Linux Fundamentals : console commands and vi editor

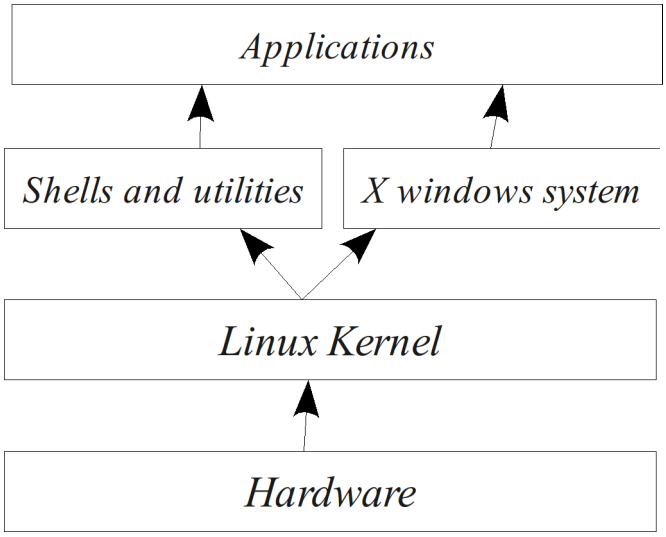
Objective:

1. Understanding the Linux computing environment – Design and Organization
2. Using Linux bash console commands:
3. Using the vi editor

Description

Linux is a fast and stable open source operating system for personal computers (PCs) and workstations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. Linux was developed in the early 1990s by Linus Torvalds, along with other programmers around the world. As an operating system, Linux performs many of the same functions as Unix, Macintosh, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility, along with being freely available.

Structure of a Linux system



Kernel

- Forms the core of LINUX operating system. This interacts with the hardware and is loaded into the memory when the system is booted.
- Functions
  - Managing the system resources
  - Allocating time for different users and processes
  - Deciding process properties and performing them.

- It does not interact with user directly but through a shell

Shell

- Shell interacts with user and kernel. Some of its features :
  - Interactive processing: communication takes place on the basis of the commands provided to system.
  - Input/Output Redirection – programs can be instructed to take inputs and outputs from sources other than the default ones (keybord/Monitor) like any text files.
  - Pipes – combinig various commands to perform complex tasks using pipes, thus reducing the necessity to write new programs for thee operations.
  - Shell Scripts – sequence of commands to be formed with a single filename.

Filesystem

- Linux treats all information as files. These are stored in a hierarchical order and are grouped together. Enablinbg easier search files.
- Directory and devices are also treated as files. All the filesa re connected together stored as a data structure called as filesystem.
- Each directory contains files and other directories. The set of directories from root forms the path name. This apath name with the filename forms the absolute path of the file.

Input/Output Redirection

Linux treatsthe keyboard as Standard input file and monitor as Visual Display Unit as Standard Output file as well as standard error file. Taking input from sopurces other than the standard input or passing output to any other sources other than standard output is called as Redirection.

File Access Permissions (FAP)

There are various types of permissions available. They are read (r), write (w) and execute (x). Associated with any Linux file is the owner (u), group (g) users who need to share that file and others (o) not belonging to the group. ls -l optin gives the list of permissions granted to each file. Permissions are given from the second position in groups of 3 for user, group and others.

Octadecimal representation of FAP

Numbers	Values	Numbers	Values
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

- If permissionis set value allocated is 1 elese 0
  - rwx------ owner has value 111 and group and others have 000 hence octadecimal

value is 700.

- 777 is to grant all permissions.

vi Editor

- Stands for visual. It gives a complete screen view of the file.

Excercises

Exercise 1

1. Observe the hierarchy under the File-system
2. Mention the significance of each directory
3. Find out about different Linux distributions

Exercise 2

Prelimnary console Commands

- date for current date and time.
- who gives details of the users who have logged
- man documentation for commands
- head display the initial part of a test file
- tail display the last part of a text file

Directory Commands

- pwd dipslays full pathname for the current working directory
- ls displays list of files in the current working directory
- mkdir create a new directory
- rmdir remove a directory
- cd change directory

File commands

- cat lists the contents of specified file
- cp (copy) create duplicate copies of ordinary files
- ln (link) establishs an additional filename for same ordinary file
- mv rename and move ordinary and directory files
- rm removes one or more files from a directory. -r for recursive and -i for interactive
- read (used in shell scripts) command waits for the user to input the value of the variable
- echo display a message on the screen.

- wc count the number of lines, words or characters in a file.
- find locatye files in a directory and in a subdirectory

Redirection

Output

- command > file

Input

- command < file

Redirect both Input and Output

- command < source > destination

Wild card patterns

- * represents any number of characters when used in the prefix or in the suffix
- ? represnts single character only either in the prefix or in the suffix
- [] access a subset of related files

Changing File access permissions

- chmod change mode of file or directory
  - + to grant permission
  - - to revoke permission
    - chmod u-x air.c
    - chmod g+w air.c
  - chmod 777 air.c

Pipes

- cat text | head -3

vi Editor

General Startup

- To use vi: vi filename
- To exit vi and save changes: ZZ or :wq
- To exit vi without saving changes: :q!
- To enter vi command mode: *[esc]*

Cursor Movement

- h move left (backspace)
- j move down
- k move up
- l move right (spacebar)
- [Return] move to beginning of next line

Inserting

- i insert before cursor
- a append after cursor
- A append at the end of Line
- O open line above Crsor and enter append mode

Deleting

- x delete character under cursor
- dd delete line under cursor
- dw delete word under cursor
- db delete word before cursor

Find Commands

- ? finds a word going backwards
- / finds a word going forwards

Miscellaneous Commands

- . repeat last command
- u ndoes last issued command

Write File

- :w Saves the current file without quitting

Moving

- :# move to line #
- :$ move to last line of fille

1. Everything in Linux is a file including the hardware and even the directories.
2. **#** : Denotes the super(root) user
3. **$** : Denotes the normal user
4. **/root**: Denotes the super user's directory
5. **/home**: Denotes the normal user's directory.
6. **Switching between Terminals**
    1. **Ctrl + Alt + F1-F6**: Console login
    2. **Ctrl + Alt + F7**: GUI login
7. **The Magic Tab**: Instead of typing the whole filename if the unique pattern for a particular file is given then the remaining characters need not be typed and can be obtained automatically using the Tab button.
8. **~(Tilde)**: Denotes the current user's home directory
9. **Ctrl + Z**: To stop a command that is working interactively without terminating it.
10. **Ctrl + C**: To stop a command that is not responding. (Cancellation).
11. **Ctrl + D**: To send the EOF(End of File) signal to a command normally when you see '>'.
12. **Ctrl + W**: To erase the text you have entered a word at a time.
13. **Up arrow key**: To redisplay the last executed command. The Down arrow key can be used to print the next command used after using the Up arrow key previously.
14. The history command can be cleared using a simple option **–c** (clear).
15. **cd** : The cd command can be used trickily in the following ways:
    1. **cd** : To switch to the home user
    2. **cd *** : To change directory to the first file in the directory (only if the first file is a directory)
    3. **cd ..** : To move back a folder
    4. **cd -** : To return to the last directory you were in
16. Files starting with a dot (.) are a hidden file.
17. To view hidden files**: ls -a**
18. **ls**: The ls command can be use trickily in the following ways:
    1. **ls -lR** : To view a long list of all the files (which includes directories) and their subdirectories recursively .
    2. **ls *.*** : To view a list of all the files with extensions only.
19. **ls -ll**: Gives a long list in the following format
    1. **drwxr-xr-x 2 root root 4096 2010-04-29 05:17 bin** where

1. **drwxr-xr-x** : permission where d stands for directory, rwx stands for owner privilege, r-x stands for the group privilege and r-x stands for others permission respectively.

Here r stands for read, w for write and x for executable.

2=> link count

root=>owner

root=>group

4096=> directory size

2010-04-29=>date of creation

05:17=> time of creation

bin=>directory file(in blue)

The color code of the files is as follows:

Blue: Directory file

White: Normal file

Green: Executable file

Yellow: Device file

Magenta: Picture file

Cyan: link file

Red: Compressed file

File Symbol

-(Hyphen) : Normal file

d=directory

l=link file

b=Block device file

c=character device file

20. **Using the rm command**: When used without any option the rm command deletes the file or directory (option -**rf**) without any warning. A simple mistake like **rm / somedir** instead of **rm /somedir** can cause major chaos and delete the entire content of the /(root) directory. Hence it is always advisable to use rm command with the -**i**(which prompts before removal) option. Also there is no undelete option in Linux.
21. Copying hidden files: **cp .*** (copies hidden files only to a new destination)
22. **dpkg -l** : To get a list of all the installed packages.
23. **Use of ‘ > ‘ and ‘ >> ‘** : The ‘ > ‘ symbol (input redirector sign) can be used to add content to a file when used with the cat command. Whereas ‘ >> ‘ can be used to append to a file. If the ‘ >> ‘ symbol is not used and content is added to a file using only the ‘>’ symbol the previous

content of the file is deleted and replaced with the new content.

e.g: $ touch text (creates an empty file)

$ cat >text

This is text's text. (Save the changes to the file using Ctrl +D)

$cat >> text

This is a new text. (Ctrl + D)

Output of the file:

This is text's text.

This is a new text.

24. To count the number of users logged in : **who |wc –l**
25. cat: The cat command can be used to trickly in the following way:
    1. To count no. of lines from a file : **cat <filename> |wc -l**
    2. To count no. of words from a file : **cat <filename> |wc -w**
    3. To count no. of characters from a file : **cat <filename> |wc –c**
26. To search a term that returns a pattern: **cat <filename> |grep [pattern]**
27. **The 'tr' command**: Used to translate the characters of a file.
    1. **tr 'a-z' 'A-Z' <text >text1** : The command for example is used to translate all the characters from lower case to upper case of the 'text' file and save the changes to a new file 'text1′.
28. **File permission using chmod**: 'chmod' can be used directly to change the file permission of files in a simple way by giving the permission for root, user and others in a numeric form where the numeric value are as follows:

    r(read-only)=>4

    w(write)=>2

    x(executable)=>1

    e.g. **chmod 754 text** will change the ownership of owner to read, write and executable, that of group to read and executable and that of others to read only of the text file.

29. **more**: It is a filter for paging through text one screenful at a time.Use it with any of the commands after the pipe symbol to increase readability.
    1. **e.g. ls -ll |more**
30. **cron** : Daemon to execute scheduled commands. Cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates.

    1 * * * * echo "hi" >/dev/tty1 displays the text "hi" after every 1 minute in tty1

    .————————- minute (0 – 59)

| .————- hour (0 – 23)

| | .———- day of month (1 – 31)

| | | .——- month (1 – 12) OR jan,feb,mar,apr …

| | | | .—- day of week (0 – 7) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat

*** * * * * command to be executed**

31. **fsck:** Used for file system checking. On a non-journaling file system the fsck command can take a very long time to complete. Using it with the option -c displays a progress bar which doesn't increase the speed but lets you know how long you still have to wait for the process to complete.
    1. e.g. **fsck -C**
32. **To find the path of the command**: **which command e.g. which clear**
33. **Setting up alias**: Enables a replacement of a word with another string. It is mainly used for abbreviating a system command, or for adding default arguments to a regularly used command e.g. **alias cls='clear' => For buffer alias of clear**
34. The du (disk usage) command can be used with the option -h to print the space occupied in human readable form. More specifically it can be used with the summation option (-s).
    1. e.g. **du -sh /home** summarizes the total disk usage by the home directory in human readable form.
35. Two or more commands can be combined with the && operator. However the succeeding command is executed if and only if the previous one is true. e.g. **ls && date** lists the contents of the directory first and then gives the system date.
36. Surfing the net in text only mode from the terminal: **elinks [URL]**
    1. e.g: **elinks www.google.com** Note that the elinks package has to be installed in the system.
37. The **ps** command displays a great more deal of information than the **kill** command does.
38. To extract a no. of lines from a file:
    1. **head -n 4 abc.c** is used to extract the first 4 lines of the file abc.c
    2. **tail -n 4 abc.c** is used to extract the last 4 lines of the file abc.c
39. Any changes to a file might cause loss of important data unknowingly. Hence Linux creates a file with the same name followed by ~ (Tilde) sign without the recent changes. This comes in really handy when playing with the configuration files as some sort of a backup is created.
40. A variable can be defined with an '=' operator. Now a long block of text can be assigned to the variable and brought into use repeatedly by just typing the variable name preceded by a $ sign instead of writing the whole chunk of text again and again.
    1. e.g **ldir=/home/my/Desktop/abc**
    2. **cp abcd $ldir** copies the file abcd to /home/my/Desktop/abc.
41. To find all the files in your home directory modified or created today: e.g. **find ~ -type f -mtime 0**

Linux Commands

Directory commands

Every command is an executable file

Basic Commands

pwd – print working diectory
mkdir: create directory
cd: change the current directory
rmdir: remove directories

File Commands

ls: list directory contents
rm: Remove files
cp: copy files
mv: move/ rename files
cat: read one or more files and print them to standard output (short file)
cmp: compare file byte by byte
wc: print number of new lines, words and bytes in files
du: estimate disk usage of each file and recursively for direcctories
find: search files in directory hirearchy
grep: print lines matching a pattern:

Network Commands

wget : a non-interactive network down loader

Even if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. The server will instruct to continue to download from where it it left off.

$ wget url-for-file

ping : send ICMP ECHO_REQUEST to network hosts, you will get back ICMP packet if the host responds. This command is useful when you are in a doubt whether your computer is connected or not.

$ ping IP or host name

hostname : show or set the system's host name

dnsdomainname : show the system's DNS domain name

netstat : displays the status of ports ie. which ports are open, closed, waiting for connections. It displays the contents of **/proc/net** file.

ifconfig : configure a network interface, or to display their current configuration. It is also useful to get

the information about IP address, Subnet Mask,set remote IP address , Netmask etc.

ifup : bring a network interface up

ifdown : take a network interface down

Archive Commands

You want to install a package from its source code. You find out that the source code of the package is archived in a file **xxx.tar**. In this situation, the command-line utility **'tar'** proves to be a vital resource for you**.** The 'tar' is probably the most popular Linux backup utility. If the **'tar'** file is compressed with the compression utility like '**bzip'** or '**gzip'**, the resulting file is the famous '**tarballs'** which is a common method to deliver software installation archives.

tar : an archiving program designed to store and extract files from an archive known as a **tarfile**.

Options :

-c : create a new archive

-r : append files to the end of an archive

-t : list the contents of an archive

-u : only append files that are newer than copy in archive

-x : extract files from an archive

-C : change to directory Dir

-j : filter archive through bzip2, use to decompress .bz2 files.

-v : verbosely list files processed

-f : use archive file

-z : filter the archive through gzip

Examples: tar -xvf test.tar (extract foo.tar to the current location)

tar -xvzf test.tar.gz (extract gzipped test.tar.gz)

tar -cvf test.tar foo/ (compress the contents of foo folder to foo.tar)

Help Commands

There are manual pages for almost all the commands of Linux. You can access the manual pages using man command. The man command offers documentation of the command. If you type:

$ man ls

You will be seeing the manual page of ls with its **name, synopsis, description, author, copyright** etc. Remember, there is a manual page for the man command itself.

If you desire to have a brief reference of the command, use **-help** option with the command. **$ ls -help**

You can even use **info** command to have a quick overview of the command. **$ info ls**

Remember, that memorizing all the commands in Linux along with all its options is a very difficult job. So memorize the command and options which has frequent usage and leave the rest to the **HELP** commands.

Package Management Utilities

On RED HAT, SUSE, and many similar Linux distributions, the RPM Package Manager (RPM) format is used. Ubuntu and Debian, however, uses the Debian Package (DEB) format. Therefore, I have categorized it into two, one for RPM and the other for Debian.

For RPM format, the **rpm** and **yum** is preferred.

rpm options rpm-package-name (use **man rpm** for further more information) The **-q** option tells you if a package is already installed, and the **-qa** option displays a list of all installed packages.

-qa : List all installed RPM applications -qf : Lists applications that own filename -qR : Lists applications on which this application depends -qi : Displays all application information -qd : Lists only documentation files in the application -qc : Lists only configuration files in the application

If you add **p** qualifier to the above options, gives information about specific package. For e.g. -qpl : Lists files in the RPM package

Yum (Yellowdog Update modifier) -**yum** is an automatic updater and package installer/remover for rpm systems. It automatically computes dependencies and figures out what things should occur to install packages. You need to install **yum** in your Linux system.

yum command package-name

e.g. $yum install package-name

Its configuration file is **/etc/yum**.

For **Debian packages**, Advanced Package Tool (**APT)** and Debian Package Tool (**dpkg)** is preferred.

apt command package-name use **apt-get install package-name** to install a package.

Similarly, if you want to upgrade a package use **apt-get upgrade package-name.** With no package specified, apt-get with the **upgrade** command will upgrade your entire system for **FTP** site, or **CD.** You can find configuration files in **/etc/apt.** There are **sources.list, apt.conf** files to look for.

dpkg(**Debian package tool)** is another method to install a binary file with the format **.deb**. To install , type **$dpkg -i xxxx.deb**

To remove, **$dpkg -r xxxx.deb**.

Process Commands

In order to execute a command in the background, place an ampersand(**&**) on the command line at the end of the command. A **user job number**(placed in brackets) and a **system process number** are displayed. A system process number is the number by which the system identifies the job whereas a

user job number is the number by which the user identifies the job.

$ sudo cp -rf * ~/ss &

[1] 9144

$

- **jobs :** lists the jobs being run at the background

$jobs

[1]- Running sudo cp -rf * ~/ss &

[2]+ Running sudo cp -rf * ~/yy &

- The '**+**' sign indicates the job currently being processed , '**-**' sign indicates the upcoming jobs to be executed. The '**% '** used with the job number refrences a job. e.g. Used in **fg.**
- **fg :** a process running in the background will be processed in the foreground

$ **fg % 2**

cat *.cpp > mytext

$

- **kill :** cancels a job running in the background, it takes argument either the user job number or the system process number.

$jobs

[1] + Running cp *.c > mytext

[2] - Running cp *.dat >>mytext

$kill %2

- **bg:** places a suspended job in the background

$ cat *.cpp > mytext

^Z

$bg

(Ctrl + Z will suspend the process running at the moment)

- **ps :** reports a snapshot of the current processes
- **top :** displays Linux tasks
- **at :** executes commands at a specified time.

$ at 8:00

at > echo "HI" > /dev/tty1

(Press 'ctrl + d' to return to the command line. This will display the message in tty1 at 8′o clock.)

- To view the schedule : $ atq
- To cancel a job : $atrm 5 [job ID]
- **crontab** :crontab is a file which contains the schedule of entries to run at specified times.

- **shutdown** : bring the system down
    - -r Requests that the system be rebooted after it has been brought down.
    - -c Cancels a running shutdown.

whoami : displays the login name of the current effective user.

- **logname :** print user´s login name
- **quota :** display disk usage and limits, e.g **$ quota -v**
- **su :** switch to super user or change user ID
- **which :** returns the pathnames of the files which would be executed in the current environment.

Type **$which ls,** you will get **/bin/ls.**

What is `vi`?

The default editor that comes with the UNIX operating system is called `vi` (**vi**sual editor). [Alternate editors for UNIX environments include `pico` and `emacs`, a product of GNU.]

The UNIX `vi` editor is a full screen editor and has two modes of operation:

1. *Command mode* commands which cause action to be taken on the file, and
2. *Insert mode* in which entered text is inserted into the file.

In the command mode, every character typed is a command that does something to the text file being edited; a character typed in the command mode may even cause the `vi` editor to enter the insert mode. In the insert mode, every character typed is added to the text in the file; pressing the `<Esc>` (*Escape*) key turns off the Insert mode.

While there are a number of `vi` commands, just a handful of these is usually sufficient for beginning `vi` users. To assist such users, this Web page contains a sampling of basic `vi` commands. The most basic and useful commands are marked with an asterisk (`*` or star) in the tables below. With practice, these commands should become automatic.

NOTE: Both UNIX and `vi` are **case-sensitive**. Be sure not to use a capital letter in place of a lowercase letter; the results will not be what you expect.

To Get Into and Out Of `vi`

To Start `vi`

To use `vi` on a file, type in `vi filename`. If the file named `filename` exists, then the first page (or screen) of the file will be displayed; if the file does not exist, then an empty file and screen are created into which you may enter text.

*	**vi filename**	*edit* `filename` *starting at line 1*
	vi -r filename	*recover* `filename` *that was being edited when system crashed*

To Exit `vi`

Usually the new or modified file is saved when you leave `vi`. However, it is also possible to quit `vi` without saving the file.

Note: The cursor moves to bottom of screen whenever a colon (`:`) is typed. This type of command is completed by hitting the `<Return>` (or `<Enter>`) key.

*	**:x<Return>**	*quit* `vi`*, writing out modified file to file named in original invocation*
	:wq<Return>	*quit* `vi`*, writing out modified file to file named in original invocation*
	:q<Return>	*quit (or exit)* `vi`
*	**:q!<Return>**	*quit* `vi` *even though latest changes have not been saved for this* `vi` *call*

Moving the Cursor

Unlike many of the PC and MacIntosh editors, **the mouse does not move the cursor** within the `vi` editor screen (or window). You must use the the key commands listed below. On some UNIX platforms, the arrow keys may be used as well; however, since `vi` was designed with the Qwerty keyboard (containing no arrow keys) in mind, the arrow keys sometimes produce strange effects in `vi` and should be avoided.

If you go back and forth between a PC environment and a UNIX environment, you may find that this dissimilarity in methods for cursor movement is the most frustrating difference between the two.

In the table below, the symbol `^` before a letter means that the `<Ctrl>` key should be held down while the letter key is pressed.

*	**j** ***or*** **<Return>** **[*or* down-arrow]**	*move cursor down one line*
*	**k [*or* up-arrow]**	*move cursor up one line*
*	**h** ***or*** **<Backspace>** **[*or* left-arrow]**	*move cursor left one character*
*	**l** ***or*** **<Space>** **[*or* right-arrow]**	*move cursor right one character*
*	**0 (zero)**	*move cursor to start of current line (the one with the cursor)*
*	**$**	*move cursor to end of current line*
	w	*move cursor to beginning of next word*

	b	*move cursor back to beginning of preceding word*
	:0<Return> *or* 1G	*move cursor to first line in file*
	:n<Return> *or* nG	*move cursor to line n*
	:$<Return> *or* G	*move cursor to last line in file*

Screen Manipulation

The following commands allow the `vi` editor screen (or window) to move up or down several lines and to be refreshed.

	^f	*move forward one screen*
	^b	*move backward one screen*
	^d	*move down (forward) one half screen*
	^u	*move up (back) one half screen*
	^l	*redraws the screen*
	^r	*redraws the screen, removing deleted lines*

Adding, Changing, and Deleting Text

Unlike PC editors, you cannot replace or delete text by highlighting it with the mouse. Instead use the commands in the following tables.

Perhaps the most important command is the one that allows you to back up and *undo* your last action. Unfortunately, this command acts like a toggle, undoing and redoing your most recent action. You cannot go back more than one step.

*	**u**	*UNDO WHATEVER YOU JUST DID; a simple toggle*

The main purpose of an editor is to create, add, or modify text for a file.

Inserting or Adding Text

The following commands allow you to insert and add text. Each of these commands puts the `vi` editor into insert mode; thus, the `<Esc>` key must be pressed to terminate the entry of text and to put the `vi` editor back into command mode.

*	**i**	*insert text before cursor, until <Esc> hit*
	I	*insert text at beginning of current line, until <Esc> hit*
*	**a**	*append text after cursor, until <Esc> hit*

	A	*append text to end of current line, until* **<Esc>** *hit*
*	**o**	*open and put text in a new line below current line, until* **<Esc>** *hit*
*	**O**	*open and put text in a new line above current line, until* **<Esc>** *hit*

Changing Text

The following commands allow you to modify text.

*	**r**	*replace single character under cursor (no* **<Esc>** *needed)*
	R	*replace characters, starting with current cursor position, until* **<Esc>** *hit*
	cw	*change the current word with new text,* *starting with the character under cursor, until* **<Esc>** *hit*
	cNw	*change* **N** *words beginning with character under cursor, until* **<Esc>** *hit;* *e.g.,* **c5w** *changes 5 words*
	C	*change (replace) the characters in the current line, until* **<Esc>** *hit*
	cc	*change (replace) the entire current line, stopping when* **<Esc>** *is hit*
	Ncc *or* **cNc**	*change (replace) the next N lines, starting with the current line,* *stopping when* **<Esc>** *is hit*

Deleting Text

The following commands allow you to delete text.

*	**x**	*delete single character under cursor*
	Nx	*delete N characters, starting with character under cursor*
	dw	*delete the single word beginning with character under cursor*
	dNw	*delete* **N** *words beginning with character under cursor;* *e.g.,* **d5w** *deletes 5 words*
	D	*delete the remainder of the line, starting with current cursor position*
*	**dd**	*delete entire current line*
	Ndd *or* **dNd**	*delete* **N** *lines, beginning with the current line;* *e.g.,* **5dd** *deletes 5 lines*

Cutting and Pasting Text

The following commands allow you to copy and paste text.

	yy	*copy (yank, cut) the current line into the buffer*
	Nyy *or* **yNy**	*copy (yank, cut) the next N lines, including the current line, into the buffer*

	p	*put (paste) the line(s) in the buffer into the text after the current line*

Other Commands

Searching Text

A common occurrence in text editing is to replace one word or phase by another. To locate instances of particular sets of characters (or strings), use the following commands.

	/string	*search forward for occurrence of* `string` *in text*
	?string	*search backward for occurrence of* `string` *in text*
	n	*move to next occurrence of search string*
	N	*move to next occurrence of search string in opposite direction*

Determining Line Numbers

Being able to determine the line number of the current line or the total number of lines in the file being edited is sometimes useful.

	:.=	*returns line number of current line at bottom of screen*
	:=	*returns the total number of lines at bottom of screen*
	^g	*provides the current line number, along with the total number of lines, in the file at the bottom of the screen*

Saving and Reading Files

These commands permit you to input and output files other than the named file with which you are currently working.

	:r filename<Return>	*read file named* `filename` *and insert after current line (the line with cursor)*
	:w<Return>	*write current contents to file named in original* `vi` *call*
	:w newfile<Return>	*write current contents to a new file named* `newfile`
	:12,35w smallfile<Return>	*write the contents of the lines numbered 12 through 35 to a new file named* `smallfile`
	:w! prevfile<Return>	*write current contents over a pre-existing file named* `prevfile`