials .face .local .update-manager-core .fontconfig .macromedia .update-notifier .gaim. metacity .Xauthority .gconf .mozilla .xsession-errors This displays all files of current directory, even hidden files.
A "cookie" is a Linux file that begins with a period. Normally, if you're in your home directory, you should have a good bunch of hidden files. These are usually configuration files program.

Long parameters (severalletters) The parameters consist of several letters are preceded by two dashes, like this: --long parameter
For instance: --all
--all is a long parameter, meaning all contents or all things.
For instance: --version
--version is a long parameter, meaning the version of the command For instance: --help
--help is a long parameter, meaning get help for current command.
If you want to put several feature parameters, it will add a space between each one: Command --long parametre1 --long parametre2
One can also combine the long and short parameters in control parameters: Command -daUh --All
Sometimes there are two possible entries for a control parameter: a short version and a long version. This will let you choose whichever you prefer one or the other.
Note that this is the command that decides the parameters it accepts: sometimes some do not offer a choice between a short version and a long.
Let's test this on the **ls** command with the --all parameter, which means "everything": **Example:**

user2014@user-linux: ~ $ ls --all.
.gconfd .mozilla-thunderbird .. gimp-2.2 .nautilus .bash_history .gksu.lock .profile .bash_logout .gnome .recently-used .bashrc .gnome2 .recently-used.xbel .config .gnome2_private .ssh Desktop .gstreamer- .sudo_as_admin_successful .dmrc .gtkrc 0.10-1.2-gnome2 .themes .esd_auth .ICEauthority .thumbnails .evolution .icons .Trash Examples .lesshst tutorials .face .local .update-manager-core .fontconfig .macromedia .update-notifier .gaim. metacity .Xauthority .gconf .mozilla .xsession-errors As you can see, is a synonym for --all -a. This illustrates what I said a moment ago, which shows that sometimes a command offers two ways to use a parameter: a short and a long.

******

Commands and Parameters Examples

OK! Let's have a further look about the commands and their parameters.

su –l : switch user
Example:
user> su –l
(su: going to login as the root super user,
-l: is a parameter meaning "login")

ls –l : list long contents
Example:
user>ls –l
(ls: shows the contents of current directory.
–l: is a parameter meaning "long list include access permissions, ownership and date & time.") **ls –a : list all contents**
Example:
user>ls –a
(ls: shows the contents of current directory.
–a: is a parameter meaning "all contents" including hidden files.) **rm –ri : remove a directory and its contents**
Example:
user>rm –ri NonEmptyDir
(rm: removes a file or a directory.

–ri: is a parameter meaning remove a non-empty directory and its contents. NonEmptyDir is a directory name.)

w –s : show current process for each user
Example:
user>w -s
(w: shows the shell working processes.
–s: is a parameter meaning “summary ”)

usermod –l : modify an existing user account.
Example:
user>usermod –l oldname newname
(usermod: modify an existing user account.
–l: is a parameter meaning “login name change”)

******

What is Virtual Console?
Virtual Console means an interface where the input device and the output device designed to enable you to interact with your system.
Linux has 7 virtual consoles, you can switch them using Ctrl+Alt+F1through F7.
Ctrl+Alt+F1~F6: switch virtual console 1~ virtual console 6
Ctrl+Alt+F7: enter graphical desktop, which is default virtual console.

Summary:

1.When user is a normal user, use:

username@hostname:~$ command parameter

2.When user is a super user, use:

root@hostname:~# command parameter

3.When the shell prompt has been customized, use: **User> command parameter**

4.Linux command: ls, pwd, su, whoami, loginname, rm, exit…

5.Command parameter: -a, -ri, -l,--all, --help…
6. Virtual Console: let you have several interface shell sessions active at the same time.

Chapter 3
Super User Commands

su: switch a normal user into a root super user
loginname: shows the login name
exit: exit the shell.
whoami: shows the current user name
hostname: shows the current host name
sudo: allows a user with proper permissions to execute a command as another user, such as the superuser **su: switch a normal user into a root super user Example:**
user2014@user-linux:~$ su –l
(su: switch a normal user into a root super user.
–l: enter root password and login.
Note: After login as a super user, the $ will become #.) **loginname: shows the login name**
Example:
root@user-linux: ~ # loginname
(loginname: shows the login name, the output is “root”.
Note: After login as a super user, the $ becomes #.) **exit: exit the shell**
Example:
root@user-linux:~# exit (exit: exit the shell. In here: exit the super user mode, and enter the normal user mode.
Note: After exit super user, the # will become $.) **whoami: shows the current user name**
Example:
user2014@user-linux: ~ $ whoami
(whoami: shows the current user name, the output is “user2014”) **hostname: shows the current host name**
Example:
user2014@user-linux: ~ $ hostname
(hostname: shows the current host name, the output is “user-linux”) **sudo: allows a user with proper permissions to execute a command as another user, such as the superuser Example:**
root> sudo -u andy ls *home*mydir (list the contents of the *home*mydir directory as user andy.
-u: specify a user) root> sudo –v

(-v: refresh the authentication timeout, the next sudo command will not require a password.) root> sudo -k
(-k: expire the authentication timeout, the next sudo command will require a password.)

Summary:

su: switch a normal user into a root super user loginname: shows the login name
exit: exit the shell.
whoami: shows the current user name
hostname: shows the current host name
sudo: allows a user with proper permissions to execute a command as another user, such as the superuser

Chapter 4
Navigating At Commands

pwd: print working directory.
cd dir: change directory.
cd~ change directory to home directory.
cd**..** change directory to a parental directory.
type: determine a command type.

pwd: print working directory.

Example: user> pwd (pwd: print working directory, the output is your current working directory.) **cd dir: change directory Example:** user> cd mydir (cd: change directory to mydir, the output is mydir.) **cd~ change directory to home directory.**

Example: user> cd **~**
(cd~ change directory to home directory, the output is home directory.) **cd.. change directory to a parental directory.**

Example:

user> cd **..**
(cd**..** change directory to a parental directory, the output is a parental directory.) **type: determine a command type Example:** user> type pwd (output: pwd is a shell builtin.) **Summary:** pwd: print working directory.
cd: change directory.
cd~ change directory to home directory.
cd**..** change directory to a parental directory.
type: determine a command type.

Chapter 5
File Operation Commands

cp: copy a file
mv: move a file
mv: rename a file
rm: remove a file
rm –ri: remove a non-empty directory vi: open vi editor and edit a file find: look for a file
wc: show word count of a file
file: estimate the type of a file ln: create a link between two files ln -s: create a symbolic link to a file readlink: show the target of a symbolic link lpr: sent a file to printer
lpq: display the print queue.

cp: copy a file

Example:
user> cp myfile /dir1
(cp: copy myfile to /dir1directory.) **mv: move a file**

Example:
user> mv myfile /dir2
(mv: move myfile to dir2 directory.) **mv: rename a file**

Example:
user> mv myfile1 myfile2
(mv: rename myfiel1 as myfile2.) **rm: remove a file**

Example:
user> rm myfile
(rm: remove myfile.)

rm –ri: remove a non-empty directory Example:
user> rm –ri NonEmptyDir
(rm: remove a directory named NonEmptyDir.
-ri: remove a directory containing contents.) **vi: open vi editor and edit a file**

Example:
user> vi myfile.txt
(vi: open vi editor and edit myfile.txt.) **find: look for a file**

Example:
user> find directory –type f –name myfile.txt -print (find: look for a file.
-type f: specify a file
-name: specify a filename
-print: print)

wc: show word count of a file Example:
user> wc myfile.txt
(wc: show word count of myfile.txt.) **file: estimate the type of a file Example:**
user> file myfile.txt
(file: estimate the type of myfile.txt.) **ln: create a link to a file**

Example:
user> ln dir1/file1.txt dir2/file2.txt (ln: create a link between file1 and file2) **ln -s: create a symbolic link between two files Example:**
user> ln –s dir1/file1.txt dir2/file2.txt (ln-s: create a symbolic link between file1 and file2) (-s: a symbolic link allows a given file to appear in many places or under many names at once. For instance, symbolic links can link to directories.) **readlink: show the target of a symbolic link Example:**
user> readlink dir2/file2.txt (the output : dir1/file1.txt) **lpr: sent a file to printer**

Example:
user> lpr myfile.txt
(lpr: sent myfile.txt to printer.) **lpq: display the print queue.**

Example:
user> lpq
(lpq: display the print queue.) **Summary:**
cp: copy a file
mv: move a file
mv: rename a file
rm: remove a file
rm –ri: remove a non-empty directory vi: open vi editor and edit a file find: look for a file
wc: show word count of a file
file: estimate the type of a file ln: create a link between two files ln -s: create a

symbolic link to a file readlink: show the target of a symbolic link lpr: sent a file to printer
lpq: display the print queue.

Chapter 6
Viewing File Commands

cat: show contents of a file
cat | less: display a file contents page by page cat | more: display a file contents screen by screen head: show the front part contents of a file tail: show the last part contents of a file aspell: spelling check for a file
cut: show the specified column of a text file paste: merge two files contents and display sort: show lines of text sorted alphabetically stat: display the attributes of a file or directory wc: display word count in a file
file: test the file type
touch: create a file or change file timestamp nl: show numbers for each line of a file vi: edit or create a text file with vi editor tr: transform text in a file
tee: print standard output, write to a file **cat: show contents of a file**

Example:
user> cat myfile.txt
(cat: show contents of myfile.txt.)

cat | less: display a file contents page by page Example:
user> cat myfile.txt | less
(cat: show contents of myfile.txt.
| : redirect the output to another command less: display myfile.txt contents page by page) **cat | more: display a file contents screen by screen Example:**
user> cat myfile.txt | more
(cat: show contents of myfile.txt.
| : redirect the output to another command more: display myfile.txt contents screen by screen) **head: show the front part contents of a file Example:**
user> head myfile.txt
(head: show the front part contents of myfile.txt.) **tail: show the last part contents of a file Example:**
user> tail myfile.txt
(tail: show the last part contents of myfile.txt.) **aspell: spelling check for a file**

Example:
user> aspell –c myfile.txt
(aspell: spelling check for myfile.txt.
-c: check)

cut: show the specified column of a text file Example:
user> cut –f2 myfile.txt
(cut: show the specified column of myfile.txt.
-f2: specify the second column)

paste: merge two files contents and display Example:
user> paste myfile1.txt myfile2.txt (paste: merge two files contents and display)
sort: show lines of text sorted alphabetically Example:
user> sort myfile.txt
(sort: show lines of text sorted alphabetically.) **stat: display the attributes of a file or directory Example:**
root> stat myfile.txt
(stat: show file name, modify date, change time etc.) **wc: display word count in a file**

Example:
root> wc myfile.txt
(wc: show the number of lines, words, bytes in a file) **file: test the file type**

Example:
root> file myfile.txt
(output: myfile.txt ASCII text)

touch: create a file or change file timestamp Example:
root> touch myfile.txt
(touch: create a file named myfile.txt) **nl: show numbers for each line of a file Example:**
root> nl myfile.txt
(output:
023 sld slwflflf gjo4ijg gj4jf9ej
024 wz wg tjletj geg4t4y
025 sjflew gjlgnu4g jgu675h dk9fh fmj6ju 026 jf5hjd fjtjfj d8gj1nfj,nuigrr ? rit ……)

vi: edit or create a text file with vi editor Example:
root> vi myfile.txt
(vi: open myfile.txt with vi editor)

tr: transform text in a file

Example:

root> echo apple | tr “apple” “banana”
(output: banana)

tee: print standard output, write to a file Example:
Root> sort file1.txt | tee file2.txt (sort file1.txt and write to file2.txt)

Summary

cat: show contents of a file
cat | less: display a file contents page by page cat | more: display a file contents screen by screen head: show the front part contents of a file tail: show the last part contents of a file aspell: spelling check for a file
cut: show the specified column of a text file paste: merge two files contents and display sort: show lines of text sorted alphabetically stat: display the attributes of a file or directory wc: display word count in a file
file: test the file type
touch: create a file or change file timestamp nl: show numbers for each line of a file vi: edit or create a text file with vi editor tr: transform text in a file
tee: print standard output, write to a file

Chapter 7
Comparing File Commands

diff: show differences between two files
cmp: compare two files byte by byte comm: compare two files line by line md5sum: create a md5 checksum number
cksum: create a crc number

diff: show differences between two files Example:
user> diff myfile1.txt myfile2.txt (diff: show differences between two files.) **cmp: compare two files byte by byte Example:**
user> cmp myfile1.txt myfile2.txt (cmp: compare two files byte by byte.) **comm: compare two files line by line Example:**
user> comm myfile1.txt myfile2.txt (comm: compare two files line by line.) **md5sum: create a md5 checksum number Example:**
user> md5sum myfile1.txt
(output: f7tkgu5orj1fjt8kelc2os95nd57jf8r myfile1.txt.) **cksum: create a crc number**

Example:
user> chsum myfile2.txt

(output: 4658791048 19 myfile2.txt.) **Summary**

diff: show differences between two files cmp: compare two files byte by byte comm: compare two files line by line md5sum: create a md5 checksum number cksum: create a crc number

Chapter 8
Matching Text Commands

grep: show all lines that contain a specified string
egrep: show all lines that contain a specified string uniq: show unique lines in a file
find: locate a file in specified directory
look: show words matching a given prefix **grep: show all lines that contain a specified string Example:**
user> grep good myfile.txt
(grep: show all lines that contain "good" string.) **egrep: show all lines that contain a specified string Example:**
user> egrep excellent myfile.txt
(egrep: show all lines that contain "excellent" string.) **uniq: show unique lines in a file**

Example:
user> uniq myfile.txt
(uniq: show unique lines in myfile.txt.) **find: locate a file in specified directory Example:**
user> find /mydir –type f myfile.txt -print (find: locate a file in a directory.
-type f: specify a file
-print: print)

look: show words matching a given prefix
Example:
User> look ab
(output: aba, abb, abc, abd**…**) **Summary**
grep: show all lines that contain a specified string egrep: show all lines that contain a specified string uniq: show unique lines in a file
find: locate a file in specified directory look: show words matching a given prefix

Chapter 9
Directory Commands

mkdir: make a new directory
rmdir: remove a empty directory basename: display the last part of a file path dirname: show the directory path only **mkdir: make a new directory Example:**
user> mkdir mydir
(mkdir: make a new directory) **rmdir: remove a empty directory Example:**
user> rmdir mydir
(rmdir: remove a empty directory) **basename: display the last part of a file path Example:**
user> basename *home*foo/usr/file.txt (output: file.txt)

dirname: show the directory path only Example:

User>dirname *foo*bar/baz/myfile.txt (output: *foo*bar/baz) **Summary**

mkdir: make a new directory rmdir: remove a empty directory basename: display the last part of a file path dirname: show the directory path only

Chapter 10
Un/Compress Commands

zip: compress a file to zip format
unzip: uncompress a file from zip format gzip: compress files to gzip format gunzip: uncompress files from gzip format bzip2: compress files to bz2 format bunzip2: uncompress files from bz2 format **zip: compress a file to zip format Example:**
user> zip myfile.txt
(zip: compress myfile.txt to zip format.) **unzip: uncompress a file from zip format Example:**
user> unzip myfile.zip
(unzip: uncompress myfile.zip.) **gzip: compress files to gzip format Example:**
user> gzip myfile.txt
(gzip: compress a file to gzip format) **gunzip: uncompress a file from gzip format Example:**
user> gzip myfile.txt.gz
(gunzip: uncompress myfile.txt.gz) **bzip2: compress files to bz2 format Example:**
user> bzip2 myfile.txt.
(bzip2: compress myfile to bz2 format) **bunzip2: uncompress files from bz2 format Example:**
user> bunzip2 myfile.txt.bz2
(bunzip2: uncompress myfile from bz2 format) **Summary**
zip: compress a file to zip format unzip: uncompress a file from zip format gzip: compress files to gzip format gunzip: uncompress files from gzip format bzip2: compress files to bz2 format bunzip2: uncompress files from bz2 format

Chapter 11
Processes Commands

ps: show the current processes of user
kill: kill a process by process id w: show all current working process.
df: show disk usage of file system uptime: show system uptime top: view the top active process or a specified process.

ps: show the current processes of user Example:
root> ps –u username
(ps: show the current processes of a user.
-u: specify a user name)

kill: kill a process by process id Example:
root> kill 6270
(kill: kill a process by process id 6270: a process id.)

w: show all current working process Example:
root> w -s
(w: show all current working process.
-s: show summary of process.) **df: show disk usage of file system Example:**
root> df -h
(df: show disk usage of file system.
-h: make the output more understandable) **uptime: show system uptime Example:**
root> uptime
(uptime: show system uptime.) **top: view the top active or specified process Example:**
root> top -p pid
(top: show a process by pid) (-p:display specified process by pid) (pid: process id)

Summary
ps: show the current processes of user kill: kill a process by process id w: show

all current working process.
df: show disk usage of file system uptime: show system uptime top: view the top active process or a specified process.

Chapter 12
Account Commands

useradd: add a new user account
usermod: modify an existing user account userdel: delete an existing user account passwd: set a user account password chfn: change personal finger information finger: display personal user finger information **useradd: add a new user account Example:**
root> useradd username
(useradd: add a new user account.) **usermod: modify an existing user account Example:**
root> usermod –l oldname newname (usermod: modify an existing user account. -l: modify login name.)

userdel: delete an existing user account Example:
root> userdel username
(userdel: delete an existing user account.) **passwd: set a user account password Example:**
root> passwd username
(passwd: set a user account password for a user.) **chfn: change personal finger information Example:**
root> chfn username
(chfn: change finger information for a user).

finger: display personal user finger information Example:
root> finger username
(finger: list the user's login name, email, domain name, time. etc.) **Summary**
useradd: add a new user account usermod: modify an existing user account userdel: delete an existing user account passwd: set a user account password chfn: change personal finger information finger: display personal user finger information

Chapter 13
Groups Commands

groups: show the group membership
groupadd: create a new group
groupmod: modify an existing group groupdel: delete an existing group **groups: show the group membership Example:**
root> groups username
(groups: show the group membership of a user.) **groupadd: create a new group Example:**
root> groupadd newgroup
(groupadd: create a new group named newgroup.) **groupmod: modify an existing group Example:**
root> groupmod newgroup
(groupmod: modify an existing group named newgroup.) **groupdel: delete an existing group Example:**
root> groupdel newgroup
(groupdel: delete an existing group named newgroup.) **Summary**
groups: show the group membership groupadd: create a new group
groupmod: modify an existing group groupdel: delete an existing group

Chapter 14
Permission Commands

chmod: change mode of access permissions
chgrp: change group membership
chown: change ownership of a file or directory (Access permission in detail will be in later chapter.) **chmod: change mode of access permissions**

Example:
root> chmod 752 myfile1.txt
(chmod: change mode of access permission for myfile1.txt.
7: set user permission with read, write, execute 5: set group permission with read, execute 2: set others permission with write only.) **chmod: change mode of access permissions**
Example:
root> chmod g+w myfile.txt
(g+w: give write permission to member of the file's group) **chgrp: change group membership**

Example:
root> chgrp groupname myfile2.txt
(chgrp: change group membership of myfile2.txt) **chown: change ownership of a file or directory Example:**
root> chown username myfile3.txt
(chown: change ownership of myfile3)

Example:
root> chown groupname *user*dir (chown: change ownership of dir.)

Summary:
chmod: change mode of access permissions chgrp: change group membership
chown: change ownership of a file or directory (Access permission in detail will be in later chapter.)

Chapter 15
Running Job Commands

job: display the status of all jobs
fg: run a suspended job in foreground bg: run a suspended job in background kill: kill a job by number or a process by pid at: schedule a job run at a specified time atq: display the scheduled jobs atrm: remove a scheduled job
ps: show current process status w: show who logged on and what doing uptime: show how long the system has been running top: view the top active process crontab: create a job to run at specified time **job: display the status of all jobs Example:**
root> jobs
(job: display the status of all jobs.) **fg: run a suspended job in foreground Example:**
root> fg %2
(fg %2: run a suspended job %2 in foreground.) **bg: run a suspended job in background Example:**
root> bg %3
(bg %3: run a suspended job %3 in background.) **kill: kill a job by number or a process by pid Example:**
root> kill %4
(kill %4: kill a running job %4) **at: schedule a job run at a specified time Example:**
root> at 9:30 pm
(at 9:30: set a schedule job at 9:30.) **atq: display the scheduled jobs Example:**
root> atq
(atq: display all scheduled jobs) **atrm: remove a scheduled job**

Example:
root> atrm 25
(atrm 25: remove a scheduled job 25.) **ps: show current process status Example:**
root> ps -f
(ps-f: show full information of current process.) **ps: show current process status Example:**
root> ps -u username
(ps -u: show a user's current process.) **w: show who logged on and what doing Example:**

root> w -s username
(-s: show summary information of a user.) **uptime: show how long the system running Example:**
root> uptime
(uptime: show system uptime.)

top: view the top active or specified process

Example:
root> top
(top: display all processes running on the system) **crontab: create a job to run at specified time Example:**
root> crontab -e

(-e:edit the crontab file, add a crontab job to the table) **Summary**

job: display the status of all jobs fg: run a suspended job in foreground bg: run a suspended job in background kill: kill a job by number or a process by pid at: schedule a job run at a specified time atq: display the scheduled jobs atrm: remove a scheduled job
ps: show current process status w: show who logged on and what doing uptime: show how long the system has been running top: view the top active process crontab: create a job to run at specified time

Chapter 16
Backup/Restore Commands

cpio: output or input an archive cpio file
tar: create, view, extract archived tar file **cpio -o: output an archive cpio file**
Example:
root> cpio -o > directory**.**cpio (-o: backup to a archive cpio file)

cpio -i: input an archive cpio file
Example:
root> cpio -i < directory**.**cpio (-i: restore from a archive cpio file)

tar -xf: extract an archived tar file
Example:
root> tar -xf archive.tar
(-xf: extract an archive tar file)

tar -cf: create an archived tar file
Example:
root> tar -cf archive.tar
(-cf: create an archive tar file)

Summary

cpio: output or input an archive cpio file tar: create, view, extract archived tar file

Chapter 17
Date & Time Commands

date: display t date and time
cal: display a calendar of month **date: display date and time Example:**
root> date
(date: display the current date and time.) **date "+%A": display current day Example:**
root> date "+%A"
(output: Sunday)

date "+%D": display current date Example:
root> date "+%D"
(output: 08/10/14)

date "+%T": display current time Example:
root> date "+%T"
(output: 11:30:28)

cal: display a calendar of month Example:
root> cal
(cal: display a month calendar.) **Summary**
date: display date and time cal: display a calendar of month

Chapter 18
Networking Commands

host: display remote hostname and IP
ifconfig: display local network configuration ping: send packets to test if remote host reachable ssh: securely connect to a remote computer ftp: files transfer by "File Transfer Protocol"
mesg: enable or disable messaging
write: write a messages to other users open: connect to an ftp server
mail: send and receive mails locally and globally.
dhclient: provides a means for configuring one or more network interfaces
nslookup: query internet name servers interactively for IP information.

host: display remote hostname and IP

Example:
root> host www.yahoo.com
(host: display remote hostname and IP.) **ifconfig: display local network configuration Example:**
root> ifconfig
(ifconfig: display local network configuration) **ping: send packets to test if remote host reachable Example:**
root> ping -c3 yahoo.com
(ping: send packets to test if yahoo host is reachable.
-c3: specify the number of pings)

ssh: securely connect to a remote computer Example:
root> ssh ray@myusername.com
(ssh: securely login to a remote computer) **ftp: files transfer by "File Transfer Protocol"**

Example:
root> ftp ftpexample.myexample.com (ftp: connecting to ftpexample.myexample.com and transfer files remotely) **mesg: show messaging**

Example:
root> mesg
(mesg: show current status of messaging) **mesg y: enable messaging**

Example:

root> mesg y
(mesg y: permit messaging)

mesg n: disable messaging

Example:
root> mesg n
mesg n: deny messaging)

write: write a messages to other users Example:
root> write ken
(write a message to ken)

open: connect to an ftp server

Example:
root> open ftp.myexamples.com
(open: connect to ftp.myexamples.com) **mail: send and receive mails locally and globally.**

Example:
root> mail username@myexamples.com (mail: send a mail to username@myexamples.com) **dhclient: provides a means for configuring one or more network interfaces.**
Example:
root> dhclient eth0
(renew the dynamically assigned IP address of a primary Ethernet device.) **nslookup: query internet name servers interactively for IP information.**
Example:
root> nslookup myexample.com (return an IP address, *e.g.* 75,126,166, 2XX) (nslookup: manually query DNS servers.
The DNS (Domain Name System) protocol allows you to get an IP address for a given host name from a name server. This process is called resolving.)

Summary

host: display remote hostname and IP
ifconfig: display local network configuration ping: send packets to test if remote host reachable ssh: securely connect to a remote computer ftp: files transfer by "File Transfer Protocol"
mesg: enable or disable messaging
write: write a messages to other users open: connect to an ftp server

mail: send and receive mails locally and globally.
dhclient: provides a means for configuring one or more network interfaces
nslookup: query internet name servers interactively for IP information.

Chapter 19
Scripting Commands

echo: display text.
expr: perform math calculation #!/bin/bash: put in the first line of a bash shell scripts file.

echo: display text.

Example:
root> STR=”Hello World!”
root> echo $STR
(echo: display text.
The output is “Hello World!”) **echo -e: display text using escape sequences.**

Example:
root> STR=”Hello World!”
root> echo –e “\n$STR\n”
(echo -e: display text using \n. \n means add a new line.) The output is
“
Hello World!

”

expr: perform math calculation Example:
user> expr 20 + 80
(The output is “100”)

Example:
user> expr 21/7
(The output is “3”)

Example:
user> expr 9 “>” 6
(The output is “1”, the 1 means true) **Example:**
user> expr 9 “<” 6
(The output is “0”, the o means false) **#!/bin/bash: put in the first line of bash**

shell scripts.
Example:
#!/bin/bash
If…then…else…fi
(#!/bin/bash: always put in the first line of bash shell scripts) **Summary**
echo: display text.
expr: perform math calculation #!/bin/bash: always put in the first line of a bash shell scripts

Chapter 20
System Commands

df: show disk usage of file system
mount: make a device available to file system umount: make a device unavailable to file system fsck: check and repair the file system init n: switch the system to run level n who -r: show the current run level free: show free disk space
du: show disk usage of a file or directory export: set an environment variable printenv: list environment variable names and values unset: remove the environment variable clear: clear the screen
exit: exit the shell or logout.
shutdown –h +n: the system is going down in n minutes!

df: show disk usage of file system Example:
root> df -h
(df: show disk usage of file system.
-h: make output understandable)

mount: make a device available to file system Example:
root> mount /cdrom
(mount: make a cd-rom available to file system.) **umount: make a device unavailable to file system Example:**
root> umount /dev/hda1
(umount: make hda1 unavailable to file system.
hda1: a hard drive partition)

fsck: check and repair the file system Example:
root> fsck
(fsck: check and repair the file system.) **init n: switch the system to run level n Example:**
root@user-linux: ~ # init 5
(init 5: switch the system to run level 5) **who -r: show the current run level Example:**
root@user-linux: ~ # who -r
(who -r: show the current run level.) **free: show free disk space**

Example:
root@user-linux: ~ # free -m
(free: show free disk space
-m: show free disk space in MB unit) **du: show disk usage of a file or directory**
Example:
root@user-linux: ~ # du –b myfile.txt (du: show disk usage of a file or directory.
-b: count the number of bytes it occupies.) **export: set an environment variable**
Example:
root@user-linux: ~ # export newvar=8
(echo $newvar. The output is 8)

printenv: list environment variable names and values Example:
root@user-linux: ~ # printenv
(printenv: list environment variable names and values) **unset: remove the environment variable Example:**
root@user-linux: ~ # unset var
(unset var: remove the environment variable var.) **clear: clear the screen**

Example:
user>clear
(clear: clear the screen.)

exit: exit the shell or logout.

Example:
user>exit
(exit: exit the shell or logout.) **shutdown –h +n: the system is going down in n minutes!**
Example:
root@user-linux: ~ # shutdown –h +5
(shutdown –h +5: the system is going down in 5 minutes!) (-h: halt the system
+n: after n seconds)

Summary:

df: show disk usage of file system mount: make a device available to file system

umount: make a device unavailable to file system fsck: check and repair the file system init n: switch the system to run level n who -r: show the current run level free: show free disk space
du: show disk usage of a file or directory export: set an environment variable printenv: list environment variable names and values unset: remove the environment variable clear: clear the screen
exit: exit the shell or logout.
shutdown –h +n: the system is going down in n minutes!

Chapter 21
Helping Commands

Commands	Operations & Examples
man	display manual for a command *e.g.* man nice (show the manual for nice command)
info	display information for a command *e.g.* info chmod (show information about chmod cmd)
whatis	display a description of what a cmd is *e.g.* whatis ifconfig (show a description of what ifconfig is)
help	display help explanation of a cmd *e.g.* help cd (show help explanation of cd cmd)
apropos	search manual pages for a keyword *e.g.* apropos download (show manual entries with “download”)
--help	-help option gets help for a command *e.g.* wget --help (get help for wget cmd)

Note:
If you are familiar with helping commands, you will know about the complete Linux commands and their usages.

Chapter 22
Skill of Commands

Make use of Tab key to auto complete Linux offers so many different commands that we easily to get lost and to forget one. Personally, it happens very regularly, but this is fortunately not a drama. Indeed, Linux offers a variety of ways to find a command that you missed.
The first "trick" to know what is to auto complete control.
Example:
For the **date** command: you're a little headache and you do not know how it is written. By cons, you are sure the first or second letters of the command are “da”.
Just type "da" in the console, then double-tap the **Tab** on the left of your keyboard By double tapping Tab, you asked the computer a list of commands that begin with "da". They said you "dash" and "date". So there are two commands that start with "da", and you just find the one you are looking for, that is to say "date".
Very nice, the computer has rewritten the prompt below and the beginning of the command you typed. You only have to complete with the letters "you" missing and hitting Enter and it will be good. :-) Even more fun, if there is only one result for your search, the computer will complete with missing letters and you only have to press Enter!
Example:
If you want to type “chsh” command and you are not sure the spelling, you can only type “ch” in the console and press two times on Tab. The command is completed magically. It will display “chfn, chsh”, then you can choose “chfn”.
Commaaaandes too!

The command history We often need to find a command that was typed there five minutes (or even five seconds). Sometimes it is because we have forgotten the command, but it's often because you like me you really too lazy to rewrite ourselves the entire command.
This shortcut is gold: press the **Up arrow key**; you will see the last command you typed.
If you press again the directional arrow Top, you will see the penultimate command, then the second-to-last, *etc.*

If you press the **Down arrow key**, you will return to the most recent commands. Thus I can successively find the commands I just type in reverse command: ls --all;
ls -a,
ls;
Date;
Etc.
If you want to "go" very far back into the history of your commands, no need to type a hundred times on the directional arrow Top like madmen.
There is the history command that reminds you of the command history:
Example:
Press the **Up arrow key**; you will see the last command you typed. Then the screen will display: Date 152 ls ls ls 153 154 155--all -a 156 157 history You will notice that the controls are numbered: thus, we can know that **date** is the 152nd command I typed into the terminal, that three **ls** are the 153rd, 154rth and 156th command. The above command you typed will always be history, of course.

Ctrl + R: find a history command

In case the directional arrow Top and history command does not suffice to find an old command you typed, there is a super useful shortcut: **Ctrl+R**. which can help you find out the history command you have just used. So Press **Ctrl+R** keys simultaneously and computer will switch "looking for a typed command" ("R" as research).
There you can type any sequence of letters that corresponds to an old command.
Example:
If you want to look for a command with “all” you have previously used, please press **Ctrl+R** and type "all". Then, Linux will find out ls --all containing just the word "all." You just have to hit Enter to run the command! :-) If this is not the command you are looking for, again press Ctrl + R to move up the list of commands containing "all".
It may look stupid on a drive like that, but some are very long and it is a pleasure not to have to rewrite them again!

Using Wildcards

You can use wildcards with a lot of Linux commands. A wildcard is a symbol or symbols that indicating other characters. There three kinds of wildcards in Linux command: **?** A question mark (?) indicates a single character.
For instance: b??k matches bank, beak, back, bilk, or any other four-letter

filename that begins with b and ends with k.

***** An asterisk (*) indicates any character or set of characters, including no character or many characters.

For instance: b*k matches bk, bkk, bark, break, backtrack.

[ ] Characters enclosed in square brackets ([ ]) usually indicates any character in the set. But please note they are case-sensitive.

For instance: b[a-z]k matches bak, bbk, bck, bzk, but not matches bAk, bBk, bCk, bZk, because of case-sensitive.

Wildcards are actually implemented in the Linux commands.

Example:

user2014@user-linux:~$ ls b??k (The wildcard b??k matches five files in current directory. The output is “bank beak back bilk bark”) **Example:**

user2014@user-linux:~$ ls a*d (The wildcard a*d matches some files in current directory. The output is “ad add acid abroad abounded abed aid”) **Example:**

user2014@user-linux:~$ ls se[a-e]

(The wildcard se[a-e] matches five files in current directory. The output is “sea seb sec sed see”) **About Run Levels** What is a Run Levels? The term run levels refers to a mode of operation in one of the computer operating systems that implement Linux System V-style initialization. Usually, seven run levels exist, numbered from zero to six; Only one "run level" is executed on boot up - run levels are not executed sequentially, *i.e.* either run level 2 OR 3 OR 4 is executed, not 2 then 3 then 4.

"run levels" defines the state of the machine after boot. Different run levels are typically assigned to: 0.halt-the system is in the process of shutting down.

1.single-user mode 2.multi-user mode without network services started 3.multi-user mode with network services started 4.system shutdown 6.system reboot

Example:

root@user-linux: ~ # init 3

(init 3: switch the system to run level 3) **Summary:**

Tab key can auto complete a missing work command.

Up arrow key can view the history commands Down arrow key can return to most recent commands Ctrl+R can find out a history command by a key word.

? A question mark (?) indicates a single character.

***** An asterisk (*) indicates any character or set of characters, including no character or many characters.

[ ] Characters enclosed in square brackets ([ ]) usually indicates any character in

the set. But please note they are case-sensitive.
Run Level number describers the level of services that have been initialized and are running.

Chapter 23
Access Permissions

What is rwx?

After using **ls –l** to view a file's access permissions, you can see something like this: **rwx ….. ……**

Explanation: **rwx** signify the access permissions which can be described by a number from 1 to 7. Really? Yes, the numbers from 1 to 7 indicate the access permissions.
r stands for read permissions, value is **4**.
w stands for write permissions, value is **2**.
x stands for execute permissions, value is **1**.

Vice versa: **4 means read permissions (r) 2 means write permissions (w) 1 means execute permissions (x) Example:** If rwx value is 4, then you can figure out its permission is read only.
If rwx value is 2, then you figure out its permission is write only.
If rwx value is 1, then you figure out its permission is execute only.

The numbers from 1 to 7 indicate the various access permissions: 7 means permissions with read, write, execute (7=4+2+1) 6 means permissions with read, write (6=4+2) 5 means permissions with read, execute (5=4+1) 4 means permissions with read only (4=4+0) 3 means permission with write, execute (3=2+1) 2 means permission with write only (2=2+0) 1 means permission with execute only (1=1+0) Example: If rwx value is 7, then you can figure out its permission is read, write and execute.
If rwx value is 6, then you can figure out its permission is read and write.
If rwx value is 3, then you can figure out its permission is write and execute.

What are rwx rwx rwx?

Access permissions for a file are divided in to three: user permissions, group permissions, others permissions, So, when you use ls –l to view the access

permission, you will find three rwx look like this: **rwx rwx rwx……**
Explanation: The **1st** rwx means **user** permissions, The **2nd** rwx means **group** permissions, The **3th** rwx means **others** permissions.

Example: If the **1st** rwx is 7, it means **user** permissions with read, write, execute.
If the **2nd** rwx is 3, it means **group** permissions with write and execute.
If the **3th** rwx is 6, it means **others** permissions with read and write.

From above you can understand the **1st** rwx, **2nd** rwx and **3th** rwx respectively indicate **user** permissions, **group** permissions and **others** permissions.

Example: 752 means: **1st** rwx is 7 meaning **user** permissions with read, write, execute.
2nd rwx is 5 meaning **group** permissions with read, and execute.
3th rwx is 2 meaning **others** permissions with write only **chmod** is a command, meaning change the mode of access permissions for a file.

Example: chmod 643 myfile.txt Explanation: Change myfile.txt permissions to 643.
1st rwx is 6 meaning **user** permissions with read, and write.
2nd rwx is 4 meaning **group** permissions with read only.
3th rwx is 3 meaning **others** permissions with write, execute **Example:** chmod 751 myfile.txt Explanation: Change myfile.txt permissions to 751.
1st rwx is 7, meaning **user** permissions with r, w, x.
2nd rwx is 5 meaning **group** permissions with r, x.
3th rwx is 1 meaning **others** permissions with x only.

Example: Chmod u+r (gives the user a read permission) **Example:** Chmod g-x (remove execute permission from members of the file's group) **Example:** Chmod o-w (remove write permission from others) **Summary:** **4** means read permissions (r) **2** means write permissions (w) **1** means execute permissions (x) The numbers from 1 to 7 indicate the various access permissions: **7** means permissions with read, write, execute (7=4+2+1) **6** means permissions with read, write (6=4+2) **5** means permissions with read, execute (5=4+1) **4** means permissions with read only (4=4+0) **3** means permission with write, execute (3=2+1) **2** means permission with write only (2=2+0) **1** means permission with

execute only (1=1+0)

Chapter 24
Linux Symbols

Commands	Operations & Examples
<	get input from a file to a command *e.g.* cat < myfile.txt
>	send output from a command to file *e.g.* cat > myfile.txt
>>	append output to a file *e.g.* cat file1.txt >> file2.txt
|	send cmd1 output to cmd2 input *e.g.* ls -al /etc | less
;	combine two or more commands *e.g.* cd~**;** ls
\	escape the special character *e.g.* echo -e "\n Hello \t World! \n"
./	run a script in the current directory *e.g.* **.**/script.sh
..	parent directory *e.g.* cd**..**
~	home directory *e.g.* cd~
$	variable prefix for variable name and value *e.g.* echo $var
$$	show the running processes number *e.g.* echo $$
!!	repeat the last command *e.g.* **!!**
!string	run recent cmd that begins with string *e.g.* **!**cat

Chapter 25
Shortcut Keys

CTRL+B	move the cursor backward only one character.
CTRL+C	cancel the running command or kill the running process.
CTRL+D	log out of the current session. similar to exit command.
CTRL+F	move the cursor forward only one character.
CTRL+H	erase one backward character. similar to pressing backspace.
CTRL+P	paste the previous line(s) to one specified location.
CTRL+R	type to bring up recent commands, return a list of commands in history
CTRL+S	stop all output on screen. freeze the shell as it locks the terminal output
CTRL+Q	resume all stopped output on screen, continue the terminal output.
CTRL+U	erase the complete line where the cursor locates.
CTRL+W	delete the last recent word you have just typed in.
CTRL+Z	suspend a running process. if want to resume, use fg or bg commands.

Chapter 26
Run Levels Table

Run Level	Description
0	**Halt the system.** When this is the current run level, the system is in the process of shutting down.
1	**Single user mode.** Only small set of kernel processes running. Almost all other services disabled.
2	**Basic multiuser mode.** This run level starts most services, but does not enable network connection service.
3	**Full multiuser mode.** This run level starts all services including network connection. but does not start X window
4	**User defined mode.** No conventional definition applies to run level 4. It is fully open to user configuration.
5	**Full multiuser mode with X window.** Starts all enabled services with Linux graphical desktop environments.
6	**Reboot.** When this is the current run level, the system is in the process of rebooting.

Chapter 27
The Vi Editor Commands

Commands	Operations
h,l,k,j	cursor move left, right, up, down
w,b	cursor move forward, backward
0	go to the beginning of line
$	go to the ending of line
G	go to the last line of the file
J	join current line with next line
z	undo last command
.	repeat last command
ZZ	save file and exit
i	insert before the character at cursor
I	insert character to beginning of line
a	append after the character at cursor
A	append character to the ending of line
c	change until…
C	change to end of line
d	delete until…
D	change to end of line
r	replace one character
R	replace more characters
o	open a new line below
O	open a new line above
esc	exit insert mode, go to cmd mode
x	delete the character at cursor
X	delete the character at left
u	undo last change
.	redo last change
U	restore line

m	mark position
M	middle of screen
dw	delete current word
dd	delete current line
cc	change current line
f x	find x on current line
F x	find x on previous line
/string	search string, look forwards
?string	search string, look backwards
[n]G	go to line number “n”
n	search forward next
N	search backward next
p	paste line(s) below current line
P	paste line(s) above current line
t	to…
T	backward to…
s	substitute
S	substitute entire line
ZZ	write and quit
:e	edit file
:e!	forget change of file
:$	go to last line of file
:number	go to line number
:w	write and save
:w!	save with read-only file
:w filename	save with new file name
:wq	write, save and quit
:n	go to next file
:rew	go to the first file
:q	quit after using :w to save
:q!	quit without saving
:r filename	read file, insert file at cursor position
:! command	run a shell command
:sh	run a temporary shell

Chapter 28
All Essential Linux Commands

Note:
cmd means command
regex means regular expression
(*) means the command run by root user

Commands	Operations & Examples
a2p	translate awk to perl *e.g.* a2p myfile.awk> myfile.pl (translate myfile.awk into pl file)
alias	create another name for a command *e.g.* alias p="pwd" (set p as alias for pwd)
apropos	view the searched term in man pages *e.g.* apropos find (list entries with "find" in man page)
apropos -e	view searched term in man pages *e.g.* apropos –e nice (-e: show exact word in man pages)
apt-get	install, remove or update a package *e.g.* apt-get install libc6 (install libc6 package)
aspell	check and correct for misspellings *e.g.* aspell -c test.txt (-c:check spelling in test.txt file)
at	run a job at a schedule time *e.g.* at 1 AM Fri (run the job at 1am Friday)
awk	match text by regular expression *e.g.* awk 'length($0) > 88' text.txt (list only lines longer than 88

	words)
basename	display the last part of a file path *e.g.* basename */home*foo*/usr*file.txt (output: file.txt)
bc	perform a calculation by a calculator *e.g.* bc 8+9 (output: 17)
bg	resume a stopped job in background *e.g.* bg %3 (resume %3 job in background)
bunzip2	uncompress a file from zip format *e.g.* bunzip2 myfile.tar.bz2 (uncompress myfile.tar.bz2)
bzip2	compress a file to zip format *e.g.* bzip2 myfile.dat (compress myfile.dat)
cal	display a month or year calendar *e.g.* cal 2014 (display 2014 calendar)
cal -3	display a month or year calendar *e.g.* cal -3 (-3:display 3 months)
cat	display contents of one of more files *e.g.* cat file1.txt file2.txt (display contents of file1 and file2)
cat -n	display contents of one of more files *e.g.* cat –n myfile.txt (-n: specify number of output lines)
cd	change directory *e.g.* cd */home*user/mydir (change current directory to mydir)
chattr	set attributes for a file *e.g.* chattr +i myfile.txt (+i make the file as read-only)
chfn	change user’s finger information *e.g.* chfn (change all users information)

chfn -f	change user's finger information *e.g.* chfn –f Full-Name (-f: change full name)
chgrp *	change the group ownership *e.g.* chgrp groupname */usr*myfile.txt (alter group ownership of myfile.txt)
chkconfig*	view and modify run level file *e.g.* chkconfig –list (-list: list services of run level)
chmod	change access permission *e.g.* chmod 755 filename (set file access permission as 755)
chown *	change ownership of file or directory *e.g.* chown username myfile.txt (alter file ownership of myfile.txt)
chpasswd*	change password for users. *e.g.* chpasswd (then enter username**:** password)
chsh *	change login shell for a user *e.g.* chsh -s */bin*bash ray (-s specify login shell)
cksum	produce a CRC checksum number *e.g.* cksum file.txt (output checksum number of file.txt)
clear	clear the screen. *e.g.* clear (clear the shell window)
cmp	compare two files text byte by byte *e.g.* cmp first.txt second.txt (compare first.txt and second.txt)
collectl	monitor the current system status *e.g.* collectl (list cpu, sys, inter .etc information)
comm	compare two files text line by line *e.g.* comm first.txt second.txt

	(compare first.txt second.txt)
cp	copy file(s) to another directory *e.g.* cp myfile.txt /mydir (copy myfile.txt to mydir)
cp -p	copy file(s) to another directory *e.g.* cp –p myfile.txt /mydir (-p: keep original permission)
cp -a	copy file(s) to another directory *e.g.* cp –a myfile.txt /mydir (-a: keep original attributes)
cpio -o	output archived cpio file *e.g.* cpio -o > directory**.**cpio (-o: backup to an archive cpio file)
cpio -i	input archived cpio file *e.g.* cpio -i < directory**.**cpio (-i: restore from an archive cpio file)
crontab	create a job to run at specified time *e.g.* crontab (set to run jobs at regular intervals.)
crontab -e	run a recurring job at a specified time *e.g.* crontab -e (-e allow edit the crontab file)
crontab -l	run a recurring job at a specified time *e.g.* crontab -l (-l: lists the crontab files)
crontab -r	run a recurring job at a specified time *e.g.* crontab -r (-r: remove the crontab file)
csplit	split a file into some separated files *e.g.* csplit myfile.txt ”/part1/” “*part2*” (separate files named xx00, xx01)
cut -d	show the specified field of a file *e.g.* cut –d “**:**” myfile.txt (-d: specify a field delimiter “**:**”)
cut -c	extract contents from a file *e.g.* cut –c 6 myfile.txt

	(-c6: the sixth character of each line)
cut -f	extract contents from a file *e.g.* cut –f 3 myfile.txt (-f3: specify a field number as 3)
date	show the date and time *e.g.* date (display the current date and time)
date -s	set the date and time *e.g.* date -s "11/20/2014" (-s set the date)
dc	open a command line desk calculator *e.g.* dc (“dc” means desk calculator)
dd	data dump to convert and copy a file *e.g.* dd if=*dev*sda of=*dev*sdb (copy data from sda to sdb)
dmesg	print out all kernel log messages *e.g.* dmesg > kmsg.txt (output kernel messages to kmsg.txt)
df	display free disk space *e.g.* df (display file system free space)
df -m	display free disk space *e.g.* df -m (-m: display sizes in Mb)
dhclient	configure network interfaces *e.g.* dhclient eth0 (renew IP address of eth0)
diff	show difference between two files *e.g.* diff firstfile.txt secondfile.txt (display difference above two files)
diff3	show difference among three files *e.g.* diff3 file1.txt file2.txt file3.txt (display difference above three files)

dig	display the details of DNS servers *e.g.* dig xvxaxx.com (list information about xvxaxx.com)
dir	show directory contents *e.g.* dir (display current directory contents)
dircolors	show color settings for “ls” command. *e.g.* dircolors (display directory coloring of ls)
dirname	remove the last part of a file path. *e.g.* dirname *foo*bar/baz/myfile.txt (output: *foo*bar/baz)
du -s	display disk usage *e.g.* du -s *.* (-s list files size in current directory)
du -h	display disk usage *e.g.* du -h myfile.txt (-h: show human readable units)
dump -f	makes backup of filesystem *e.g.* dump -f0 filebk /mydir (-f:backup -0:dump-level filebk: dump-file)
echo	display input on standard output *e.g.* echo “Hello World!” (show “Hello World!”)
echo -e	display text using escape sequence *e.g.* echo –e “\n Hello World!” (-e: allow use \n to show text)
ed	open a command-line text editor *e.g.* ed myfile.txt (open myfile.txt with text editor)
egrep	search file(s) for a specified regex *e.g.* egrep "new | string" myfile.txt (search myfile.txt for “new | string”)
eject	eject the cd or dvd tray *e.g.* eject cdrom

	(eject cdrom tray)
eject -t	eject or close the cd or dvd tray *e.g.* eject –t cdrom (-t: close an open cdrom tray)
emacs	powerful, extensible file editor *e.g.* emacs file.txt (launch emacs and open file.txt)
env	show, set the environment variables *e.g.* env (list current environment variables)
eval	make a command from its arguments *e.g.* UPLS=”eval cd**.. ;** ls ” (create a command named UPLS)
exit	exit the shell *e.g.* exit (terminate the program and log out)
expand	convert tabs into spaces *e.g.* expand myfile.txt (convert tabs to spaces for myfile.txt)
expand -t	convert tabs into spaces *e.g.* expand -t 3 myfile.txt (-t: set tabs 3 characters apart)
export	set an environment variable & value *e.g.* export newvar=8 echo $newvar (output: 8)
expr	evaluate an expression *e.g.* expr 10+8 (output: 18)
factor	show the prime factors of a number *e.g.* factor 1001 (output: 7 11 13)
fc	list, edit, re-execute last commands *e.g.* fc -l (-l:list the history of commands)
fdisk*	manipulate the hard disk partitions

	e.g. fdisk *dev*hdb (list hard disk partitions information)
fg	resume a stopped job in foreground *e.g.* fg %3 (resume the job 3 in foreground)
fgrep	search file(s) for a specified string *e.g.* fgrep "good" myfile.txt (search myfile.txt for “good”)
file	detect the file type. *e.g.* file myfile.tar (determine file type of myfile.tar)
find –print –name	find file(s) in a directory named dir *e.g.* find dir -print -name 'abc.txt' (-print:print, -name:specify file name)
finger	show user’s information *e.g.* finger username (list the user's login name, time. etc.)
fmt	format text files *e.g.* fmt myfile1.txt > myfile2.txt (format myfile1 & output to myfile2)
fmt -u	format text files *e.g.* fmt –u myfile.txt (-u: provide uniform word spacing)
fold	wrap each line to fit a specified width *e.g.* fold -w 15 myfile.txt (-w specify how many words)
for in	set conditional parameter for loop *e.g.* for n in 3 6 9 do echo $n done (output: 3 6 9)
free	displays free memory information *e.g.* free (list free, used, total memory…)
free -m	displays free memory information

	e.g. free -m (-m: show sizes in Mb.)
fsck *	file system check *e.g.* fsck (check or fix Linux file system)
ftp	transfer files by File Transfer Protocol *e.g.* ftp ServerURL (transfer files using ftp)
gawk	find or replace text in a file *e.g.* gawk 'length($0) > 88' (list lines longer than 88 characters)
grep	match a specified string or regex. *e.g.* grep onestring myfile.txt (search myfile.txt for onestring)
groups	list groups to which the user belongs *e.g.* groups (print the groups of user)
groupadd *	add a new group *e.g.* groupadd newgroup (create a new group)
groupadd * -f	add a new group *e.g.* groupadd –f newgroup (-f: check group doesn't exist)
groupdel *	delete an existing group. *e.g.* groupdel existinggroup (remove an existing group)
groupmod *-n	modify an existing group *e.g.* groupmod -n newgrp oldgrp (-n change group name)
gunzip	uncompress a file from gzip format *e.g.* gunzip myfile.txt.gz (uncompress myfile.txt.gz)
gzip	compress a file to gzip format *e.g.* gzip myfile.txt (compress myfile.txt)
halt	shutdown the system *e.g.* halt

	(power off the system)
hash	access the hash table *e.g.* hash (list commands from hash table)
head	display some front lines in a file *e.g.* head myfile.txt (output the first ten lines of myfile.txt)
head -n	display some front lines in a file *e.g.* head –n 4 myfile.txt (-n: specify a number of lines)
help	show help information of commands *e.g.* help echo (show information about echo)
history	show the commands history *e.g.* history (list commands in this shell session)
host	find the ip address of a domain name *e.g.* host websprogram.com (show ip of websprogram.com)
hostid	display id of the current host in hex. *e.g.* hostid (print the current host id)
hostname	show or set the host name *e.g.* hostname (display the name of current host)
id	show the user or group id number *e.g.* id (display the root user uid, gid. etc.)
ifconfig	show, configure the network interface *e.g.* ifconfig (display the network settings)
init *	set the system run level *e.g.* init 5 (change to run level 5)
info	show help information of a

	command *e.g.* info man (show help page for man)
install	copy files, set permission, ownership *e.g.* install myfiles *home*user (copy myfiles to user directory)
install -o	copy files, set permission, ownership *e.g.* install –o myfiles ray /home (-o: specify ownership)
jobs	show all jobs' status *e.g.* jobs (list all running jobs' information)
join	join lines of files having common field *e.g.* join myfile1.txt myfile2.txt (join lines of two files by same field)
join	join lines of two files *e.g.* join –i myfile1.txt myfile2.txt (-i: ignore the differences)
kill	stop a job by number. *e.g.* kill %3 (terminate job %3)
kill	stop a process by pid. *e.g.* kill 3956 (terminate process 3956)
killall	stop a process by name *e.g.* killall no respond (terminate process no respond)
last	show most recently logged-in users *e.g.* last (list recent users' date, time…)
lastb	show bad login attempts *e.g.* lastb (display bad login attempts)
lastlog	display the last login information *e.g.* lastlog -u username

	(-u: specify a user)
less	show contents page by page *e.g.* less myfile.txt (display myfile.txt page by page)
let	perform arithmetic on shell variables *e.g.* let a=12; let a=a+8; echo $a (output: 20)
link	create a link to a file. *e.g.* link file1.txt file2.txt (create a link from file1 to file2)
ln	create a hard link to a file *e.g.* ~/myfile.txt (create a hard link to myfile.txt)
ln -s	create a link between two files *e.g.* ln -s file1.txt file2.txt (create a symbolic link to file1/file2)
locate	find the location of a file or a directory *e.g.* locate myfile.txt (locate myfile.txt on local machine)
logname	show the current user's login name *e.g.* logname (display the login name of user)
look	show words matching a given prefix *e.g.* look ab (output: aba, abb, abc, abd…)
lpc	run the line printer control program *e.g.* lpc status (show status of current print queue)
lpq	show the printer queue status *e.g.* lpq (list the print queue)
lpr	send a print request to printer *e.g.* lpr myfile.txt (send myfile.txt to printer)
lprm	cancel the printing job in print queue *e.g.* lprm 2 (remove printing job 2)

ls	list the contents of current directory *e.g.* ls (list files and sub-directories)
ls -l	long list contents of current directory *e.g.* ls -l (-l:long lists including permissions)
ls -a	lists all entries including hidden files *e.g.* ls -a (-a: show all files)
ls -t	lists all entries by time stamps *e.g.* ls -t (-t: show by time stamps)
ls -lh	lists contents in current directory *e.g.* ls -lh (-lh: list files with size in mb and gb.)
lsattr	list the attribute of a file or a directory *e.g.* lsattr myfile.txt (show myfile.txt attribute)
lsof	list opened files *e.g.* lsof (list all opened files)
man	get command help from manual *e.g.* man cat (show manual page for cat)
man -k	search manual pages for keyword *e.g.* man –k printf (-k: specify a keyword)
md5sum	create a md5 checksum number *e.g.* md5sum -c file.txt (-c validate file against a checksum)
mesg	enable or disable messaging *e.g.* mesg (show the current write status)
mesg y/n	enable or disable messaging *e.g.* mesg y/n (y or n: permit or deny messaging)

mail	send and receive mails *e.g.* mail ray@websprogram.com (email to ray@websprogram.com)
mkdir	make a new directory *e.g.* mkdir mydir (create a directory named mydir)
mknod	make a device file *e.g.* mknod */dev*dk b 45 0 (dk:device; b:block; 45:major no.; 0:minor no.)
more	show content one screen at one time *e.g.* more +2 myfile.txt (+2: beginning at line 2)
mount	mount a storage device *e.g.* mount */dev*cd (mount a device cd)
mount -l	mount or show devices *e.g.* mount -l (-l: list all mounted devices)
mt	magnetic tape drive control *e.g.* mt -f */dev*tape eod (-f select eod; move to end of data)
mv	move a file to another directory *e.g.* mv myfile.txt */home*user/mydir (move myfile.txt to mydir directory)
mv	rename a file *e.g.* mv myfile1.txt myfile2.txt (rename mayfile1 to myfile2)
netstat	display network status *e.g.* netstat (print network connections, routing tables.etc)
nice *	set the priority level of a job *e.g.* nice -19 ftp (set priority level as 19 for ftp)
nl	add text with number lines *e.g.* nl mylist.txt (make number lines for mylist.txt)

nohup	ignore hangup signals *e.g.* nohup find ftp (run ftp ignoring hangup signals)
nslookup	query internet name servers for IP *e.g.* nslookup myxxexample.com (return IP like 75.126.162.XXX)
passwd	modify a user password *e.g.* passwd username (change password for username)
paste	merge lines of multiple files *e.g.* paste file1.txt file2.txt (merge contents for file1 and file2)
pidof	show process ID of running program *e.g.* pidof console (display console's process id)
ping	send data to a host, await response *e.g.* ping xvxmjfz.com (test if remote host can be reached)
ping -c	test if remote host can be reached *e.g.* ping –c5 xvxmjfz.com (-c5: specify the number of pings)
pkill	kill a running process *e.g.* pkill firefox (stop web browser firefox)
pr	prepare text files for printing *e.g.* pr myfile.txt (prepare myfile.txt for printing)
pr -n	prepare text files for printing *e.g.* pr –n myfile.txt (-n: specify number in each line)
pr -h	prepare text files for printing *e.g.* pr –h "Good" myfile.txt (-h: specify a header)
printenv	show the environment variables *e.g.* printenv (list values of environment

	variables)
printf	format and print data *e.g.* printf "start\b" (\b: backspace output: star)
printf	format and print data *e.g.* printf 'hello \n world \n !' (\n: prints by newlines. output 3 lines)
ps -f	show the process status *e.g.* ps -f (-f full information of current process)
ps -u	show the process status *e.g.* ps -u ray (-u specify a user's current process)
pstree	displays process in tree structure *e.g.* pstree (show all process as a tree)
pwd	print working directory *e.g.* pwd (display current directory)
rcp	remotely copy file between two hosts *e.g.* rcp file.txt host2**:**/dir2/file.txt (remotely copy file.txt to host2)
read	read a line from standard input *e.g.* read name (input ray) echo "$name" (output ray)
reboot*	restart the system *e.g.* reboot (cause the computer to restart)
renice *	change the priority level of a job *e.g.* renice 3 23001 (set priority level as 3 for job 23001)
restore	restores data from the backup file *e.g.* restore -f databackup

	(-f:specify a backup file)
rlogin	remotely login to a system *e.g.* rlogin -l username domain.com (-l: specify a username)
rm	remove one or more files *e.g.* rm myfile.txt (remove myfile.txt)
rm -r	remove non-empty directory *e.g.* rm -r /NonEmptyDir (-r: remove directory and its content)
rm -i	remove non-empty directory or a file *e.g.* rm -i myfile.txt (-i: ask before removing)
rmdir	remove empty directory *e.g.* rmdir /emptydir (delete directory without contents)
route	show or modify the IP routing table *e.g.* route -n (-n: show in numerical format)
rsync	remotely synchronize files *e.g.* rsync myfile host2**:**/dir2/myfile (sync. myfile with remote host2)
scp	securely copy files amid two hosts *e.g.* scp file.txt host2**:**/dir2/file.txt (securely copy file to remote host2)
screen	open the terminal window manager *e.g.* screen (start a new screen)
sdiff	show two files' difference side by side *e.g.* sdiff myfile1.txt myfile2.txt (compare two files side by side)
sed	filter and transform input text *e.g.* sed "{print $3}" myfile.txt (display the third word of each line)
seq	list sequent numbers in given range

	e.g. seq 1 8 (output: 1 2 3 4 5 6 7 8)
seq -w	list sequent numbers in given range *e.g.* seq –w 1 3 (-w: with zeros output: 01 02 03)
seq	list sequent numbers in given range *e.g.* seq –s| 1 3 (-s: with separators output: 1| 2 |3)
set	set shell variable or function *e.g.* set n=`who am i`; echo $n (output: who am i)
sftp	securely transfer files by ftp *e.g.* sftp SeverURL (securely transfer files to a Server)
shopt	show the shell option settings *e.g.* shopt (show the shell behavior settings)
shutdown *	close system *e.g.* shutdown 22:00 (shut down at 22:00 o'clock)
shutdown-h	close system *e.g.* shutdown –h +5 (-h+5: halt after 5 minutes)
shutdown -r	shutdown and restart *e.g.* shutdown -r now (-r:shut down and instantly restart)
sleep	pause for a specified amount of time *e.g.* sleep 10 (pause for 10 seconds)
sort	show sorted contents alphabetically *e.g.* sort -r myfile.txt (-r sort file in reverse order)
split -b	split a file to some files in given size *e.g.* split -b 11 file.txt (split file to some 11 byte files named xaa,xab,xac, etc.)
split -l	split a file to some files in given size

	e.g. split -l 8 file.txt (split file to some 8 line files named xaa,xab,xac, etc.)
ssh	login to remote secure shell *e.g.* ssh ray@myexample.com (securely connect to a remote host)
ssh -l	login to remote secure shell *e.g.* ssh -l username hostname (-l specify your remote username)
stat	list status about file size, access, *etc.* *e.g.* stat myfile.txt (show myfile.txt statistics)
su	switch user *e.g.* su user2 (switch user named user2)
su -l	login as a root super user *e.g.* su –l (-l:enter password, login root account)
sudo -u	execute a command as another user *e.g.* sudo -u user2 ls *home*mydir (-u: specify user2 to execute ls cmd)
sudo -v	refresh the authentication timeout *e.g.* sudo -v (next sudo will not require password.)
sudo -k	expire the authentication timeout *e.g.* sudo -k (next sudo will require password.)
sum	summarize a file with a checksum *e.g.* sum myfile.txt (create a checksum for myfile.txt)
suspend	suspend the working shell *e.g.* suspend (pause system during execution)
sync	synchronize disk data with memory *e.g.* sync

	(flush all file system buffers to disk)
tac	display lines of a file in reverse order *e.g.* tac myfile.txt (print file from last line to first line)
tail	show the final part of a file *e.g.* tail -n 20 file.txt (-n20:output last 20 lines of file.txt)
talk	communicate with another user *e.g.* talk ray (talk to user ray)
tar -xf	extract an archived tar file *e.g.* tar -xf archive.tar (-xf: extract an archive tar file)
tar -cf	create an archived tar file *e.g.* tar -cf archive.tar (-cf: create an archive tar file)
tee	print standard output, write to a file *e.g.* sort file1.txt | tee file2.txt (sort file1.txt and write to file2.txt)
tee -a	print standard output, write to a file *e.g.* sort file1.txt | tee –a file2.txt (-a: append instead of overwrite)
test	calculate a boolean expression *e.g.* [8 -gt 6]; echo $? (output: 0 0:true; 1:false)
test	calculate a boolean expression *e.g.* [5 -eq 6]; echo $? (output: 1 0:true; 1:false)
time	show the time taken to run a program *e.g.* time ftp (display the time taken to execute ftp)
times	show the uptime of sell *e.g.* times (display the system uptime)
tload	show a graphic report of system load *e.g.* tload (show the current

	system load average to a specified process)
top	list the top active or specified process *e.g.* top -p pid (-p:display specific process by pid)
touch	update timestamp for an existing file *e.g.* touch myfile.txt (modify myfile.txt to the current time)
touch -t	update timestamp for an existing file *e.g.* touch –t myfile.txt (-t: specify a time)
tr	translates sets of characters *e.g.* echo apple | tr “apple” “banana” (output: banana)
traceroute*	trace the route to a host *e.g.* traceroute xvauhdhxv.com (trace packets route to another host)
trap	run a command on receiving a signal *e.g.* trap (display the current signal traps)
tree -p	list directory contents in tree format *e.g.* tree –p (-p: also show the file permissions)
tty	show the name of the terminal device *e.g.* tty (show the terminal filename)
type	detect the type of a command *e.g.* type wait (output: wait is a shell builtin)
ulimit -a	limit user resources *e.g.* ulimit -a (-a:display all limits for the system)
umask	show or set the file permission value *e.g.* umask 0022 (allow user read, write privileges and all others to read)

umask	show or set the file permission value *e.g.* umask 0002 (allow group read, write privileges and all others to read)
umask	show or set the file permission value *e.g.* umask 0077 (allow user read, write privileges and no for others)
umount	unmount a device or filesystem *e.g.* umount *dev*dvd (unmount a device DVD)
unalias	remove an alias *e.g.* unalias aliasname (delete a specified alias)
uname -a	show the current system information *e.g.* uname -a (-a: display all information)
uname -n	show the current system information *e.g.* uname -n (-n: display the host name)
unexpand	convert spaces into tabs *e.g.* unexpand myfile.txt (convert spaces to tabs for myfile.txt)
uniq	filter out repeated lines in a file *e.g.* uniq myfile.txt (show unique line in myfile.txt)
unset	remove shell variable or function *e.g.* unset var (delete a variable)
unzip	uncompress files from zip format *e.g.* unzip archive.zip (uncompress file from archive.zip)
uptime	show system uptime *e.g.* uptime (display system uptime)
useradd *	add a new user account *e.g.* useradd username

	(create a user account)
useradd* -d	display default value for new users *e.g.* useradd -d (show default data for a new user)
userdel *	delete an existing user account *e.g.* userdel username (remove a user account)
usermod* -d	modify home directory *e.g.* usermod -d *home*mydir andy (-d: specify home directory for andy.)
usermod * -l	modify an existing user account *e.g.* usermod -l oldname newname (-l: login name change)
usermod * -p	modify an existing user account *e.g.* usermod -p password username (-p:modify password of a user)
users	display current logged-in users *e.g.* users (list users currently logged in)
vdir	verbosely show directory contents *e.g.* vdir (vdir just like ls, but more verbose)
vi	open the vi text editor *e.g.* vi filename (open a file with vi text editor)
vmstat	report virtual memory statistics… *e.g.* vmstat (also report swap, disk i/o devices…)
w	list current processes for each users *e.g.* w username (show the user's process)
w -s	list current processes by summary *e.g.* w -s (-s: show a summary of shell process)
wait	wait for a process to change state

	e.g. wait 10788 (wait for 10788 to change state)
watch	execute a command periodically *e.g.* watch –n 5 date (-n5:update date every 5 seconds)
wc	show word count, line count, etc *e.g.* wc myfile.txt (list word, line count… for myfile.txt)
wc -c	show word count, line count, etc *e.g.* wc –c myfile.txt (-c: show the byte counts)
wget	download a web page from a website *e.g.* wget http://www.xvfwkaljo.com (download webpage from above url)
wget -c	download a web page from a website *e.g.* wget –c http://www.examp.com (-c: continue download previous web)
whatis	show manual page of a command *e.g.* whatis ping (show manual page of ping)
whereis	locate source, man for a command *e.g.* whereis ls (show source, man locations of ls)
which	show path of a executable command *e.g.* which ftp (show the full path of ftp command)
who	show who currently logged in *e.g.* who (list all logged-in users, date, time…)
who -a	list all users currently logged in *e.g.* who -a

	(-a: all information)
whoami	show the current user's login name *e.g.* whoami (show your own user name)
whois	show domain owner's information *e.g.* whois xvqizx.com (list xvqizx.com owner's information)
write	write a message to another user *e.g.* write username (then write your message…)
xargs	execute a command with arguments *e.g.* find -name " a*.* " | xargs rm (find files named a*.*, remove them)
xcalc	launch a graphical calculator *e.g.* xcalc (open a scientific calculator)
xclock	launch graphical clock *e.g.* xclock -digital (-digital: specify a digital clock)
yes	output a string repeatedly until killed *e.g.* yes "hello" (output hello repeatedly until killed)
yum	rpm-based package manager *e.g.* yum install update (install a package named "update")
zcat	output compressed text *e.g.* zcat myfiles.txt.gz | less (uncompress file and show contents)
zless	show un/compressed file contents *e.g.* zless myfile.txt.gz (zless: show contents by page)
zmore	show un/compressed file contents *e.g.* zmore myfile.txt.gz (zmore: show contents by screen)
zip	compress files to zip format

	e.g. zip documents * (create documents**.**zip for all files)
unzip	uncompress files from zip format *e.g.* unzip myfile.zip (uncompress myfile.zip.)

Conclusion

My friends,
This book is only for a basic Linux commands quick learning. Thank you for your support!
I will greatly appreciate if you kindly give a positive review to this book.
Thank you very much!

Best Regards

Sincerely

Ray Yao

My friend, See you!