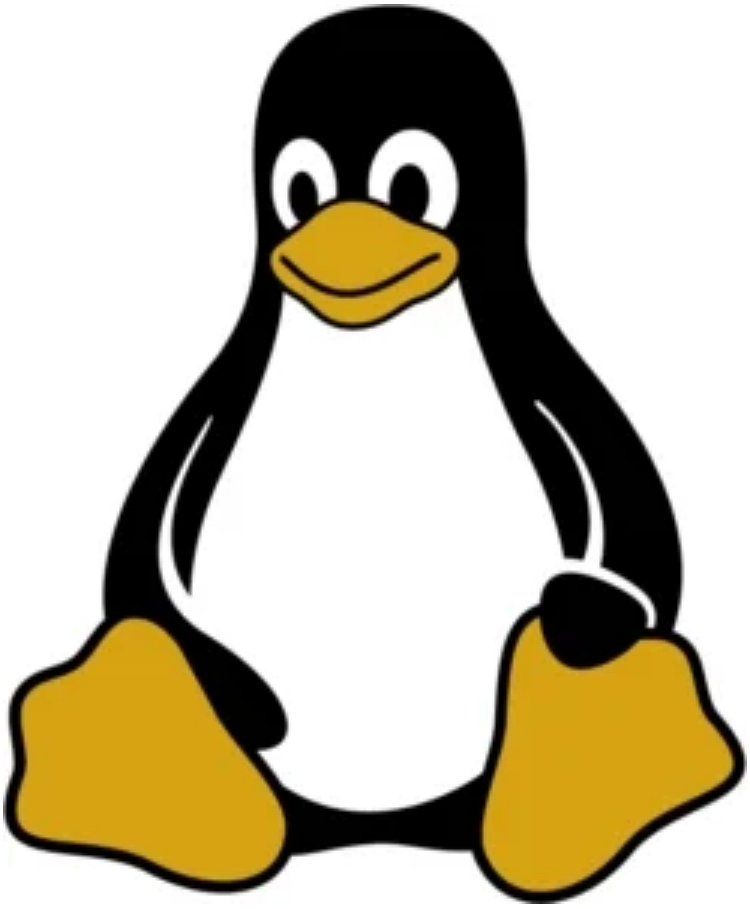


Linux is an operating system's kernel. You might have heard of UNIX. Well, Linux is a UNIX clone. But it was actually created by Linus Torvalds from Scratch. Linux is free and open-source, that means that you can simply change anything in Linux and redistribute it in your own name! There are several Linux Distributions, commonly called “distros”.

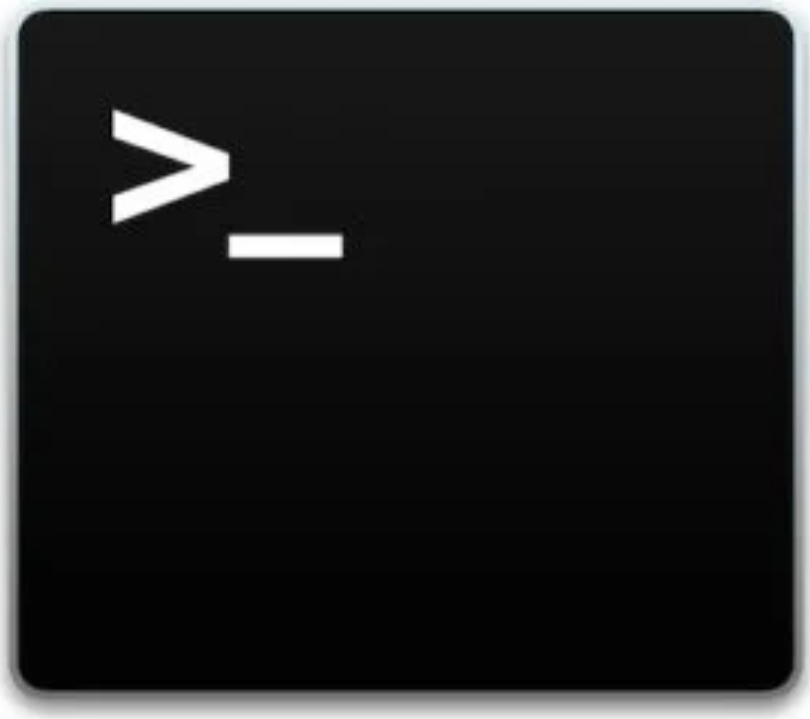
- Ubuntu Linux
- Red Hat Enterprise Linux
- Linux Mint
- Debian
- Fedora

Linux is Mainly used in servers. About 90% of the internet is powered by Linux servers. This is because Linux is fast, secure, and free! The main problem of using Windows servers are their cost. This is solved by using Linux servers. The OS that runs in about 80% of the smartphones

in the world, Android, is also made from the Linux kernel. Most of the viruses in the world run on Windows, but not on Linux!

Linux Shell or "Terminal"

So, basically, a shell is a program that receives commands from the user and gives it to the OS to process, and it shows the output. Linux's shell is its main part. Its distros come in GUI (graphical user interface), but basically, Linux has a CLI (command line interface). In this tutorial, we are going to cover the basic commands that we use in the shell of Linux.



To open the terminal, press Ctrl+Alt+T in Ubuntu, or press Alt+F2, type in gnome-terminal, and press enter. In Raspberry Pi, type in lxterminal. There is also a GUI way of taking it, but this is better!

Linux Commands

Basic Commands

1. pwd — When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the **“pwd”** command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash(/). The user directory is usually something like "/home/username".

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

2. ls — Use the **"ls"** command to know what files are in the directory you are in. You can see all the hidden files by using the command **“ls -a”**.

```
nayso@Alok-Aspire:~$ ls
Desktop             itsuserguide.desktop  reset-setting
Documents           Music                 School_Resou
Downloads           Pictures              Students_Worl
examples.desktop    Public                Templates
GplatesProject      Qgis Projects         TuxPaint-Pict
```

3. cd — Use the **"cd"** command to go to a directory. For example, if you are in the home folder, and you want to go to the downloads folder, then you can type in **“cd Downloads”**. Remember, this command is case sensitive, and you have to type in the name of the folder exactly as it is. But there is a problem with these commands. Imagine you have a folder named “Raspberry Pi”. In this case, when you type in **“cd Raspberry Pi”**, the shell will take the second argument of the command as a different one, so you will get an error saying that the

directory does not exist. Here, you can use a backward slash. That is, you can use **“cd Raspberry\ Pi”** in this case. Spaces are denoted like this: If you just type **“cd”** and press enter, it takes you to the home directory. To go back from a folder to the folder before that, you can type “**cd** ..” . The two dots represent back.

```
nayso@Alok-Aspire:~$ cd Downloads
nayso@Alok-Aspire:~/Downloads$ cd
nayso@Alok-Aspire:~$ cd Raspberry\ Pi
nayso@Alok-Aspire:~/Raspberry Pi$ cd ..
nayso@Alok-Aspire:~$
```

4. mkdir & rmdir — Use the **mkdir** command when you need to create a folder or a directory. For example, if you want to make a directory called “DIY”, then you can type **“mkdir DIY**”. Remember, as told before, if you want to create a directory named “DIY Hacking”, then you can type “mkdir **DIY\ Hacking**”. Use **rmdir** to delete a directory. But **rmdir** can only be used to delete an empty directory. To delete a directory containing files, use **rm**.

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ mkdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
DIY
nayso@Alok-Aspire:~/Desktop$ rmdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

5. rm - Use the **rm** command to delete files and directories. Use "**rm -r**" to delete just the directory. It deletes both the folder and the files it contains when using only the **rm** command.

```
nayso@Alok-Aspire:~/Desktop$ ls
newer.py  New Folder
nayso@Alok-Aspire:~/Desktop$ rm newer.py
nayso@Alok-Aspire:~/Desktop$ ls
New Folder
nayso@Alok-Aspire:~/Desktop$ rm -r New\ Folder
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

6. touch — The **touch** command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, "**touch new.txt**".

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ touch new.txt
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
```

7. man & --help — To know more about a command and how to use it, use the **man** command. It shows the manual pages of the command. For example, "**man cd**" shows the manual pages of the **cd** command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., **cd –help**).

```
TOUCH(1)                          User Commands

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times o
       time.

       A FILE argument that does not exist is create
       is supplied.

       A  FILE  argument  string of - is handled spe
       change the times of the file associated with

       Mandatory arguments to long options are  mand
       too.

       -a     change only the access time

 Manual page touch(1) line 1 (press h for help or q
```

8. cp — Use the **cp** command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

```
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ cp new.txt /home/nayso/M
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
new.txt
```

9. mv — Use the **mv** command to move files through the command line. We can also use the **mv** command to rename a file. For example, if we want to rename the file "**text**" to "**new**", we can use "**mv text new**". It takes the two arguments, just like the **cp** command.

```
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
nayso@Alok-Aspire:~/Desktop$ mv new.txt newer.txt
nayso@Alok-Aspire:~/Desktop$ ls
newer.txt
```

10. locate — The **locate** command is used to locate a file in a Linux system, just like the search command in Windows. This command is useful when you don't know where a file is saved or the actual name of the file. Using the -i argument with the command helps to ignore the case (it doesn't matter if it is uppercase or lowercase). So, if you want a file that has the word "hello", it gives the list of all the files in your Linux system containing the word "hello" when you type in "**locate -i hello**". If you remember two words, you can separate them using an asterisk (*). For example, to locate a file containing the words "hello" and "this", you can use the command "**locate -i *hello*this".**

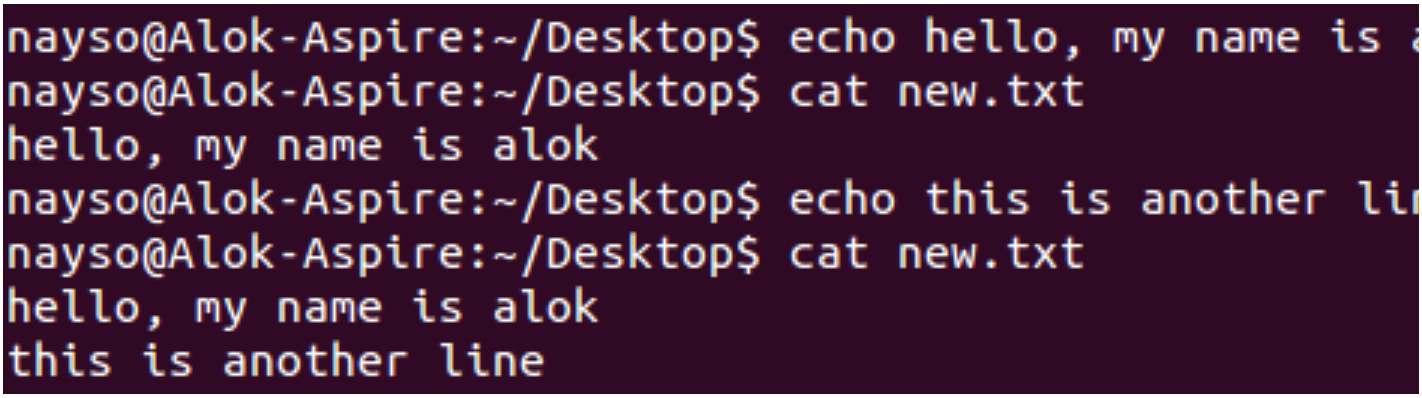
```
nayso@Alok-Aspire:~$ locate newer.txt
/home/nayso/Desktop/newer.txt
nayso@Alok-Aspire:~$ locate *DIY*Hacking*
/home/nayso/DIY Hacking
```

Intermediate Commands

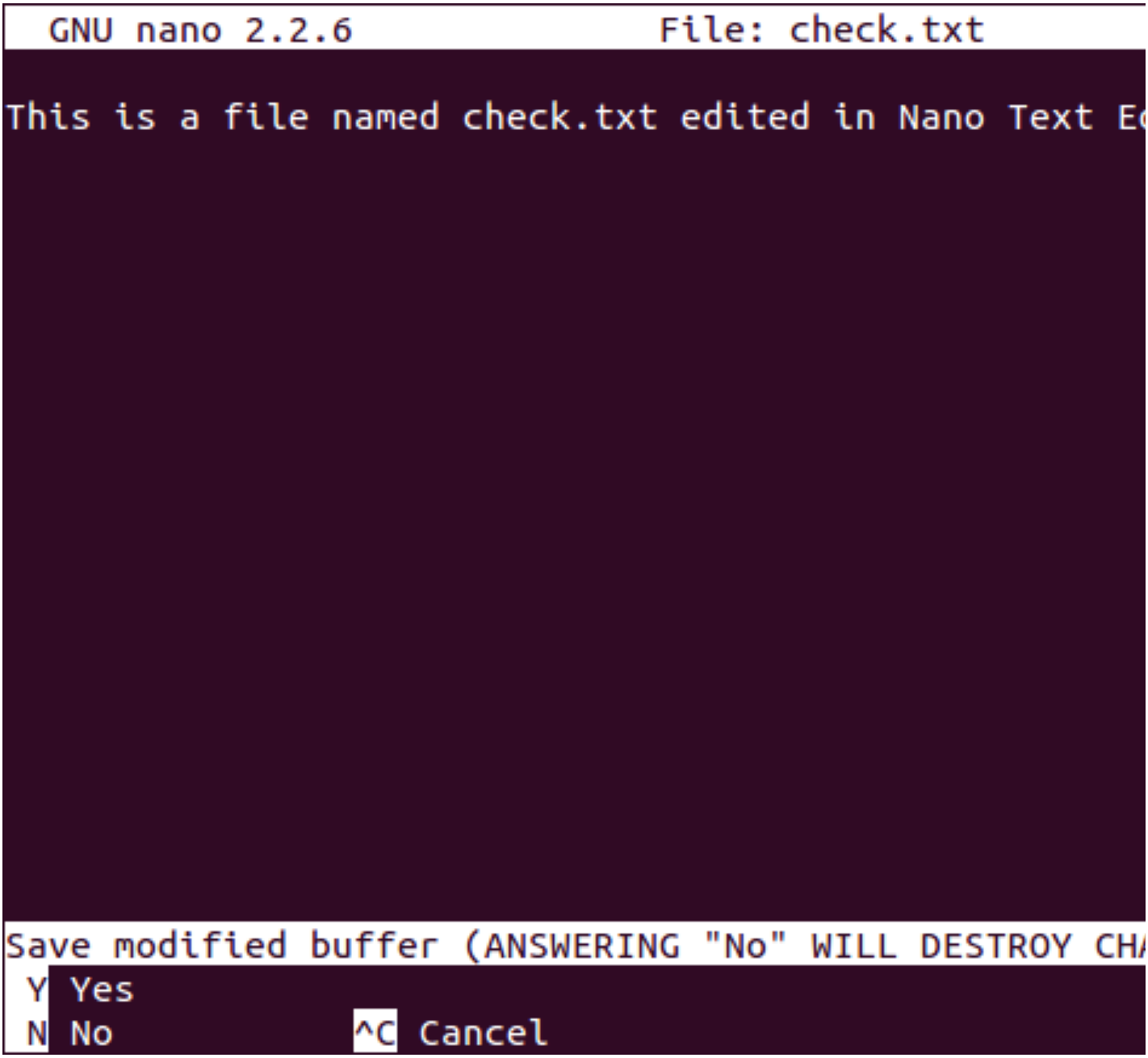
1. echo — The "**echo**" command helps us move some data, usually text into a file. For example, if you want to create a new text file or add to an already made text file, you just need to type in, "**echo hello, my name is alok >> new.txt**". You do not need to separate the spaces by using the backward slash here, because we put in two triangular

brackets when we finish what we need to write.

2. cat — Use the **cat** command to display the contents of a file. It is usually used to easily view programs.



3. nano, vi, jed — nano and **vi** are already installed text editors in the Linux command line. The **nano** command is a good text editor that denotes keywords with color and can recognize most languages. And **vi** is simpler than **nano**. You can create a new file or modify a file using this editor. For example, if you need to make a new file named **"check.txt"**, you can create it by using the command "**nano check.txt**". You can save your files after editing by using the sequence Ctrl+X, then Y (or N for no). In my experience, using **nano** for HTML editing doesn't seem as good, because of its color, so I recommend **jed** text editor. We will come to installing packages soon.



4. sudo — A widely used command in the Linux command line, **sudo** stands for "SuperUser Do". So, if you want any command to be done with administrative or root privileges, you can use the **sudo** command. For example, if you want to edit a file like **viz. alsa-base.conf**, which needs root permissions, you can use the command – **sudo nano alsa-base.conf**. You can enter the root command line using the command "**sudo bash**", then type in your user password. You can also use the command "**su**" to do this, but you need to set a root password before that. For that, you can use the command "**sudo passwd**"(not misspelled,

it is **passwd**). Then type in the new root password.

```
nayso@Alok-Aspire:~/Desktop$ sudo passwd
[sudo] password for nayso:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
nayso@Alok-Aspire:~/Desktop$ su
Password:
root@Alok-Aspire:/home/nayso/Desktop#
```

5. df — Use the **df** command to see the available disk space in each of the partitions in your system. You can just type in **df** in the command line and you can see each mounted partition and their used/available space in % and in KBs. If you want it shown in megabytes, you can use the command "**df -m**".

```
root@Alok-Aspire:/home/nayso/Desktop# df -m
Filesystem     1M-blocks  Used Available Use% Mounted
udev                 940     1       940   1% /dev
tmpfs                191     2       189   1% /run
/dev/sda5          96398 23466     68013  26% /
none                   1     0         1   0% /sys/fs
none                   5     0         5   0% /run/lo
none                 951     1       950   1% /run/sh
none                 100     1       100   1% /run/us
```

6. du — Use **du** to know the disk usage of a file in your system. If you want to know the disk usage for a particular folder or file in Linux, you can type in the command **df** and the name of the folder or file. For example, if you want to know the disk space used by the documents folder in Linux, you can use the command "**du Documents**". You can also use the command "**ls -lah**" to view the file sizes of all the files in a folder.

```
nayso@Alok-Aspire:~$ du Documents
516     Documents/DIYHacking
548     Documents
```

7. tar — Use **tar** to work with tarballs (or files compressed in a tarball archive) in the Linux command line. It has a long list of uses. It can be used to compress and uncompress different types of tar archives like **.tar, .tar.gz, .tar.bz2**,etc. It works on the basis of the arguments given to it. For example, "**tar -cvf**" for creating a **.tar** archive, -**xvf** to untar a tar archive, -**tvf** to list the contents of the archive, etc. Since it is a wide topic, here are some examples of tar commands.

8. zip, unzip — Use **zip** to compress files into a zip archive, and **unzip** to extract files from a zip archive.

9. uname — Use **uname** to show the information about the system your Linux distro is running. Using the command "**uname -a**" prints most of the information about the system. This prints the kernel release date, version, processor type, etc.

```
nayso@Alok-Aspire:~$ uname -a
Linux Alok-Aspire 4.4.0-22-generic #40~14.04.1-Ubuntu
C 2016 i686 i686 i686 GNU/Linux
```

10. apt-get — Use **apt** to work with packages in the Linux command line. Use **apt-get** to install packages. This requires root privileges, so use the **sudo** command with it. For example, if you want to install the text editor **jed** (as I mentioned earlier), we can type in the command "**sudo apt-get install jed**". Similarly, any packages can be installed like this. It is good to update your repository each time you try to install a new package. You can do that by typing "**sudo apt-get update**". You can upgrade the system by typing "**sudo apt-get upgrade**". We can also upgrade the distro by typing "**sudo apt-get dist-upgrade**". The command "**apt-cache search**" is used to search for a package. If you

want to search for one, you can type in "**apt-cache search jed**"(this doesn't require root).

```
nayso@Alok-Aspire:~$ sudo apt-get install jed
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  jed-common libslang2-modules slsh
Suggested packages:
  gpm
The following NEW packages will be installed:
  jed jed-common libslang2-modules slsh
0 upgraded, 4 newly installed, 0 to remove and 419 n
Need to get 810 kB of archives.
After this operation, 2,992 kB of additional disk sp
Do you want to continue? [Y/n]
```

11. chmod — Use **chmod** to make a file executable and to change the permissions granted to it in Linux. Imagine you have a python code named **numbers.py** in your computer. You'll need to run "**python numbers.py**" every time you need to run it. Instead of that, when you make it executable, you'll just need to run "**numbers.py**" in the terminal to run the file. To make a file executable, you can use the command "**chmod +x numbers.py**" in this case. You can use "**chmod 755 numbers.py**" to give it root permissions or "**sudo chmod +x numbers.py**" for root executable. Here is some more information about the chmod command.

```
nayso@Alok-Aspire:~/Desktop$ ls
numbers.py
nayso@Alok-Aspire:~/Desktop$ chmod +x numbers.py
nayso@Alok-Aspire:~/Desktop$ ls
numbers.py
```

12. hostname — Use **hostname** to know your name in your host or network. Basically, it displays your hostname and IP address. Just typing "**hostname**" gives the output. Typing in "**hostname -I**" gives you your IP address in your network.

```
nayso@Alok-Aspire:~/Desktop$ hostname
Alok-Aspire
nayso@Alok-Aspire:~/Desktop$ hostname -I
192.168.1.36
```

13. ping — Use **ping** to check your connection to a server. Wikipedia says, "**Ping** is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network". Simply, when you type in, for example, "**ping google.com**", it checks if it can connect to the server and come back. It measures this round-trip time and gives you the details about it. The use of this command for simple users like us is to check your internet connection. If it pings the Google server (in this case), you can confirm that your internet connection is active!

```
nayso@Alok-Aspire:~/Desktop$ ping google.com
PING google.com (172.217.26.206) 56(84) bytes of data
64 bytes from google.com (172.217.26.206): icmp_seq=1
64 bytes from google.com (172.217.26.206): icmp_seq=2
64 bytes from google.com (172.217.26.206): icmp_seq=3
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, ti
rtt min/avg/max/mdev = 47.959/49.388/51.299/1.417 ms
```

Tips and Tricks for Using Linux Command Line

- You can use the **clear** command to clear the terminal if it gets filled up with too many commands.
- **TAB** can be used to fill up in terminal. For example, You just need to type "**cd Doc**" and then **TAB** and the terminal fills the rest up and makes it "**cd Documents**".
- **Ctrl+C** can be used to stop any command in terminal safely. If it doesn't stop with that, then **Ctrl+Z** can be used to force stop it.
- You can exit from the terminal by using the **exit** command.
- You can power off or reboot the computer by using the command **sudo halt** and **sudo reboot**.

Once you've mastered the Linux commands for beginners, you can move onto these Useful Intermediate Linux Commands.

Alok Naushad
I love working with a Raspberry Pi and Arduino! A Linux Programmer, also an Android Developer. Love Making things!
Author

Comments (61)

Unix/Linux Command Reference

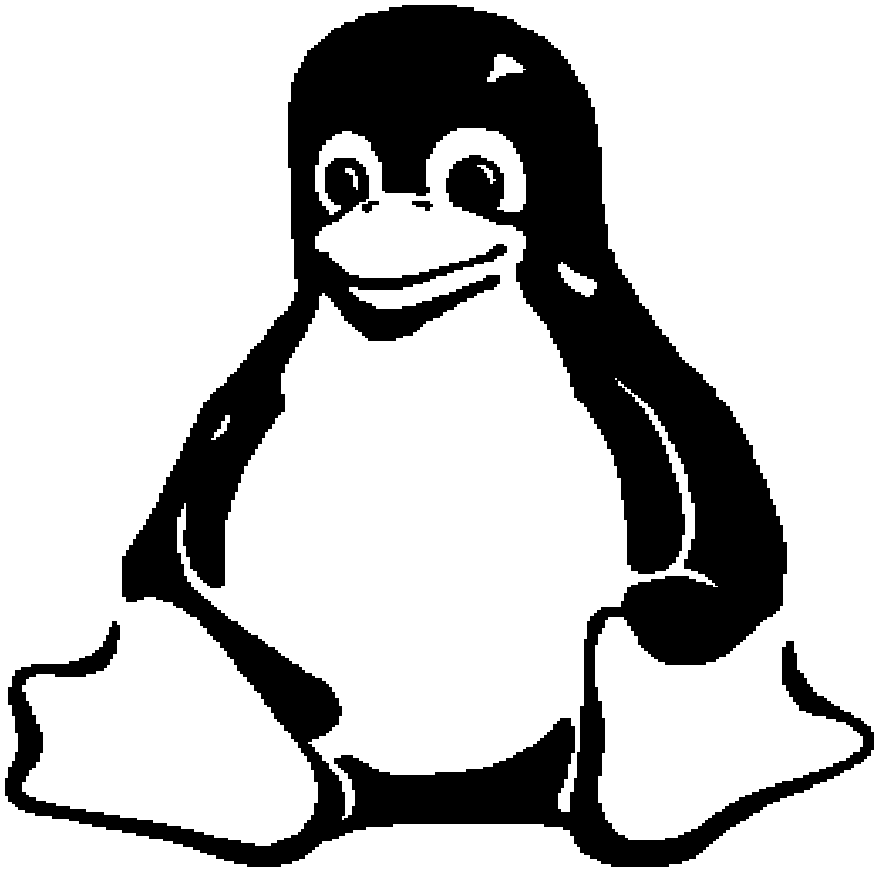
File Commands		
1.	ls	Directory listing
2.	ls -al	Formatted listing with hidden files
3.	ls -lt	Sorting the Formatted listing by time modification
4.	cd dir	Change directory to dir
5.	cd	Change to home directory
6.	pwd	Show current working directory
7.	mkdir dir	Creating a directory dir
8.	cat >file	Places the standard input into the file
9.	more file	Output the contents of the file
10.	head file	Output the first 10 lines of the file
11.	tail file	Output the last 10 lines of the file
12.	tail -f file	Output the contents of file as it grows,starting with the last 10 lines
13.	touch file	Create or update file
14.	rm file	Deleting the file
15.	rm -r dir	Deleting the directory
16.	rm -f file	Force to remove the file
17.	rm -rf dir	Force to remove the directory dir
18.	cp file1 file2	Copy the contents of file1 to file2
19.	cp -r dir1 dir2	Copy dir1 to dir2;create dir2 if not present
20.	mv file1 file2	Rename or move file1 to file2,if file2 is an existing directory
21.	ln -s file link	Create symbolic link link to file
Process management		
1.	ps	To display the currently working processes
2.	top	Display all running process

3.	kill pid	Kill the process with given pid
4.	killall proc	Kill all the process named proc
5.	pkill pattern	Will kill all processes matching the pattern
6.	bg	List stopped or background jobs,resume a stopped job in the background
7.	fg	Brings the most recent job to foreground
8.	fg n	Brings job n to the foreground
File permission		
1.	chmod octal file	Change the permission of file to octal,which can be found separately for user,group,world by adding, • 4-read(r) • 2-write(w) • 1-execute(x)
Searching		
1.	grep pattern file	Search for pattern in file
2.	grep -r pattern dir	Search recursively for pattern in dir
3.	command | grep pattern	Search pattern in the output of a command
4.	locate file	Find all instances of file
5.	find . -name filename	Searches in the current directory (represented by a period) and below it, for files and directories with names starting with filename
6.	pgrep pattern	Searches for all the named processes , that matches with the pattern and, by default, returns their ID
System Info		
1.	date	Show the current date and time
2.	cal	Show this month's calender
3.	uptime	Show current uptime
4.	w	Display who is on line
5.	whoami	Who you are logged in as

6.	finger user	Display information about user
7.	uname -a	Show kernel information
8.	cat /proc/cpuinfo	Cpu information
9.	cat proc/meminfo	Memory information
10.	man command	Show the manual for command
11.	df	Show the disk usage
12.	du	Show directory space usage
13.	free	Show memory and swap usage
14.	whereis app	Show possible locations of app
15.	which app	Show which applications will be run by default
Compression		
1.	tar cf file.tar file	Create tar named file.tar containing file
2.	tar xf file.tar	Extract the files from file.tar
3.	tar czf file.tar.gz files	Create a tar with Gzip compression
4.	tar xzf file.tar.gz	Extract a tar using Gzip
5.	tar cjf file.tar.bz2	Create tar with Bzip2 compression
6.	tar xjf file.tar.bz2	Extract a tar using Bzip2
7.	gzip file	Compresses file and renames it to file.gz
8.	gzip -d file.gz	Decompresses file.gz back to file
Network		
1.	ping host	Ping host and output results
2.	whois domain	Get whois information for domains
3.	dig domain	Get DNS information for domain
4.	dig -x host	Reverse lookup host
5.	wget file	Download file
6.	wget -c file	Continue a stopped download

Shortcuts		
1.	ctrl+c	Halts the current command
2.	ctrl+z	Stops the current command, resume with fg in the foreground or bg in the background
3.	ctrl+d	Logout the current session, similar to exit
4.	ctrl+w	Erases one word in the current line
5.	ctrl+u	Erases the whole line
6.	ctrl+r	Type to bring up a recent command
7.	!!	Repeats the last command
8.	exit	Logout the current session

An Introduction to the Linux Command Shell For Beginners



Presented by:

Victor Gedris

In Co-Operation With:

The Ottawa Canada Linux Users Group

and

ExitCertified

Copyright and Redistribution

This manual was written with the intention of being a helpful guide to Linux users who are trying to become familiar with the Bash shell and basic Linux commands. To make this manual useful to the widest range of people, I decided to release it under a free documentation license, with the hopes that people benefit from it by updating it and re-distributing modified copies. You have permission to modify and distribute this document, as specified under the terms of the GNU Free Documentation License. Comments and suggestions for improvement may be directed to: vic@gedris.org.

This document was created using an Open Source office application called *Open Office*. The file format is non-proprietary, and the document is also published in various other formats online. Updated copies will be available on Vic Gedris' web site [http://vic.dyndns.org/]. For more information on Open Office, please visit http://www.openoffice.org/.

Document Version: 1.2, 2003-06-25

1.0 Introduction

The purpose of this document is to provide the reader with a fast and simple introduction to using the Linux command shell and some of its basic utilities. It is assumed that the reader has zero or very limited exposure to the Linux command prompt. This document is designed to accompany an instructor-led tutorial on this subject, and therefore some details have been left out. Explanations, practical examples, and references to DOS commands are made, where appropriate.

1.1 What is a command shell?

- A program that interprets commands
- Allows a user to execute commands by typing them manually at a terminal, or automatically in programs called *shell scripts*.
- A shell is *not* an operating system. It is a way to interface with the operating system and run commands.

1.2 What is BASH?

- BASH = **B**ourne **A**gain **SH**ell
- Bash is a shell written as a free replacement to the standard Bourne Shell (`/bin/sh`) originally written by Steve Bourne for UNIX systems.
- It has all of the features of the original Bourne Shell, plus additions that make it easier to program with and use from the command line.
- Since it is Free Software, it has been adopted as the default shell on most Linux systems.

1.3 How is BASH different from the DOS command prompt?

- **Case Sensitivity:** In Linux/UNIX, commands and filenames are case sensitive, meaning that typing "`EXIT`" instead of the proper "`exit`" is a mistake.
- **"\" vs. "/":** In DOS, the forward-slash "`/`" is the command argument delimiter, while the backslash "`\`" is a directory separator. In Linux/UNIX, the "`/`" is the directory separator, and the "`\`" is an escape character. More about these special characters in a minute!
- **Filenames:** The DOS world uses the "eight dot three" filename convention, meaning that all files followed a format that allowed up to 8 characters in the filename, followed by a period ("dot"), followed by an option extension, up to 3 characters long (e.g. `FILENAME.TXT`). In UNIX/Linux, there is no such thing as a file extension. Periods can be placed at any part of the filename, and "extensions" may be interpreted differently by all programs, or not at all.

1.4 Special Characters

Before we continue to learn about Linux shell commands, it is important to know that there are many symbols and characters that the shell interprets in special ways. This means that certain typed characters: a) cannot be used in certain situations, b) may be used to perform special operations, or, c) must be "escaped" if you want to use them in a normal way.

Character	*Description*
****	Escape character. If you want to reference a special character, you must "escape" it with a backslash first. Example: `touch /tmp/filename\*`
/	Directory separator, used to separate a string of directory names. Example: `/usr/src/linux`
.	Current directory. Can also "hide" files when it is the first character in a filename.
..	Parent directory
~	User's home directory
*****	Represents 0 or more characters in a filename, or by itself, all files in a directory. Example: `pic*2002` can represent the files `pic2002`, `picJanuary2002`, `picFeb292002`, etc.
?	Represents a single character in a filename. Example: `hello?.txt` can represent `hello1.txt`, `helloz.txt`, but not `hello22.txt`
[ ]	Can be used to represent a range of values, e.g. `[0-9]`, `[A-Z]`, etc. Example: `hello[0-2].txt` represents the names `hello0.txt`, `hello1.txt`, and `hello2.txt`
|	"Pipe". Redirect the output of one command into another command. Example: `ls | more`
>	Redirect output of a command into a new file. If the file already exists, over-write it. Example: `ls > myfiles.txt`
>>	Redirect the output of a command onto the end of an existing file. Example: `echo "Mary 555-1234" >> phonenumbers.txt`
<	Redirect a file as input to a program. Example: `more < phonenumbers.txt`
;	Command separator. Allows you to execute multiple commands on a single line. Example: `cd /var/log ; less messages`
&&	Command separator as above, but only runs the second command if the first one finished without errors. Example: `cd /var/logs && less messages`
&	Execute a command in the background, and immediately get your shell back. Example: `find / -name core > /tmp/corefiles.txt &`

1.5 Executing Commands

The Command `PATH`:

- Most common commands are located in your shell's "`PATH`", meaning that you can just type the name of the program to execute it.

  Example: Typing " `ls`" will execute the " `ls`" command.

- Your shell's "`PATH`" variable includes the most common program locations, such as `/bin`, `/usr/bin`, `/usr/X11R6/bin`, and others.

- To execute commands that are not in your current PATH, you have to give the complete location of the command.

  Examples: `/home/bob/myprogram`

  `./program` (Execute a program in the current directory)

  `~/bin/program` (Execute program from a personal `bin` directory)

Command Syntax

- Commands can be run by themselves, or you can pass in additional arguments to make them do different things. Typical command syntax can look something like this:

  ```
  command [-argument] [-argument] [--argument] [file]
  ```

- Examples:

`ls`	List files in current directory
`ls -l`	Lists files in "long" format
`ls -l --color`	As above, with colourized output
`cat filename`	Show contents of a file
`cat -n filename`	Show contents of a file, with line numbers

2.0 Getting Help

When you're stuck and need help with a Linux command, help is usually only a few keystrokes away! Help on most Linux commands is typically built right into the commands themselves, available through online help programs ("man pages" and "info pages"), and of course online.

2.1 Using a Command's Built-In Help

Many commands have simple "help" screens that can be invoked with special command flags. These flags usually look like "`-h`" or "`--help`".

Example: `grep --help`

2.2 Online Manuals: "Man Pages"

The best source of information for most commands can be found in the online manual pages, known as "man pages" for short. To read a command's man page, type "`man command`".

Examples: `man ls` Get help on the "ls" command.

`man man` A manual about how to use the manual!

To search for a particular word within a man page, type "`/word`". To quit from a man page, just type the "Q" key.

Sometimes, you might not remember the name of Linux command and you need to search for it. For example, if you want to know how to change a file's permissions, you can search the man page descriptions for the word "permission" like this:

```
man -k permission
```

If you look at the output of this command, you will find a line that looks something like:

```
chmod                (1)  - change file access permissions
```

Now you know that "`chmod`" is the command you were looking for. Typing "`man chmod`" will show you the `chmod` command's manual page!

2.3 Info Pages

Some programs, particularly those released by the Free Software Foundation, use info pages as their main source of online documentation. Info pages are similar to man page, but instead of being displayed on one long scrolling screen, they are presented in shorter segments with links to other pieces of information. Info pages are accessed with the "`info`" command, or on some Linux distributions, "`pinfo`" (a nicer info browser).

For example: `info df` Loads the "`df`" info page.

3.0 Navigating the Linux Filesystem

The Linux filesystem is a tree-like hierarchy hierarchy of directories and files. At the base of the filesystem is the "`/`" directory, otherwise known as the "root" (not to be confused with the root user). Unlike DOS or Windows filesystems that have multiple "roots", one for each disk drive, the Linux filesystem mounts all disks somewhere underneath the `/` filesystem. The following table describes many of the most common Linux directories.

3.1 The Linux Directory Layout

Directory	*Description*
	The nameless base of the filesystem. All other directories, files, drives, and devices are attached to this root. Commonly (but incorrectly) referred to as the "slash" or "/" directory. The "/" is just a directory separator, not a directory itself.
`/bin`	Essential command binaries (programs) are stored here (`bash`, `ls`, `mount`, `tar`, etc.)
`/boot`	Static files of the boot loader.
`/dev`	Device files. In Linux, hardware devices are acceessd just like other files, and they are kept under this directory.
`/etc`	Host-specific system configuration files.
`/home`	Location of users' personal home directories (e.g. `/home/susan`).
`/lib`	Essential shared libraries and kernel modules.
`/proc`	Process information pseudo-filesystem. An interface to kernel data structures.
`/root`	The root (superuser) home directory.
`/sbin`	Essential system binaries (`fdisk`, `fsck`, `init`, etc).
`/tmp`	Temporary files. All users have permission to place temporary files here.
`/usr`	The base directory for most shareable, read-only data (programs, libraries, documentation, and much more).
`/usr/bin`	Most user programs are kept here (`cc`, `find`, `du`, etc.).
`/usr/include`	Header files for compiling C programs.
`/usr/lib`	Libraries for most binary programs.
`/usr/local`	"Locally" installed files. This directory only really matters in environments where files are stored on the network. Locally-installed files go in `/usr/local/bin`, `/usr/local/lib`, etc.). Also often used for software packages installed from source, or software not officially shipped with the distribution.
`/usr/sbin`	Non-vital system binaries (`lpd`, `useradd`, etc.)
`/usr/share`	Architecture-independent data (icons, backgrounds, documentation, terminfo, man pages, etc.).
`/usr/src`	Program source code. E.g. The Linux Kernel, source RPMs, etc.
`/usr/X11R6`	The X Window System.
`/var`	Variable data: mail and printer spools, log files, lock files, etc.

3.2 Commands for Navigating the Linux Filesystems

The first thing you usually want to do when learning about the Linux filesystem is take some time to look around and see what's there! These next few commands will: a) Tell you where you are, b) take you somewhere else, and c) show you what's there. The following table describes the basic operation of the `pwd`, `cd`, and `ls` commands, and compares them to certain DOS commands that you might already be familiar with.

Linux Command	*DOS Command*	*Description*
`pwd`	`cd`	"Print Working Directory". Shows the current location in the directory tree.
`cd`	`cd, chdir`	"Change Directory". When typed all by itself, it returns you to your home directory.
`cd directory`	`cd directory`	Change into the specified directory name. Example: `cd /usr/src/linux`
`cd ~`		"~" is an alias for your home directory. It can be used as a shortcut to your "home", or other directories relative to your home.
`cd ..`	`cd..`	Move up one directory. For example, if you are in `/home/vic` and you type "`cd ..`", you will end up in `/home`.
`cd -`		Return to previous directory. An easy way to get back to your previous location!
`ls`	`dir /w`	List all files in the current directory, in column format.
`ls directory`	`dir directory`	List the files in the specified directory. Example: `ls /var/log`
`ls -l`	`dir`	List files in "long" format, one file per line. This also shows you additional info about the file, such as ownership, permissions, date, and size.
`ls -a`	`dir /a`	List all files, including "hidden" files. Hidden files are those files that begin with a ".", e.g. The `.bash_history` file in your home directory.

`ls -ld directory`		A "long" list of "directory", but instead of showing the directory contents, show the directory's detailed information. For example, compare the output of the following two commands:
		`ls -l /usr/bin`
		`ls -ld /usr/bin`

`ls /usr/bin/d*`	`dir d*.*`	List all files whose names begin with the letter "d" in the `/usr/bin` directory.

4.0 Piping and Re-Direction

Before we move on to learning even more commands, let's side-track to the topics of piping and re-direction. The basic UNIX philosophy, therefore by extension the Linux philosophy, is to have many small programs and utilities that do a particular job very well. It is the responsibility of the programmer or user to combine these utilities to make more useful command sequences.

4.1 Piping Commands Together

The pipe character, "|", is used to chain two or more commands together. The output of the first command is "piped" into the next program, and if there is a second pipe, the output is sent to the third program, etc. For example:

```
ls -la /usr/bin | less
```

In this example, we run the command "`ls -la /usr/bin`", which gives us a long listing of all of the files in `/usr/bin`. Because the output of this command is typically very long, we pipe the output to a program called "`less`", which displays the output for us one screen at a time.

4.2 Redirecting Program Output to Files

There are times when it is useful to save the output of a command to a file, instead of displaying it to the screen. For example, if we want to create a file that lists all of the MP3 files in a directory, we can do something like this, using the ">" redirection character:

```
ls -l /home/vic/MP3/*.mp3 > mp3files.txt
```

A similar command can be written so that instead of creating a new file called `mp3files.txt`, we can append to the end of the original file:

```
ls -l /home/vic/extraMP3s/*.mp3 >> mp3files.txt
```

5.0 Other Linux Commands

The following sections describe many other commands that you will find on most Linux systems. I can't possibly cover the details of all of these commands in this document, so don't forget that you can check the "man pages" for additional information. Not all of the listed commands will be available on all Linux or UNIX distributions.

5.1 Working With Files and Directories

These commands can be used to: find out information about files, display files, and manipulate them in other ways (copy, move, delete).

Linux Command	***DOS Command***	***Description***
`file`		Find out what kind of file it is. For example, "`file /bin/ls`" tells us that it is a Linux executable file.
`cat`	`type`	Display the contents of a text file on the screen. For example: `cat mp3files.txt` would display the file we created in the previous section.
`head`		Display the first few lines of a text file. Example: `head /etc/services`
`tail`		Display the last few lines of a text file. Example: `tail /etc/services`
`tail -f`		Display the last few lines of a text file, and then output appended data as the file grows (very useful for following log files!). Example: `tail -f /var/log/messages`
`cp`	`copy`	Copies a file from one location to another. Example: `cp mp3files.txt /tmp` (copies the mp3files.txt file to the /tmp directory)
`mv`	`rename, ren, move`	Moves a file to a new location, or renames it. For example: `mv mp3files.txt /tmp` (copy the file to /tmp, and delete it from the original location)
`rm`	`del`	Delete a file. Example: `rm /tmp/mp3files.txt`
`mkdir`	`md`	Make Directory. Example: `mkdir /tmp/myfiles/`
`rmdir`	`rd, rmdir`	Remove Directory. Example: `rmdir /tmp/myfiles/`

5.2 Finding Things

The following commands are used to find files. "`ls`" is good for finding files if you already know approximately where they are, but sometimes you need more powerful tools such as these:

Linux Command	*Description*
`which`	Shows the full path of shell commands found in your path. For example, if you want to know exactly where the "grep" command is located on the filesystem, you can type "`which grep`". The output should be something like: `/bin/grep`
`whereis`	Locates the program, source code, and manual page for a command (if all information is available). For example, to find out where "`ls`" and its man page are, type: "`whereis ls`" The output will look something like: `ls: /bin/ls /usr/share/man/man1/ls.1.gz`
`locate`	A quick way to search for files anywhere on the filesystem. For example, you can find all files and directories that contain the name "mozilla" by typing: `locate mozilla`
`find`	A *very* powerful command, but sometimes tricky to use. It can be used to search for files matching certain patterns, as well as many other types of searches. A simple example is: `find . -name \*mp3` This example starts searching in the current directory "." and all sub-directories, looking for files with "mp3" at the end of their names.

5.3 Informational Commands

The following commands are used to find out some information about the user or the system.

Linux Command	*Explanation*
`ps`	Lists currently running process (programs).
`w`	Show who is logged on and what they are doing.
`id`	Print your user-id and group id's
`df`	Report filesystem disk space usage ("Disk Free" is how I remember it)
`du`	Disk Usage in a particular directory. "`du -s`" provides a summary for the current directory.
`top`	Displays CPU processes in a full-screen GUI. A great way to see the activity on your computer in real-time. Type "Q" to quit.
`free`	Displays amount of free and used memory in the system.
`cat /proc/cpuinfo`	Displays information about your CPU.
`cat /proc/meminfo`	Display lots of information about current memory usage.
`uname -a`	Prints system information to the screen (kernel version, machine type, etc.)

5.4 Other Utilities

Here are some other commands that are useful to know.

Linux Command	*Description*
clear	Clear the screen
echo	Display text on the screen. Mostly useful when writing shell scripts. For example: echo “Hello World”

more	Display a file, or program output one page at a time. Examples:
	`more mp3files.txt`
	`ls –la | more`

less	An improved replacement for the “`more`” command. Allows you to scroll backwards as well as forwards.
grep	Search for a pattern in a file or program output. For example, to find out which TCP network port is used by the “nfs” service, you can do this: `grep “nfs” /etc/services` This looks for any line that contains the string “nfs” in the file “/etc/services” and displays only those lines.
lpr	Print a file or program output. Examples: `lpr mp3files.txt` - Print the mp3files.txt file `ls –la | lpr` - Print the output of the “`ls –la`” command.
sort	Sort a file or program output. Example: `sort mp3files.txt`
su	“Switch User”. Allows you to switch to another user's account temporarily. The default account to switch to is the root/superuser account. Examples: `su` - Switch the root account `su –` - Switch to root, and log in with root's environment `su larry` - Switch to Larry's account

5.5 Shortcuts to Make it all Easier!

When you start using the Bash shell more often, you will appreciate these shortcuts that can save you very much typing time.

Shortcut	*Description*
Up/Down Arrow Keys	Scroll through your most recent commands. You can scroll back to an old command, hit ENTER, and execute the command without having to re-type it.
"`history`" command	Show your complete command history.
TAB Completion	If you type a partial command or filename that the shell recognizes, you can have it automatically completed for you if you press the TAB key. Try typing the first few characters of your favourite Linux command, then hit TAB a couple of times to see what happens.

Complete recent commands with "!"	Try this: Type "`!`" followed by the first couple of letters of a recent command and press ENTER! For example, type:
	`find /usr/bin -type f -name m\*`
	...and now type:
	`!fi`

Search your command history with `CTRL-R`	Press `CTRL-R` and then type any portion of a recent command. It will search the commands for you, and once you find the command you want, just press ENTER.
Scrolling the screen with `Shift-PageUp` and `Page Down`	Scroll back and forward through your terminal.

6.0 Further Reading

Link Address	*Description*
http://www.oclug.on.ca	Ottawa Canada Linux Users Group. A group with an active mailing list, monthly meetings, and much more.
http://www.exitcertified.com	Ottawa's source for Sun training, and the host of OCLUG's technology seminars.
http://www.fsf.org	The Free Software Foundation. Documentation, source code, and much more for many programs commonly found on Linux systems.
http://linux.org.mt/article/terminal	"A Beginner's Bash". Another very good introduction to Bash.
http://www.oreilly.com/catalog/bash2	An excellent book if you want to learn how to customize Bash and use it for shell script programming.