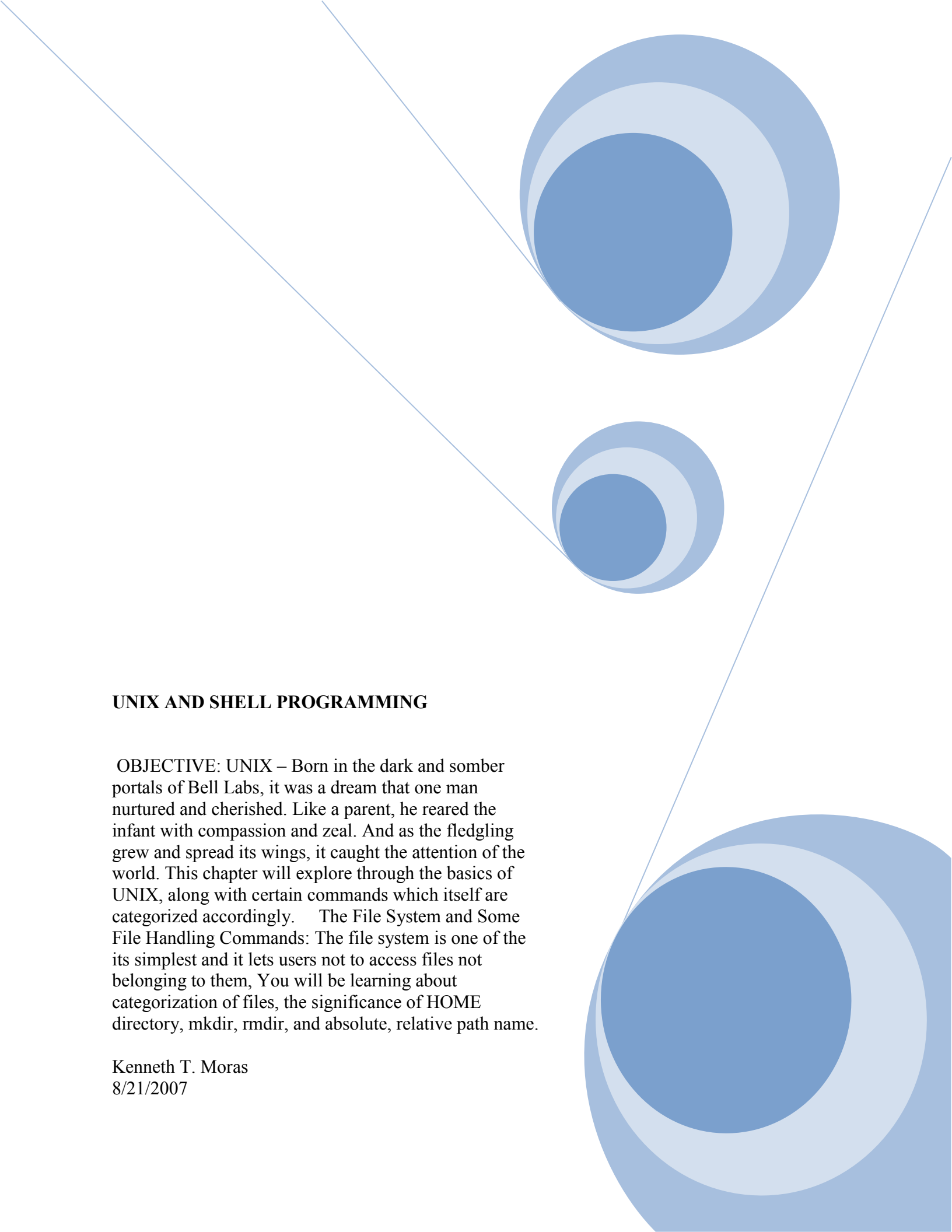


UNIX AND SHELL PROGRAMMING

OBJECTIVE: UNIX – Born in the dark and somber portals of Bell Labs, it was a dream that one man nurtured and cherished. Like a parent, he reared the infant with compassion and zeal. And as the fledgling grew and spread its wings, it caught the attention of the world. This chapter will explore through the basics of UNIX, along with certain commands which itself are categorized accordingly. The File System and Some File Handling Commands: The file system is one of the its simplest and it lets users not to access files not belonging to them, You will be learning about categorization of files, the significance of HOME directory, mkdir, rmdir, and absolute, relative path name.

Kenneth T. Moras
8/21/2007

BRIEF HISTORY

1966

The Multiplexed Time Sharing and Computing System or MULTICS project was a joint attempt by General Electric (GE), AT&T Bell Labs and the Massachusetts Institute of Technology (MIT) at developing a stable multiuser operating system

The aim is to create an operating system that could support a lot of simultaneous users (thousands!).

Multics stands for Multiplexed Information and Computer service.

Left Ken Thompson, Right Dennis Ritchie

The people involved in the project at this time are Ken Thompson, Dennis Ritchie, Joseph Ossanna, Stuart Feldman, Doug McIlroy and Bob Morris.

Although a very simple version of MULTICS could now run on a GE645 computer, it could only support 3 people and therefore the original goals of this operating system had not been met, the research and development is just so expensive and Bell Labs withdraws their sponsorship. This meant that the other interested parties could not afford to carry the project on their own and so they also withdrew.

Dennis Ritchie and Ken Thompson now decide to continue this project on their own.

1969 to 1970

Ken Thompson Dennis Ritchie wrote a Space Travel Game that was actually a serious scientific astronomical simulation program. However the game was a disaster as the spaceship was hard to maneuver and used a lot of resources to run.

After developing the Space Travel Program they had learnt a lot more. With Canaday involved as well they were able to create the design for a new file system, which they built on PDP-7, UNICS (Uniplexed Information and Computing Service), called and this later became UNIX.

Thomson and Ritchie then designed and built a small system having an elegant file system, a command interpreter (SHELL) and a set of UTILITIES. They wanted a general OS running on more than one type of hardware. For this purpose they had to rewrite the entire Unix system in a new efficient programming language, thus C was developed by Dennis Ritchie.

In **1973** they rewrote the entire system in C as a result they succeeded in making UNIX portable.

Berkeley: The second school

In 1974 the source code was made available to certain universities like University of California, Berkeley (UCB). They modified it and made an improved version of Unix of their own. They called it BSD UNIX.

They were the one who created the vi editor and popular C shell. Later they included a networking protocol (TCP/IP) that made Internet possible

OTHER COMPANIES

After AT&T, & Berkeley products, UNIX had turned commercial.

As each vendor modified and enhanced UNIX to create its own version, the original UNIX lost its identity as a separate product.

In 1979 various commercial vendors began to adopt UNIX under license from AT &T. the number of flavors increased (system v, BSD, HP-UX, SOLARIS, IRIX, etc.).

AT&T took it upon themselves to rework mainly the BSD product , and ultimately unify their own System V 3.2 ,BSD,SUN OS & XENIX flavours into its last release . System v release 4 (SVR4)

Shortly thereafter AT & T sold it unix buisness to Novell, who later turned the unix trademark to standards body called X/OPEN , now merged with the open group.

In 1984 Richard Stallman drove the beginnings of the OPEN SOURCE movement with the foundation of GNU. Later this became the free software foundation. They began to introduce open source products to work under UNIX.

In the beginning Unix was purely command based later due to stiff competition given my GUI based OS, Unix also implemented GUI which was first introduced by Massachusetts institute of Tech. known as x window system

FEATURES OF UNIX

1. Multiuser system: So what does it mean that UNIX is a multiuser system???

a) As the name suggests multiuser means many users connected to the single system can simultaneously share the resources of the system.
b) Windows is essentially a single user system where the CPU, memory & hard disk are all dedicated to a single user, which is not, is the case of UNIX.
c) For creating such an illusionary effect the computer breaks up a unit of time into several segments, and each user is allotted a segment. So at any point of time, the machine will be doing the job of a single user. The moment the allocated time expires, the previous job is kept in abeyance and the next users job is taken up. This process goes on until the clock has turned full circle and the first user's job is taken up once again. Thus the kernel does several time in one second and keeps all ignorant and happy

2. Multitasking system:

Multitasking means a single user can run multiple tasks concurrently.

E.g.: user can edit a file, print another file on the printer, send email to a friend & browse the www --- all without leaving any of the application.

In multitasking environment, a user sees one job running in the foreground; the rest run in the background. User can switch between foreground & background, suspend or even terminate them.

4. Pattern matching:

UNIX supports pattern matching by using set of Meta characters.

E.g. * is a Meta character used by the system to indicate that it can match a number of filenames

$ls chap*

Chap1

Chap2

Chap3

Chap4

5. Programming facility:

UNIX shell is also a programming language .It has all the necessary ingredients like control structures, loops and variables.

6. Hierarchical directory structure to support the organization and maintenance of files.

7. Portability

8. Wide range of support tools of like debuggers, compilers.

9. Security is very high in UNIX system.

10. Files and Process:

Files have places and processes have life. A file is considered as being situated in space so that it can be located. Process is the name given to a file when it is executed as a program.

Flexibility in command usage:

1. ***Combining commands***

UNIX allows to specify more than one command in the same line command line. Each command has to be separated by a semicolon (;)

Syntax

$ command1; command 2

eg :

$ (wc note; ls –l note) > filename

2. ***Splitting command line into multiple lines***
   If a command has lengthy syntax, the command line can be split into multiple lines.

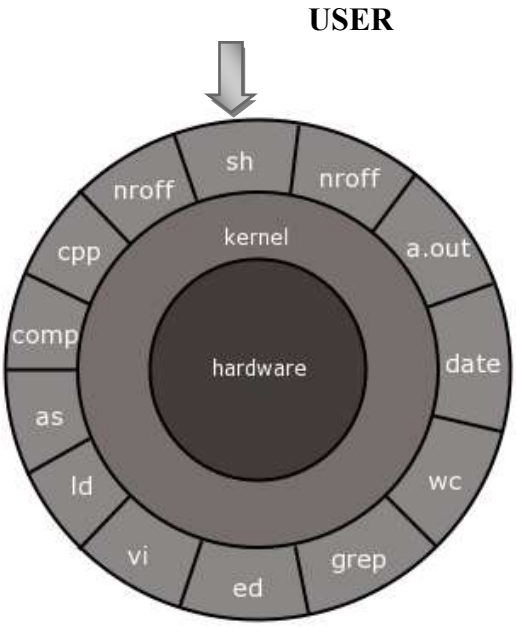
In such case the shell issues a secondary prompt, usually > to indicate that the command line is not complete

E.g.: $ echo “peoples
>education
>institute of
>technology”

Whenever “> “appears after pressing enter key, it is generally due to the absence of matching quote of parenthesis.

3. Entering a command before the previous command has finished:
   Whenever user runs a long program, the prompt will not appear till the program execution has completed.
   Subsequent command can be entered at the keyboard without waiting for the prompt, & they may not be displayed on the screen.
   This is possible because the input remains stored in a buffer (a temporary memory) that is maintained by the kernel for all keyboard input.
   Then the command is passed on to the shell for interpretation after the execution of the previous program is complete.

THE UNIX ARCHITECHTURE



1. **KERNEL**
   a. The kernel is the core of the OS
   b. It is collection of routines written in C, these routines communicate with the hardware directly.
   c. This part of UNIX system gets loaded into the memory when the system is booted.
   d. Application programs that needs to communicate with the hardware use the services of kernel through set of functions called system calls
   e. Some of the tasks performed by kernel are: memory management, schedule processes, decides their priorities etc.

2. SHELL

   a. Computers don't have any inherent capability of translating commands into action. That requires a command interpreter, a job that is handled by the 'OUTER PART' of the OS ----THE SHELL
   b. It is actually the interface between the user and kernel.
   c. Shell provides user with a prompt to interpret commands typed by user.
   d. When the user enters a command through the keyboard, the shell examines the keyboard input for special characters, if it finds any it rebuilds it into simplified command line.

eg.

e. $echo Sun Solaris
   Output would be
f. Sun Solaris ignoring the extra spaces
   In order if user doesn't like such kind of tampering to happen the shell itself provides features to prevent its own interference.
g. Shell is represented by sh (bourne shell), ksh (korn shell), bash (bash shell).

h. To view which shell is being used type $echo $SHELL in command prompt
i. NOTE: UNIX commands are case sensitive

3. UTILITIES

A utility is a standard program, which comes bundled along with Unix OS that provides support for users

common utilities are text editors, browsers, search program & sort programs, interpreters, spread sheets etc...

E.g.

a) Common text editors are vi, emacs , pico etc
b) ls utility(command) displays the files that reside on disk

4. APPLICATION

a. These are the programs that are not the standard part of the UNIX OS.
b. They are written by system admin, professional programmers, or users. They provide extended capability for the system
c. Many of the standard utilities started out as application years ago and have proved to be useful that they are now standard part of OS

INTERNAL AND EXTERNAL COMMANDS

External commands:

The command which have an independent existence in a directory specified by the PATH variable are known as external commands E.g. cat, ls

ls is a command (program) or file having an independent existence in /bin directory

Internal commands:

Commands, which do not have an independent existence in a directory. e.g. echo, vi

It also includes commands, which are normally not executed even if they are in one of the directories specified by the PATH

ex

echo is a shell built in function or command

if a command exists both as an internal & external one, the shell gives priority to its own internal command.

FILES & PROCESSES

a) A file is just an array of bytes and can be virtually contain anything; in general any thing which occupies space is a file.
b) UNIX considers even directories and devices as members of file system, the dominant type is text, and behavior of the system is mainly controlled by text files.

Process is nothing but the name given to a file when it is executed as a program.

a) Process can be considered as the " time image " of a executable file
b) We can treat process as a living organism, which have parents, children and grandchildren, and are born & die.
c) Unix provides tools for process management to move process in foreground and background.

SYSTEM CALLS

a) The UNIX system comprises the kernel, shell & application is written in C. Though there are over a thousand commands in the system, they all have a handful of functions, called system calls
b) One thing in common in all flavors of UNIX is that they use the same system calls
c) LINUX also uses the same system calls hence Linux is UNIX.
d) System calls are built into the kernel, & interaction through them represents an efficient means of communication with the system, which means UNIX can be easily ported to another UNIX machine

Commands in UNIX

echo

$ echo text

This command will print anything written in place of text

Eg $ echo my name is Kenneth T. Moras

Will output

my name is Kenneth T. Moras

printf :An alternative to echo

$ printf “write anything between this double quotes \n”

Here \n is newline character used in the same manner as used in C programming

If the newline character is not used then the prompt will not come in the next line

Another example

Printf “ my current shell is %s \n” $SHELL

This will the current shell under use ,here %s is a string format specifier

bc(basic calculator)

$ bc

12+1

13

[ctrl d] to come out of bc command

script :RECORDING A SESSION

$ script

Script started, the file is typescript

This command is used to record an user's login session in a file, it would be helpful since it stores all keystrokes, error messages as well as output in a file called typescript.

Command to end script :

$ exit

Script done, file is typescript

User can view the contents of the file typescript by typing

$ cat typescript

Next time we use the script command the contents of the file typescript will be overwritten. In order to avoid this we can use the following command

$ script –a appends to existing file typescript

script logfile logs activities to file logfile

passwd : CHANGING YOUR PASSWORD

Use the passwd command to change the password

$ passwd

passwd :changing password for Kenneth

Enter login password *********

New password**********

Reenter new password**********

passwd(SYSTEM) : passwd successfully changed for kenneth

date –This command gives both date & time

$ date

Wed aug 31 16:22:40 IST 2007

By using format specifiers as arguments user can modify the output

$ date +%m will give you the month

08

$ date +%h will give you month name

Aug

OR

These two above commands can be combined as follows

$ date +"%h %m"

Aug 08

cal

This command will print a calendar for a specified month and/or year.

To show this month's calendar, enter:

$ cal

To show a twelve-month calendar for 2008, enter:

$ cal 2008

To show a calendar for just the month of June 1970, enter:

$ cal 6 1970

cd

This command changes your current directory location. By default, your Unix login session begins in your home directory.

To switch to a subdirectory (of the current directory) named myfiles, enter:

cd myfiles

To switch to a directory named /home/dvader/empire_docs, enter:

cd /home/dvader/empire_docs

To move to the parent directory of the current directory, enter:

cd ..

To move to the root directory, enter:

cd /

To return to your home directory, enter:

cd

ls

This command will list the files stored in a directory. To see a brief, multi-column list of the files in the current directory, enter:

ls

To also see "dot" files (configuration files that begin with a period, such as .login), enter:

ls -a

To see the file permissions, owners, and sizes of all files, enter:

ls -la

If the listing is long and scrolls off your screen before you can read it, combine ls with the less utility, for example:

ls -la | less

man

This command displays the manual page for a particular command. If you are unsure how to use a command or want to find out all its options, you might want to try using man to view the manual page.

For example, to learn more about the ls command, enter:

man ls

To learn more about man, enter:

man man

If you are not sure of the exact command name, you can use man with the -k option to help you find the command you need. To see one-line summaries of each reference page that contains the keyword you specify, enter:

man -k keyword

Replace keyword in the above example with the keyword that you want to reference.

To get help on Unix commands, use the man command (from manual) together with the command in question as an argument.

$ man ls
Reformatting page. Please Wait... done

User Commands ls(

NAME
ls - list contents of directory

SYNOPSIS
/usr/bin/ls [-aAbcCdfFgilLmnopqrRstux1] [file ...]

/usr/xpg4/bin/ls [-aAbcCdfFgilLmnopqrRstux1] [file ...]

DESCRIPTION
For each file that is a directory, ls lists the contents of
the directory; for each file that is an ordinary file, ls
.
.
.
.

Understanding the man page

A man page is divided into a number of compulsory and optional sections. Every command doesn't have all the sections, but the first 3(NAME, SYNOPSIS, DESCRIPTION) are generally seen in all man pages.

NAME: Presents one line description of the command

SYNOPSIS: shows the syntax used by the command

DESCRIPTION: Provides detailed description

ps

The ps command displays information about programs (i.e., processes) that are currently running. Entered without arguments, it lists basic information about interactive processes you own. However, it also has many options for determining what processes to display, as well as the amount of information about each. Like lp and lpr, the options available differ between BSD and System V implementations. For example, to view detailed information about all running processes, in a BSD system, you would use ps with the following arguments:

ps

-alxww

To display similar information in System V, use the arguments:

ps -elf

This command displays or changes various settings and options associated with your Unix session.

To see the status of all settings, enter the command without options:

set

If the output scrolls off your screen, combine set with less:

set | less

The syntax used for changing settings is different for the various kinds of Unix shells; see the man entries for set and the references listed at the end of this document for more information.

w and who

The w and who commands are similar programs that list all users logged into the computer. If you use w, you also get a list of what they are doing. If you use who, you also get the IP numbers or computer names of the terminals they are using.

$ who

Root console aug 1 07:51 (:0)

Kenneth pts/10 aug 1 07:56 9 kenneth.moras.com

The first column shows the username of the user currently working on the system

The second column shows the device names of their respective terminal

The third, fourth, & fifth columns show the date and time of logging in.

$ who –Hu

NAME LINE TIME IDLE PID COMMENTS

Root console aug 1 07:51 0:48 11235 (:0)

Kenneth pts/10 aug 1 07 07:56 9 0:33 11142 kenneth.moras.com

$ who am i

Kenneth pts/10 aug 1 07:56 9 kenneth.moras.com

tty: Knowing your terminal

$ tty

/dev/pts/10

UNAME

$ uname

Sun os

This command gives you the version of unix used

$ uname –r

5.8

ie: gives the version of OS used

stty

This command displays and sets various terminal attributes. You can define the key that interrupts a program and marks end of file. Use stty sane to set the terminal to some standard values

df

This command reports file system disk usage (i.e., the amount of space taken up on mounted file systems). For each mounted file system, df reports the file system device, the number of blocks used, the number of blocks available, and the directory where the file system is mounted.

To find out how much disk space is used on each file system, enter the following command:

df

If the df command is not configured to show blocks in kilobytes by default, you can issue the following command:

df -k

du

This command reports disk usage (i.e., the amount of space taken up by a group of files). The du command descends all subdirectories from the directory in which you enter the command, reporting the size of their contents, and finally reporting a total size for all the files it finds.

To find out how much disk space your files take up, switch to your home directory with the cd command, and enter:

du

The numbers reported are the sizes of the files; on different systems, these sizes will be in units of either 512 byte blocks or kilobytes. To learn which is the case, use the man command, described below. On most systems, du -k will give sizes in kilobytes.

find

The find command lists all of the files within a directory and its subdirectories that match a set of conditions. This command is most commonly used to find all of the files that have a certain name.

To find all of the files named myfile.txt in your current directory and all of its subdirectories, enter:

```
find .
```

-name myfile.txt -print

To look in your current directory and its subdirectories for all of the files that end in the extension .txt , enter:

find . -name "*.txt" -print

In these examples, the. (period) represents your current directory. It can be replaced by the full pathname of another directory to search. For instance, to search for files named myfile.txt in the directory /home/user/myusername and its subdirectories, enter:

find /home/user/myusername/ -name myfile.txt -print

On some systems, omitting the final / (slash) after the directory name can cause find to fail to return any results.

As a shortcut for searching in your home directory, enter:

find "$HOME/" -name myfile.txt -print

jobs

This command reports any programs that you suspended and still have running or waiting in the background (if you had pressed Ctrl-z to suspend an editing session, for example). For a list of suspended jobs, enter:

jobs

Each job will be listed with a number; to resume a job, enter % (percent sign) followed by the number of the job. To restart job number two, for example, enter:

%2

This command is only available in the csh, bash, and ksh shells.

The File System

Different types of files

Ordinary file – also known as regular file. It contains only data as a stream of characters

Eg. Text file (often contains printable contents, all c, java, shell & perl scripts are text files),

Binary file (it contains both printable and non printable contents that covers the ASCII

range, most unix commands are binary files and the object code & executables produced by compiling are binary files.

Directory file – its commonly said that a directory contains files and other directories, but strictly speaking it contains their names and a number associated with each name.

A directory file contains an entry for every file and subdirectory that it houses. If you have 20 files in a dir there will be 20 entries in the directory. Each entry has 2 components.

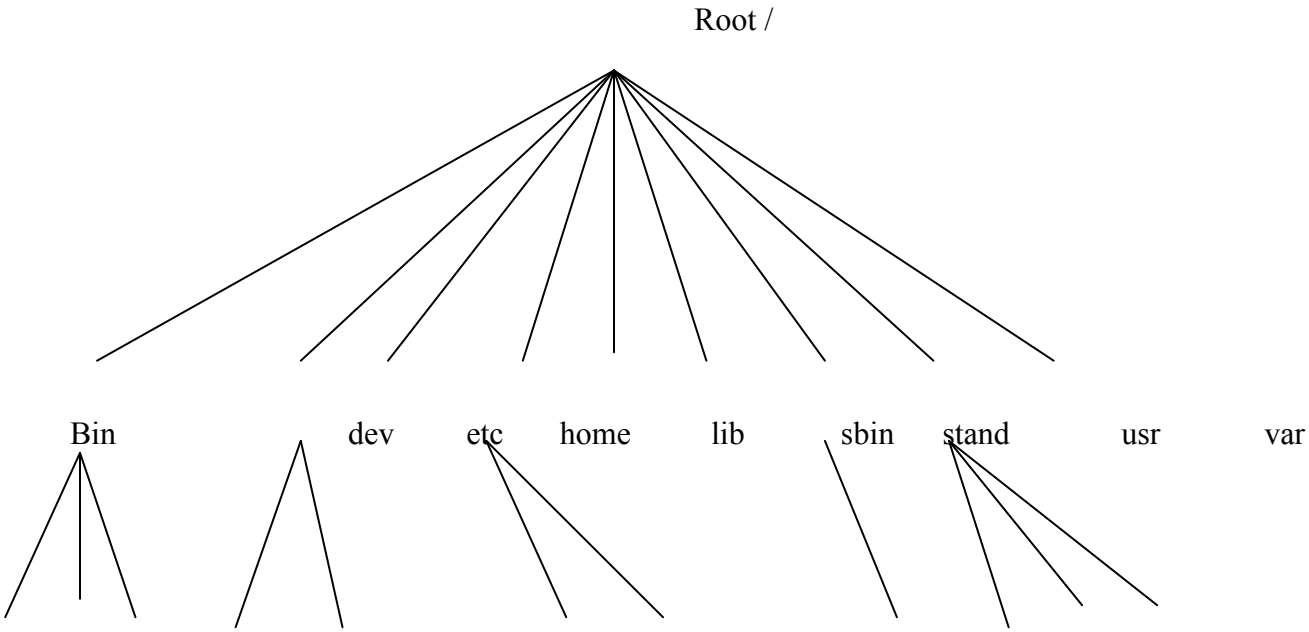
- The filename
- A unique identification number for the file or directory (called the inode number).

Device file – All devices & peripherals are represented by files. To read or write a device, you have to perform these operations on its associated file.

All the device files are stored in the directory /dev.

The kernel identifies a device form its attributes & then uses them to operate the device.

The UNIX file system tree diagram



The root dir has a number of subdirectories under it. These subdir in turn have more subdir & other files under them. Every file apart from the root should have a parent.

The Home Directory

Each unix account is associated with a default home directory. The default home directory for your account will look something like /home/moraskenneth where moraskenneth is replaced with your UW User ID. To print the home directory, type:

$ echo $HOME
/home/moraskenneth

The echo command simply prints information to the screen while $HOME is a special variable recognized by the terminal

pwd(present working directory)

This command reports the current directory path. Enter the command by itself:

$ pwd

/home/moraskenneth

cd

Use cd to change directory. Use pwd to see what directory you are in.

```
  $  cd english
  $  pwd
  $  /u/ma/amritha/english
  $  ls
novel poems
  $ cd novel
  $  pwd
  $ /u/ma/amritha/english/novel
  $ ls
ch1 ch2 ch3 journal scrapbook
  $  cd ..
  $ pwd
  s /u/ma/amritha/english
  $  cd poems
  $ cd
  $ /u/ma/amritha
```

Amritha began in her home directory, and then went to his english subdirectory. He listed this directory using ls , found that it contained two entries, both of which happen to be directories. She cd'd to the diretory novel, and found that she had gotten only as far as chapter 3 in his writing. Then she used cd.. to jump back one level. If had wanted to jump back one level, then go to poems she could have said cd ../poems. Finally she used cd with no argument to jump back to her home directory.

mkdir

This command will create a new subdirectory.

To create a subdirectory named mystuff in the current directory, enter:

mkdir mystuff

To create a subdirectory named morestuff in the existing directory named /tmp, enter:

mkdir /tmp/morestuff

You can create a number of subdirectories using one mkdir command

Note: To make a subdirectory in a particular directory, you must have permission to write to that directory

ABSOLUTE PATHNAMES:

Many UNIX command use file and directory names as arguments, which are presumed to exist in the current directory, for instance the command

cat kenneth

will work only if the file Kenneth exists in your current directory, however if your placed in other directory then you cant access the file using the previous command, in order to display the file you would perhaps have to locate the file in this manner.

Cat /home/Kent/Kenneth

ie the files location should be determined with respect to the root. This is called absolute pathname.

$ /bin/date works perfectly.

NOTE: two files cannot have same absolute pathnames.

RELATIVE PATHNAME:

You do not always have to specify full pathnames to refer to files. As convenient shorthand you can also specify a path to a file relative to your present directory. Such a pathname is called a relative pathname. For ex if you are in your home directory (/home/Kenneth) the two-file name ken. / letters/Ken and proposals/ken as their relative pathnames

HANDLING ORDINARY FILES

cat

This command outputs the contents of a text file. You can use it to read brief files or to concatenate files together.

To append file1 onto the end of file2, enter:

cat file1 >> file2

To view the contents of a file named myfile, enter:

cat myfile

Because cat displays text without pausing, its output may quickly scroll off your screen. Use the less command (described below) or an editor for reading longer text files.

Cat options:

Displaying nonprinting characters (-v) : cat is normally used for displaying text file only. Executables when seen in cat simply displays junk .If you have nonprinting ASCII characters in your input, you can use cat with –v option to display these characters.

Numbering lines: The -n option numbers lines.

Using cat to create files

$ cat > kenneth

Creates a file named Kenneth

Go on typing content on the terminal when finished press [ctrl d]. Now the file is saved.

cp

This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, cp will overwrite and destroy the duplicate. For this reason, it's safest to always add -i after the cp command, to force the system to ask for your approval before it destroys any files. The general syntax for cp is:

cp -i oldfile newfile

To copy a file named meeting1 in the directory /home/dvader/notes to your current directory, enter:

cp -i /home/dvader/notes/meeting1.

The . (period) indicates the current directory as destination, and the -i ensures that if there is another file named meeting1 in the current directory, you will not overwrite it by accident.

To copy a file named oldfile in the current directory to the new name newfile in the mystuff subdirectory of your home directory, enter:

cp -i oldfile ~/mystuff/newfile

The ~ character (tilde) is interpreted as the path of your home directory.

Note: You must have permission to read a file in order to copy it.

lpr and lp

These commands print a file on a printer connected to the computer network. The lpr command is used on BSD systems, and the lp command is used in System V. Both commands may be used on the UITS systems.

To print a file named myfile on a printer named lp1 with lpr, enter:

lpr -Plp1 myfile

To print the same file to the same printer with lp, enter:

lp -dlp1 myfile

Note: Do not print to a printer whose name or location is unfamiliar to you.

rm

This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion. To remove a file named junk, enter:

rm -i junk

Note: Using rm will remove a file permanently, so be sure you really want to delete a file before you use rm.

To remove a non-empty subdirectory, rm accepts the -r option. On most systems this will prompt you to confirm the removal of each file. This behavior can be prevented by adding the -f option. To remove an entire subdirectory named oldstuff and all of its contents, enter:

rm -rf oldstuff

Note: Using this command will cause rm to descend into each subdirectory within the specified subdirectory and remove all files without prompting you. Use this command with caution, as it is very easy to accidently delete important files. As a precaution, use the ls command to list the

files within the subdirectory you wish to remove. To browse through a subdirectory named oldstuff, enter:

ls -R oldstuff | less

$ rm -r*

With the –r or –R option rm performs a tree walk – a through recursive search for all subdirectories and files within these subdirectories .At each stage .it deletes everything it finds .rm wont normally remove directories but when used with this option it will .

wc (word count)

$ wc filename

This command counts the number of lines, words & characters

$ wc –l filename

This command counts only number of lines

$ wc –w filename

This command counts number of words

$ wc –c filename

This command gives number of characters

cmp : comparing two files

You may often need to know whether two files are identical so one of them can be deleted . there are 3 command in UNIX system that can tell you to do that.

$ cmp chap1 chap2

The two files are compared line by line and the location of the first mismatch is echoed to the screen. By default cmp doesn't bother about possible subsequent mismatches but displays a detailed list when used with –l option

If two files are same then cmp displays no message.

$ comm file1 file2

This command is for comparing the files and displaying the lines comman to them.

od: displays data in octal

$ od filename

$ od –b filename

Converts a file into octal representation of that file.

$ od –bc filename

Each line of the output is split into two, one containing the original character followed by the octal representation of that character.

.

mv

This command will move a file. You can use mv not only to change the directory location of a file, but also to rename files. Unlike the cp command, mv will not preserve the original file.

Note: As with the cp command, you should always use -i to make sure you do not overwrite an existing file.

To rename a file named oldname in the current directory to the new name newname, enter:

mv -i oldname newname

[i]To move a file named hw1 from a subdirectory named newhw to another subdirectory named oldhw (both subdirectories of the current directory), enter:

mv -i newhw/hw1 oldhw

If, in this last operation, you also wanted to give the file a new name, such as firsthw, you would enter:

mv -i newhw/hw1 oldhw/firsthw

gzip & gunzip

gzip command is used for compressing the file. It provides the extension .gz to the compressed file. To compress a file named hello, you would enter:

$ gzip hello

gunzip command is used for restoring the original file.

$ gunzip hello.gz

tar

tar command is used for creating a disk archive that contains a group of files or an entire directory structure.

-c option : create an archive

-x option: Extract files from archive

-t option: Display files in archive

To create an archive containing files: file1 and file2, enter following command:

$ tar –cvf archive.tar file1 file2

To extract files from the archive, enter following command:

$ tar –xvf archive.tar

To view contents of an archive, enter following command:

$ tar –tvf archive.tar

less and more

Both less and more display the contents of a file one screen at a time, waiting for you to press the Spacebar between screens. This lets you read text without it scrolling quickly off your screen. The less utility is generally more flexible and powerful than more, but more is available on all Unix systems while less may not be.

To read the contents of a file named textfile in the current directory, enter:

less textfile

The less utility is often used for reading the output of other commands. For example, to read the output of the ls command one screen at a time, enter:

ls -la | less

In both examples, you could substitute more for less with similar results. To exit either less or more, press q.

Note: Do not use less or more with executables (binary files), such as output files produced by compilers. Doing so will display garbage and may lock up your terminal.

kill

Use this command as a last resort to destroy any jobs or programs that you suspended and are unable to restart. Use the jobs command to see a list of suspended jobs. To kill suspended job number three, for example, enter:

kill %3

Now check the job command again. If the job has not been cancelled, harsher measures may be necessary. Enter:

kill -9 %3

NAVIGATION

Irrespective of versions, more uses the spacebar to scroll forward a page at a time. You can also scroll by small and large increments of lines or screens .To move forward one page use

f for spacebar

b to move one page backword

Repeat factor

The repeat factor10 f & 4b are all valid.

BASIC FILE ATTRIBUTES

ls -l (listing file attributes)

$ ls –l this command displays most attributes of a file like its permissions, size and ownership details

$ ls followed by the directory name gives the contents of the directory. To force ls to list to list the attributes of a directory , rather than its contents ,you need to use –d option along with ls

$ ls -ld

- A file can have more than a link ie more than one name without having multiple copies on disk.
- A file has a owner, usually the name of the user (UID) who creates the file .A file is also owned by a group (GID)
  By default the group is to which the user belongs. Both UID & GID are maintained in /etc/passwd. Apart from the superuser, it's only the owner who can change all file attributes.
- The file size displayed by the listing is not the actual space the file occupies on disk , but the number of bytes it contains.
- A file can have read(r), write (w), or execute (x) permission, & there are three sets of such permissions for the user, group, & others.
- A files owner uses chmod to alter file permissions. The permissions can be relative when used with the + or – symbols, or absolute when used with octal numbers. The octal digit 7 includes read (4), write (2) & execute permissions (1) bits.

The vi editor

The vi editor is a screen-based editor used by many UNIX users. The vi editor has powerful features to aid programmers, but many beginning users avoid using vi because the different features overwhelm them.

Starting the vi Editor

The vi editor lets a user create new files or edit existing files. The command to start the vi editor is vi, followed by the filename. For example, to edit a file called temporary, you would type vi temporary and then return. You can start vi without a filename, but when you want to save your work, you will have to tell vi which filename to save it into later.

When you start vi for the first time, you will see a screen filled with tildes (A tilde looks like this: ~) on the left side of the screen. Any blank lines beyond the end of the file are shown this way. At the bottom of your screen, the filename should be shown if you specified an existing file, and the size of the file will be shown as well, like this:

"filename" 21 lines, 385 characters

If the file you specified does not exist, then it will tell you that it is a new file, like this:

"newfile" [New file]

If you started vi without a filename, the bottom line of the screen will just be blank when vi starts.

The vi editor has two modes and in order to get out of vi, you have to be in command mode. Hit the key labeled "Escape" or "Esc" (If your terminal does not have such a key, then try ^[, or control-[.) to get into command mode.

The command to quit out of vi is :q. Once in command mode, type colon, and 'q', followed by return. If your file has been modified in any way, the editor will warn you of this, and not let you quit. To ignore this message, the command to quit out of vi without saving is :q!. This lets you exit vi without saving any of the changes.

If you want to quit saving changes, just type :wq or :x. Both work the same.

If you just want to save, type :w.

3 modes of operation in a vi editor.

Command mode : the default mode of the editor where every key pressed is interpreted as a command to run on text . You will have to be in this mode to copy & delete text.

Input mode : every key pressed after switching to this mode actually shows up as text. This mode is invoked by pressing i .

Ex mode: also known as last line mode, this mode is used to handle files & perform substitutions. Pressing : in the command mode invokes this mode

Cutting and Pasting/Deleting text

"

Specify a buffer to be used any of the commands using buffers. Follow the " with a letter or a number, which corresponds to a buffer.

D

Delete to the end of the line from the current cursor position.

P

Paste the specified buffer before the current cursor position or line. If no buffer is specified (with the " command.) then 'P' uses the general buffer.

X

Delete the character before the cursor.

Y

Yank the current line into the specified buffer. If no buffer is specified, then the general buffer is used.

p

Paste the specified buffer after the current cursor position or line. If no buffer is specified (with the " command.) then 'p' uses the general buffer.

x

Delete character under the cursor. A count tells how many characters to delete. The characters will be deleted after the cursor.

y

Yank until, putting the result into a buffer. "yy" yanks the current line. a count yanks that many lines. The buffer can be specified with the " command. If no buffer is specified, then the general buffer is used.

Inserting New Text

A

Append at the end of the current line.

I

Insert from the beginning of a line.

O

(letter oh) Enter insert mode in a new line above the current cursor position.

a

Enter insert mode, the characters typed in will be inserted after the current cursor position. A count inserts all the text that had been inserted that many times.

i

Enter insert mode, the characters typed in will be inserted before the current cursor position. A count inserts all the text that had been inserted that many times.

o

Enter insert mode in a new line below the current cursor position.

Moving the Cursor within the File

^b

Scroll backwards one page. A count scrolls that many pages.

^d

Scroll forwards half a window. A count scrolls that many lines.

^f

Scroll forwards one page. A count scrolls that many pages.

^u

Scroll backwards half a window. A count scrolls that many lines.

^h

Move the cursor one space to the left. A count moves that many spaces.

^l

Move the cursor one space to the right. A count moves that many spaces.

^k

Move the cursor up one line in the same column. A count moves that many lines up.

^j

Move the cursor down one line in the same column. A count moves that many lines down.

^m

Move to the first character on the next line.

^n

Move the cursor down one line in the same column. A count moves that many lines down.

^p

Move the cursor up one line in the same column. A count moves that many lines up.

$

Move the cursor to the end of the current line. A count moves to the end of the following lines.

%

Move the cursor to the matching parenthesis or brace.

^

Move the cursor to the first non-whitespace character.

(

Move the cursor to the beginning of a sentence.

)

Move the cursor to the beginning of the next sentence.

{

Move the cursor to the preceding paragraph.

}

Move the cursor to the next paragraph.

|

Move the cursor to the column specified by the count.

+

Move the cursor to the first non-whitespace character in the next line.

-

Move the cursor to the first non-whitespace character in the previous line.

_

Move the cursor to the first non-whitespace character in the current line.

0

(Zero) Move the cursor to the first column of the current line.

B

Move the cursor to the first non-whitespace character on the top of the screen.

L

Move the cursor to the first non-whitespace character on the bottom of the screen.

M

Move the cursor to the first non-whitespace character on the middle of the screen.

W

Move forward to the beginning of a word, skipping over punctuation.

b

Move the cursor back one word. If the cursor is in the middle of a word, move the cursor to the first character of that word.

e

Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the last character of that word.

w

Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the first character of the next word.

Moving the Cursor Around the Screen

^e

Scroll forwards one line. A count scrolls that many lines.

^Y

Scroll backwards one line. A count scrolls that many lines.

z

Redraw the screen with the following options. "z<return>" puts the current line on the top of the screen; "z." puts the current line on the center of the screen; and "z-" puts the current line on the bottom of the screen. If you specify a count before the 'z' command, it changes the current line to the line specified. For example, "16z." puts line 16 on the center of the screen.

Replacing Text

C

Change to the end of the line from the current cursor position.

R

Replace text as cursor moves right.

S

Change an entire line.

r

Replace one single character under the cursor by another.

s

Replace one single character under the cursor with any no. of characters.

Search the current line backwards for the character specified after the 'F' command. If found, move the cursor to the position.

Manipulating Character/Line Formatting

~

Switch the case of the character under the cursor.

<

Shift the lines up to where to the left by one shiftwidth. "<<" shifts the current line to the left, and can be specified with a count.

>

Shift the lines up to where to the right by one shiftwidth. ">>" shifts the current line to the right, and can be specified with a count.

J

Join the current line with the next one. A count joins that many lines.

Saving and Quitting

^\

Quit out of "vi" mode and go into "EX" mode. The EX editor is the line editor vi is build upon. The EX command to get back into vi is ":vi".

Q

Quit out of "vi" mode and go into "EX" mode. The ex editor is a line-by-line editor. The EX command to get back into vi is ":vi".

ZZ

Exit the editor, saving if any changes were made.

Miscellany

^G

Show the current filename and the status.

^L

Clear and redraw the screen.

^R

Redraw the screen removing false lines.

^[

Escape key. Cancels partially formed command.

^^

Go back to the last file edited.

!

Execute a shell. If a is specified, the program which is executed using ! uses the specified line(s) as standard input, and will replace those lines with the standard output of the program executed. "!!" executes a program using the current line as input. For example, "!4jsort" will take five lines from the current cursor position and execute sort. After typing the command, there will be a single exclamation point where you can type the command in.

&

Repeat the previous ":s" command.

.

Repeat the last command that modified the file.

:

Begin typing an EX editor command. The command is executed once the user types return. (See section below.)

@

Type the command stored in the specified buffer.

U

Restore the current line to the state it was in before the cursor entered the line.

m

Mark the current position with the character specified after the 'm' command.

u

Undo the last change to the file. Typing 'u' again will re-do the change.

EX Commands

The vi editor is built upon another editor, called EX. The EX editor only edits by line. From the vi editor you use the : command to start entering an EX command. This list given here is not complete, but the commands given are the more commonly used. If more than one line is to be modified by certain commands (such as ":s" and ":w") the range must be specified before the command. For example, to substitute lines 3 through 15, the command is ":3,15s/from/this/g".

:ab string strings

Abbreviation. If a word is typed in vi corresponding to string1, the editor automatically inserts the corresponding words. For example, the abbreviation ":ab usa United States of America" would insert the words, "United States of America" whenever the word "usa" is typed in.

:map keys new_seq

Mapping. This lets you map a key or a sequence of keys to another key or a sequence of keys.

:q

Quit vi. If there have been changes made, the editor will issue a warning message.

:q!

Quit vi without saving changes.

:s/pattern/to_pattern/options

Substitute. This substitutes the specified pattern with the string in the to_pattern. Without options, it only substitutes the first occurence of the pattern. If a 'g' is specified, then all occurences are substituted. For example, the command ":1,$s/Dwayne/Dwight/g" substitutes all occurences of "Dwayne" to "Dwight".

:set [all]

Sets some customizing options to vi and EX. The ":set all" command gives all the possible options. (See the section on customizing vi for some options.)

:una string

Removes the abbreviation previously defined by ":ab".

:unm keys

Removes the mapping defined by ":map".

:vi filename

Starts editing a new file. If changes have not been saved, the editor will give you a warning.

:w

Write out the current file.

:w filename

Write the buffer to the filename specified.

:w >> filename

Append the contents of the buffer to the filename.

:wq

Write the buffer and quit.

[i] NOTES PREPARED BY KENNETH T. MORAS