Comand line interface with DOS

Objectives: At the end of this lesson you shall be able to
- **describe the hierarchical directory system in DOS**
- **use dos commands to create directories and subdirectories**
- **use dos commands to change and list directory**
- **use dos commands to access specific files.**

Hierarchical Directory System: Hierarchy in simple terms, is, organisation or an arrangement of entities. Entities can be anything such as objects, files, people, ideas, or any other thing.

Arrangement refers to, for example, Currency can be arranged by denomination. Pebbles can be arranged by their size .

There are many other ways to organize entities besides hierarchically. But, hierarchical organization is special because by this arrangement you can name each entity by its relationship to other entities.

In DOS, entities are the *Directories* in a directory system. Here, the hierarchy begins with the essential *core* or *root entity*. For instance, in a family tree, we may consider great-great-grand father who was the root cause of our existence as the core entity. In DOS , this core entity is referred to as the the *root directory*.

As in the example considered above, if we consider great-great-grand father as the *root directory*, then, great-grand father, grand father, father are referred as *sub directories*. So the directories under the root directory are called subdirectories in DOS. These subdirectrories can trace their paths back to the root directory.

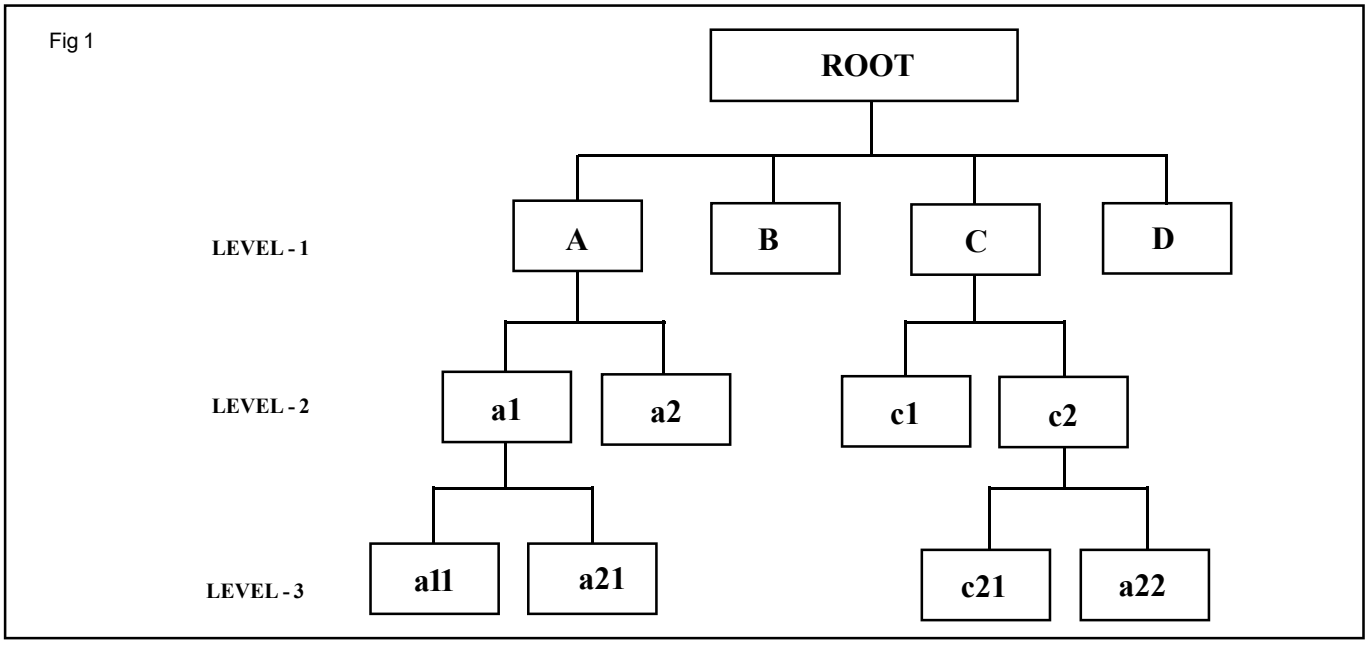
The DOS hierarchical file system is called a *tree-structured file system*. At the base of this tree structure is the root directory.

In a family tree, say, Govinda is the son of Rajappa, who is son of Ramappa who is son of Venkappa. Venkappa is the head or root of the family tree for Govinda.

One can create many directories from the root. The root will then be the parent of each of these directories. You can also create subdirectories that stem from other subdirectories that stem from other subdirectories and so on. These new subdirectories have a subdirectory as their parent directory. How subdirectories are arranged hierarchically from the root is illustrated in Fig 1. The DOS directory system is often called a *tree-structured directory system*.

Three levels of subdirectories are represented in Fig 1. Regardless of the number of levels, the relationship of the subdirectories is important. Each subdirectory, as well as the root directory, can contain user files. Two files can have the same file name and extension as long as the files reside in different directories. This is because, DOS needs to know which of two same-named files your command specifies. For this, DOS needs the name of the directories,starting from the root, that lead to the desired file. This sequence of directory names leading to a file is called a ***path.***

Fig 1



A path is a chain of directory names that tell DOS how to find a file that you want. Each directory is seperated from the other by a **'\'** character. This **'\'** is referred to as the DOS *directory delimiter*. A files full path name including the drive specifier (C: or D: etc.,) is the absolute indicator of where the file is located. Typical path notation are given below;

D:\Animals\Domestic\Pets\Dog.txt

C:\Admin\Accounts\Tours\Bata.txt

Further details of path and directory structure will be discussed at appropriate lessons.

DOS COMMANDS

1 MKDIR Makes or Creates a new Directory.

or

MD

Syntax

MKDIR C:path**dirname**

Or

MD d:path**dirname**

Where,

C: is the disk drive for the sub directory

path\ indicates the path to the directory that will hold the subdirectory being created.

dirname is the name of the subdirectory being creating.

Switch

(None)

Important Notes

– MKDIR **or its short form** MD **makes new subdirectories under the selected root directory.**

– It is possible to create as many subdirectories as you want , but remember: *DOS accepts no more than 63 characters, including backlashes, for the path name*.

– Do not create too many levels of subdirectories and with long names.

– You cannot create a directory name that is identical to a file name in the current directory.

For example, if you have a file named FLIES in the current directory, you cannot create a subdirectory by the name FLIES in this directory. However, if the file FLIES is given an extension FLIES. DOC , then the names will not conflict and you can create a subdirectory by name FLIES.

Examples

To create the subdirectory by name **Drivers** under the **current drive**, the instruction will be,

MKDIR**Drivers**

Or

MD**Drivers**

C:\Devices>MD \Printers

This instruction creates a subdirectory by name **Printers** under the current drive C:. Note that although the command is issued from another subdirectory named devices, the newly created subdirectory **Printers** does not get created under the directory Devices but directly under the root C:. This may be verified by issuing DIR command under C:\ and under C:\Devices.

To create a subdirectory under the directory Devices the instruction will be,

C:\Devices>MD Printers

Discuss the following different varieties of creating directories:

C:\Devices\Printers>MD C:\Devices\Plotters

2 CHIDR or CD

Changes or shows the path of the current directory.

Syntax

CHIDR d: path

Or, using the short form:

CD d : path

D : path are valid disk drive and directory names.

Switch

(None)

You have two methods for maneuvering through the hierarchical directories with CD: (1) starting at the root , or top, directory of the disk and moving down, or (2) starting with the current directory and moving in either direction.

To start at the root directory of a disk, you must begin the path with the path character (\), as in \ or **B:\.** When DOS sees \ as the first character in the path, the system starts with the root directory. Otherwise, DOS starts with the current directory.

Changing Drives: Computer will have built in memory, the hard disk and it will also have provision to store/read data from floppy disk, compact disk etc. Every disk is identified by a name such as C drive, A drive, B drive etc. C drive is represented by C: and A drive is represented by A: and so on. DOS allows to change from current or default drive by typing the letter identification of disk drive desired followed immediately by a colon as shown in the example below:

C\> a:

This command instructs to change control from **C** drive to **D** drive.

If the disk drive is not accessed due to non availability of floppy or any other reason, DOS will display an error message

Not ready error reading drive A

Abort, Retry, Fail ?

It is required to press either A,R or F keys, which are defined below

A Directs DOS to abort the command that was responsible for the error. If this option is selected DOS will terminate the command and redisplay prompt.

R Directs DOS to retry the command that caused the error. In most cases this option is selected to correct the the problem that was causing the error. (Floppy disk might not be inserted).

F Directs DOS to ignore the error and attempt to continue processing. In some cases DOS will have an error when it reads a portion of disk.

DOS COMMAND

DIR Displays a list of files and subdirectories in a directory.

Syntax

DIR C:path/filename [/P] [/W] [/A[[:]attribs]] [/O[[:]sortord]]

[/S] [/B] [/L] [/C[H]]

Where,

- **C:** is the disk drive holding the directory for displaying list of files and subdirectories
- **path/** specifies directory and/or files to list.
- **filename** specifies file or list of files to display, if file name is not specified all the files in the directory will be listed.
- **[/P] [/W] .....** specifies the switches for formatting the output.

Switch

/ P Pauses after each screenful of information and waits to press any key. On pressing any key another screenful or remaining information will be displayed. Command is DIR/P

/ W Uses wide format of 80-column to display file names only and information about file size, date, and time is not displayed. Command is DIR/W

/ A Displays files with specified attributes.

attribs

D Directories

R Read-only files

H Hidden files

S System files

A Files ready to archive - Prefix meaning “not”

/ O List be files in sorted order.

sorted N By name (alphabetic)

S By size (smallest first)

E By extension (alphabetic)

D By date & time (earliest first)

G Group directories first

- Prefix to reverse order

C By compression ratio (smallest first)

/ S Displays files in specified directory and all subdirectories.

/ B Uses bare format (no heading information or summary).

/ L Uses lowercase.

/ C[H] Displays file compression ratio; /CH uses host allocation unit size.

Important Notes:

– In the directory listing similar files can be listed by using wildcards (* and ?), where (*) star and (?) question mark are called wild characters or wild cards. * can replace remaining charecters and ? can replace any single character.

– When DIR is used without parameters or switches, It displays the disks volume label and serial number; one directory or filename per line, including the file size in bytes, and the date and time the file was modified; and the total number of files listed, their cumulative size and the free space (in bytes) remaining on the disk.

Examples

DIR *.txt

*.txt instruction will list all files having txt extension in the specified directory.

DIR ???T.*

???T instruction will search for files having four characters which ends with T like TEST, REST etc. And * instructs that these files may have any extension like .txt, .dat etc.

Methods to display the contents of a text file

Objectives: At the end of this lesson you shall be able to
- **use DOS commands to display the contents of a text file**
- **use DOS commands to copy,rename,delete and undelete files.**

DOS Commands

TYPE Displays the contents of a text file.

Syntax

TYPE C:path/filename

Where,

– **C:** is the disk drive holding the file for displaying.

– **path/** Specifies the location of file for displaying.

– **filename** specifies file to display.

Switch

(none)

Important notes:

– **TYPE** command provides a quick way to display contents of an ASCII file with out having to use another program. The file is stored on the disk as ASCII (American Standard code for Information Interchange) text. which is standard way the computer translates binary (ones and zeros) into letters, numbers & symbols. If the information is not stored in the ASCII format, on using TYPE command the information will look like gibberish.

– On issuing command DOS will look in drive specified, moves into the path to reach the filename specified. Then it simply translates ASCII format into the characters, numbers and symbols and displays on the monitor. The video monitor can show 24 lines of information only. if the file contains more than 24 lines starting lines can not be seen since the type command simply scrolls all information on to the screen. Scrolling can be controlled by pressing Control + S keys together (on holding control key press S key and release both the keys is called as Control + S) scrolling of information will stop on the monitor. After viewing the contents on the screen any key can be pressed to scroll through the remaining contents. To view the contents of the file screen page by screen page, MORE command can also be used. which will stop the scrolling of information on the screen exactly after a screen page and in the screen page at 24 line a prompt message — More— is displayed. After pressing any key another screen page will be displayed. MORE is a filter e.g. itis a program that manipulates the stream of standard characters to the file to the standard output (monitor) screen page by screen page.

Examples

1 C:\COPA\DOS\PRACT_3>**TYPE TEST1.txt**

C:\COPA\DOS\PRACT_3 is the path to the file TEST.txt and TYPE is the command to be executed by DOS.

2 C:\>TYPE C:\COPA\DOS\PRACT_3\>**TEST1.txt**

This results in the same output as in example1. While working from C: (C drive) this command can be issued with out changing the directories.

3 C:\COPA\DOS\PRACT_3>**TYPE TEST1.txt | MORE**

This will also result in the same output but displayed screen page by screen page. Screen page can be changed on press of any key. Along with MORE another character is prefixed '|' this called the piping command, Which will route the output of TYPE command to another command MORE and the MORE filter outputs the information.

Renaming of file(s)

RENAME This command allows to change
Or the name of a file.
REN

Syntax

REN C: PATH\filename1.ex1 filename2.ex2

Where,

– **C:** is the disk drive holding the file for displaying.

– **PATH/** Specifies the location of file for displaying.

– filename1.ex1 is the file to be renamed

– filename2.ex2 is the new filename

Important Notes:

– If the drive is not specified current disk drive will be used.

– If the path is not specified current directory will be used

– Exact file name with extension is to be given for the file to be renamed.

– A valid file name with appropriate extension is to be given for new filename.

– Wild characters are permitted in the file names by which required group of files can be renamed.

– Only file names will be changed and contents remain same.

– If attempted to change a file name to a name that already exists in the directory.

  DOS prompts an error message

  Duplicate file name or file not found

– If a invalid file name or the new name is not given, then also DOS prompts an error message

Rules for the file names.

– A File name must have 1 to 8 characters.

– An optional extension of 1 to 3 characters

– A period (.) between the name and extension name, if extension is used

– All letters from A through Z (lower case letters are automatically transferred to uppercase), 0 to 9 numbers and special characters & symbols $ # & @ ! ^ ( ) _ - { } ‘ ~ are permitted in the file name.

– The control characters such as Esc, Del, or space bar cannot be used in the file name.

– The characters + = / [ ] : ; ? * < > : are not permitted.

– Each file name in a directory must be unique.

Examples:

1 C:\COPA\DOS\PRACT_3\>**REN TEST2.txt CHECKED.txt**

  C:\COPA\DOS\PRACT_3\ is the drive and path to the TEST2.txt file

  TEST2.txt is the file name to be renamed

  CHECKED.txt is the new filename

2 C:\COPA\DOS\PRACT_3\>REN *.pic *.jpg the pic extension will be changed to jpg extension files.

Copying files:

COPY Copies one or more files to another location.

Syntax

COPY [/A | /B] source [/A | /B] [+ source [/A | /B] [+ ...]] [destination[/A | /B]] [/V] [/Y | /-Y] source specifies the file or files to be copied. Destination specifies the directory and/or filename for the new file(s).

Switches

/A Indicates an ASCII text file.

/B Indicates a binary file.

/V Verifies that new files are written correctly.

/Y Suppresses prompting to confirm you want to overwrite an existing destination file.

/-Y Causes prompting to confirm you want to overwrite an existing destination file.

Instructor shall discuss the simple switches with at least two examples in each case .

For further details on COPY command switches refer any tutorial or hand book on DOS

Important Notes:

– DOS command COPY can duplicate one or more files. In the same directory with different names or from one directory to other directory either in the same name or in different name.

– If the drive is not specified current disk drive will be used.

– If the path is not specified current directory will be used.

– Exact file name with extension is to be given for the file to be copied

– A valid file name with appropriate extension is to be given for new copied filename

– Wild characters are permitted in the file names by which required group of files can be copied

– On copying, both source and target files will have same contents.

– Copy overwrites the target file with the same name

– Copy will not allow to copy a file to it self that is source and target files should not be same

– If the destination file name is not specified while concatenation the first file name will become the destination name. After the first file name, additional source files must be preceded by a plus (+) sign.

Example

1 C:\COPA\DOS\PRACT_3\>**COPY TEST2.txt TRIAL.txt**

  With the above command C:\COPA\DOS\PRACT_3 directory TEST2.txt file will be copied as TRIAL.txt file in the same directory. On listing the directory both the files will have same details and on viewing the contents of both the file will be same. After copying DOS prompts a message 1 file copied

2 C:\COPA\DOS\PRACT_3\>**COPY *.bmp *.pic**

  With the above command C:\COPA\DOS\PRACT_3 directory all files with bmp extension file will be copied as pic extension files in the same directory. While copying DOS prompts the name of file it has copied and after completion of copying it prompts the number of files copied.

3 C:\COPA\DOS\PRACT_3\>**COPY *.pic C:\COPA\DOS\PRACT_4**

  All files with pic extension in C:\COPA\DOS\PRACT_3 directory will be copied to C:\COPA\DOS\PRACT_4 directory with same name & extension.

  Using *.* after the copy command will copy all files with all extension to the destination.

Copy concatenating: Multiple file can be combined to form a single file by use of **+** between the source files and is called as concatenation

Example 4 C:\COPA\DOS\PRACT_3\ **COPY TEST2.txt + TRIAL.txt CONCAT1.txt**

With the above command TEST2.txt and TRIAL.txt will be combined and CONCAT1.txt file will be created which will have the contents of first two source files.

Deleting file

DEL Deletes the files specified.

or

Erase

Syntax

DEL C:path/filename [/P]

ERASE C:path/filename [/P]

Where,

- **C:** is the disk drive holding the file to be deleted.
- **path/** Specifies the location of file to be deleted.
- filename is the file to be deleted

Switch

/P Prompts for confirmation before deleting the specified file. Using the /P switch

If the /P switch is used, DEL displays the name of a file and prompts with a message in the following format:

filename, Delete (Y/N)?

Press Y to confirm the deletion, N to cancel the deletion and display the next filename (if a group of files are specified), or CRTL+C to stop the DEL command.

Important Notes

- If the drive is not specified current disk drive will be used
- If the path is not specified current directory will be used
- Exact file name with extension is to be given for the file to be deleted
- Wild characters are permitted in the file names by which required group of files can be deleted
- On deleting, files name(s) will be removed from the directory.
- All the files in a directory can be deleted by typing the DEL command followed by [drive:]path. Wildcard also can be used (* and ?) to delete more than one file at a time. However, Wildcards should be used cautiously with the DEL command to avoid deleting files unintentionally.

The following command is given for deleting all the files.

del *.*

DEL displays the following prompt:

All files in directory will be deleted! Are you sure (Y/N)?

Press Y and then ENTER to delete all files in the current directory, or

press N and then ENTER to cancel the deletion.

- Directories can not be removed with DEL command a separate command is available for removing the directory.
- Once the file is deleted it can not be recovered if the memory space is occupied by a new file. If accidentally file (s) are deleted immediately it can be recovered by using utility command.
- The space occupied by the deleted file on the disk or diskette is freed.
- Check for the typographic errors in the file names before the press of enter key to activate delete command

Example

1 C:\COPA\DOS\PRACT_3\>**DEL TEST2.txt**

With the above command TEST2.txt file will be deleted from the C:\COPA\DOS\PRACT_3 directory. On listing the directory TEST2,txt will not be available.

2 C:\COPA\DOS\PRACT_4\>**DEL *.txt**

With the above command in the C:\COPA\DOS\PRACT_4 directory all files with txt extension will be deleted.

3 C:\COPA\DOS\PRACT_3\TEMP \> **DEL *.***

All files with any extension in C:\COPA\DOS\ PRACT_3\TEMP directory will be deleted.

Recovering deleted files:

UNDELETE delete protection facility

Syntax

UNDELETE C:path/filename [/DT | /DS | /DOS]

UNDELETE [/LIST | /ALL | /PURGE[DRIVE] | / STATUS | /LOAD | /UNLOAD

/UNLOAD | /S[DRIVE] | /T[DRIVE]-entrys]]

Where,

- **C:** is the disk drive holding the files to be undeleted.
- **path/** Specifies the location of file to be undeleted.
- filename is the file to be undeleted

Switches

/LIST	Lists the deleted files available to be recovered.
/ALL	Recovers files without prompting for confirmation.
/DOS	Recovers files listed as deleted by MS-DOS.
/DT	Recovers files protected by Delete Tracker.
/DS	Recovers files protected by Delete Sentry.
/LOAD	Loads Undelete into memory for delete protection.
/UNLOAD	Unloads Undelete from memory.
/PURGE[drive]	Purges all files in the Delete Sentry directory.
/STATUS	Display the protection method in effect for each drive.
/S[drive]	Enables Delete Sentry method of protection.
/T[drive][-entrys]	Enables Delete Tracking method of protection.

Important Notes:

Once a file is deleted from disk, it may not be possible to retrieve it. Although the UNDELETE command can retrieve deleted files, it can do so with certainty only if no other files have been created or changed on the disk. If a file is accidentally deleted and it is required to keep, stop what all other activities on the computer and immediately use the UNDELETE command to retrieve the file.

Example

1 C:\COPA\DOS\PRACT_3\>UN**DELETE TEST2.txt**

With the above command TEST2.txt file will be recovered. On listing TEST2.txt file will be available in C:\COPA\DOS\PRACT_3 directory.

2 C:\COPA\DOS\PRACT_4\TEMP\>UN**DELETE**

With the above command multiple files can be recovered. DOS will prompt for confirmation of undeletion of each file and asks to type the first letter of the file. After undeletion and listing of C:\COPA\DOS\PRACT_4 directory, undeleted file names can be seen .

3 C:\COPA\DOS\PRACT_4\TEMP\>UN**DELETE /ALL**

With the above command multiple files can be recovered. DOS will not prompt for confirmation of undeletion of each file. After undeletion and listing of C:\COPA\DOS\PRACT_4 directory, undeleted file names can be seen.

Introduction to Linux operating system

Objectives: At the end of this lesson you shall be able to
- **overview of linux**
- **define futures of linux**
- **explain application area of linux**
- **describe about kernel.**

Overview of Linux

The operating system

Developers need special tools (like the compilers and command lines found in GNU) to write applications that can talk to the kernel. They also need tools and applications to make it easy for outside applications to access the kernel after the application is written and installed.

This collective set of tools, combined with a kernel, is known as the operating system. It is generally the lowest layer of the computer's software that is accessible by the average user. General users get to the operating system when they access the command line.

Linux provides powerful tools with which to write their applications: developer environments, editors, and compilers are designed to take a developer's code and convert it to something that can access the kernel and get tasks done.

Like the kernel, the Linux operating system is also modular. Developers can pick and choose the operating tools to provide users and developers with a new flavor of Linux designed to meet specific tasks.

Introduction to Linux

Linux (pronounced Lih-nucks) is a UNIX-like operating system that runs on many different computers. Although many people might refer to Linux as the operating system and included software, strictly speaking, Linux is the operating system kernel, which comes with a distribution of software.

Linux was first released in 1991 by its author Linus Torvalds at the University of Helsinki. Since then it has grown tremendously in popularity as programmers around the world embraced his project of building a free operating system, adding features, and fixing problems.

Linux is popular with today's generation of computer users for the same reasons early versions of the UNIX operating system enticed fans more than 20 years ago. Linux is portable, which means you'll find versions running on name-brand or clone PCs, Apple Macintoshes, Sun workstations, or Digital Equipment Corporation Alpha-based computers. Linux also comes with source code, so you can change or customize the software to adapt to your needs. Finally, Linux is a great operating system, rich in features adopted from other versions of UNIX.

Where is Linux?

One of the most noted properties of Linux is where it can be used. Windows and OS X are predominantly found on personal computing devices such as desktop and laptop computers. Other operating systems, such as Symbian, are found on small devices such as phones and PDAs, while mainframes and supercomputers found in major academic and corporate labs use specialized operating systems such as AS/400 and the Cray OS.

Linux, which began its existence as a server OS and Has become useful as a desktop OS, can also be used on all of these devices. ,ÄúFrom wristwatches to supercomputers,,Äù is the popular description of Linux' capabilities.

The future of Linux

Linux is already successful on many different kinds of devices, but there are also many technological areas where Linux is moving towards, even as desktop and server development continues to grow faster than any other operating system today.

Linux is being installed on the system BIOS of laptop and notebook computers, which will enable users to turn their devices on in a matter of seconds, bringing up a streamlined Linux environment. This environment will have Internet connectivity tools such as a web browser and an e-mail client, allowing users to work on the Internet without having to boot all the way into their device's primary operating system-even if that operating system is Windows.

At the same time, Linux is showing up on mobile Internet devices (MIDs). This includes embedded devices such as smart phones and PDAs, as well as netbook devices-small laptop-type machines that feature the core functionality of their larger counterparts in a smaller, more energy-efficient package.

The growth of cloud computing is a natural fit for Linux, which already runs many of the Internet's web servers. Linux enables cloud services such as Amazon's A3 to work with superior capability to deliver online applications and information to users.

Related to Linux' growth in cloud computing is the well-known success of Linux on supercomputers, both in the high-performance computing (HPC) and high-availability (HA) areas, where academic research in physics and bioengineering, and firms in the financial and energy

industries need reliable and scalable computing power to accomplish their goals.

Many of the popular Web 2.0 services on the Internet, such as Twitter, Linked In, YouTube, and Google all rely on Linux as their operating system. As new web services arrive in the future, Linux will increasingly be the platform that drives these new technologies.

Current application of Linux operating systems

Today Linux has joined the desktop market. Linux developers concentrated on networking and services in the beginning, and office applications have been the last barrier to be taken down. They don't like to admit that Microsoft is ruling this market, so plenty of alternatives have been started over the last couple of years to make Linux an acceptable choice as a workstation, providing an easy user interface and MS compatible office applications like word processors, spreadsheets, presentations and the like. On the server side, Linux is well-known as a stable and reliable platform, providing database and trading services for companies like Amazon, the well-known online bookshop, US Post Office, the German army and many others. Especially Internet providers and Internet service providers have grown fond of Linux as firewall, proxy- and web server, and you will find a Linux box within reach of every UNIX system administrator who appreciates a comfortable management station. In post offices, they are the nerve centres that route mail and in large search engine, clusters are used to perform internet searches. These are only a few of the thousands of heavy-duty jobs that Linux is performing day-to-day across the world. It is also worth to note that modern Linux not only runs on workstations, mid- and high-end servers, but also on "gadgets" like PDA's, mobiles, a shipload of embedded applications and even on experimental wristwatches. This makes Linux the only operating system in the world covering such a wide range of hardware.

The code

Linux is also unique from other operating systems in that it has no single owner. Torvalds still manages the development of the Linux kernel, but commercial and private developers contribute other software to make the whole Linux operating system.

The kernel

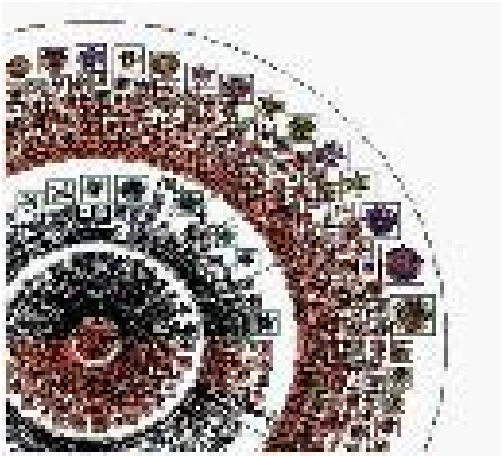
All operating systems have kernels, built around the architectural metaphor that there must be a central set of instructions to direct device hardware, surrounded by various modular layers of functionality. The Linux kernel is unique and flexible because it is also modular in nature.

Modularity is desirable because it allows developers to shed parts of the kernel they don't need to use. Typically a smaller kernel is a faster kernel, because it isn't running processes it does not need.

If a device developer wants a version of Linux to run on a cell phone, she does not need the kernel functionality that deals with disk drives, Ethernet devices, or big monitor screens. She can pull out those pieces (and others), leaving just the optimized kernel to use for the phone.

The kernel of the Window operating system (which few people outside of Microsoft are allowed to look at without paying for the privilege) is a solidly connected piece of code, unable to be easily broken up into pieces. It is difficult (if not impossible) to pare down the Windows kernel to fit on a phone.

Fig 1



This modularity is significant to the success of Linux. The ability to scale down (or up) to meet the needs of a specific platform is a big advantage over other operating systems constrained to just a few possible platforms.

Modularity also effects stability and security as well. If one piece of the kernel code happens to fail, the rest of the kernel will not crash. Similarly, an illicit attack on one part of the kernel (or the rest of the operating system) might hamper that part of the code, but should not compromise the security of the whole device.

The environments

The windows, menus, and dialog boxes most people think of as part of the operating system are actually separate layers, known as the windowing system and the desktop environment.

These layers provide the human-oriented graphical user interface (GUI) that enables users to easily work with applications in the operating system and third-party applications to be installed on the operating system.

In Linux, there a lot of choices for which windowing system and desktop environment can be used, something that Linux allows users to decide. This cannot be done in Windows and it's difficult to do in OS X.

Like the operating system and kernel, there are tools and code libraries available that let application developers to more readily work with these environments (e.g., gtk+ for GNOME, Qt for KDE).

The applications

Operating systems have two kinds of applications: those that are essential components of the operating system itself, and those that users will install later. Closed operating systems, like Windows and OS X, will not let users (or developers) pick and choose the essential component applications they can use. Windows developers must use Microsoft's compiler, windowing system, and so on.

Linux application developers have a larger set of choices to develop their application. This allows more flexibility to build an application, but it does mean a developer will need to decide which Linux components to use.

The distributions

A Linux distribution is a collection of (usually open source) software on top of a Linux kernel. A distribution (or short, distro) can bundle server software, system management tools, documentation and many desktop applications in a central secure software repository. A distro aims to provide a common look and feel, secure and easy software management and often a specific operational purpose.

Let's take a look at some popular distributions.

Red hat

Red Hat is a billion dollar commercial Linux company that puts a lot of effort in developing Linux. They have hundreds of Linux specialists and are known for their excellent support. They give their products (Red Hat Enterprise Linux and Fedora) away for free. While Red Hat Enterprise Linux (RHEL) is well tested before release and supported for up to seven years after release, Fedora is a distro with faster updates but without support.

Ubuntu

Canonical started sending out free compact discs with Ubuntu Linux in 2004 and quickly became popular for home users (many switching from Microsoft Windows). Canonical wants Ubuntu to be an easy to use graphical Linux desktop without need to ever see a command line. Of course they also want to make a profit by selling support for Ubuntu.

Debian

There is no company behind Debian. Instead there are thousands of well organised developers that elect a Debian Project Leader every two years. Debian is seen as one of the most stable Linux distributions. It is also the basis of every release of Ubuntu. Debian comes in three versions: stable, testing and unstable. Every Debian release is named after a character in the movie Toy Story.

Other

Distributions like Cent OS, Oracle Enterprise Linux and Scientific Linux are based on Red Hat Enterprise Linux and share many of the same principles, directories and system administration techniques. Linux Mint, Edubuntu and many other ubuntu named distributions are based on Ubuntu and thus share a lot with Debian. There are hundreds of other Linux distributions.

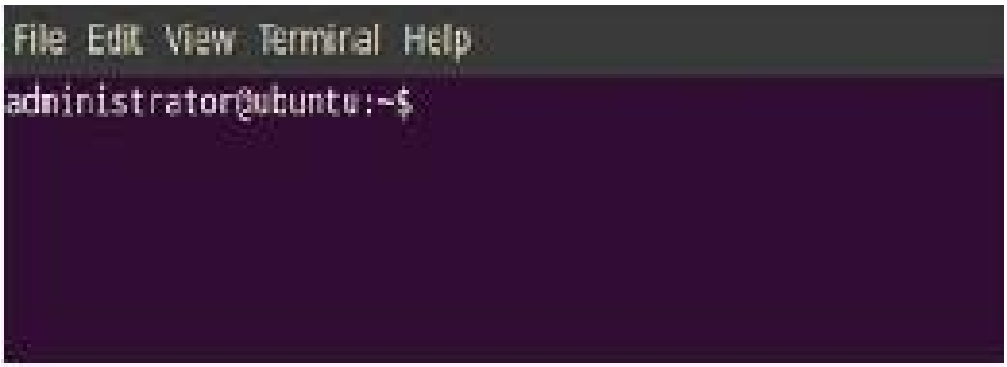
Handling commands and various editors

Objectives: At the end of this lesson you shall be able to
- **know about terminal**
- **explain the command shell**
- **list out the directory layout of linux**
- **define the linux commands**
- **list out the special characters of linux OS**
- **explain various editors in linux OS.**

Starting up a terminal

To access the shell we will use a shell-like application, also called a terminal emulator. There is a huge number of good terminal applications out there, including the default ones in GNOME or KDE, or Yakuake, Guake, rxvt and so on. For now let's just stick with the default that comes with your system. If you're using GNOME you can access the terminal by going to **Applications -> Accessories -> Terminal** or pressing Alt+F2 and typing gnome-terminal in the run box that appears, followed by Enter. If you're running KDE you can type instead **console** after pressing Alt+F2.

Fig 1



Depending on the distribution, the prompt may look something like **user@host$**. The first part before the ampersand is the login username, and the other one is the hostname of the computer.

Command shell

A shell is a **command interpreter** which allows you to interact with the computer. The way things work is pretty simple: you type in commands, the shell interprets them, performs the tasks it was asked to do, and finally it sends the results to the standard output, which is usually the screen.

This is a list of files inside the root directory. The root directory is the first location in the file system tree hierarchy, and it is represented by the **slash** character : **/.**

Some of the most popular shells are:

- **bash -** the Bourne-Again Shell, the default shell on most Linux systems.
- **sh -** the Bourne Shell, an older shell which is not so widely used anymore.
- **csh -** the 'C' Shell, which accepts a syntax which resembles the 'C' programming language.
- **tcsh -** an improved version of the 'C' Shell.
- **ksh -** the Korn Shell, initially developed in the early 1980's.
- **dash -** Debian Almquist Shell, a shell created by the Debian distribution.

Listing of shells available in the system

$ cat /etc/shells/

The above command will display the following output as on Fig 2.

In this tutorial we will focus on **Bash**, since it is the most widely used and also one of the most powerful shells out there. Bash is a modern implementation of the older Bourne Shell (**sh**), developed by the GNU project, which provides a huge amount of tools and which, together with the Linux kernel, desktop environments like GNOME or KDE and applications which run on top of them, comprise the whole Linux platform. On a Debian or Ubuntu distribution, the default shell used by the system is specified in the file **/etc/passwd** (default being Bash).

Fig 2

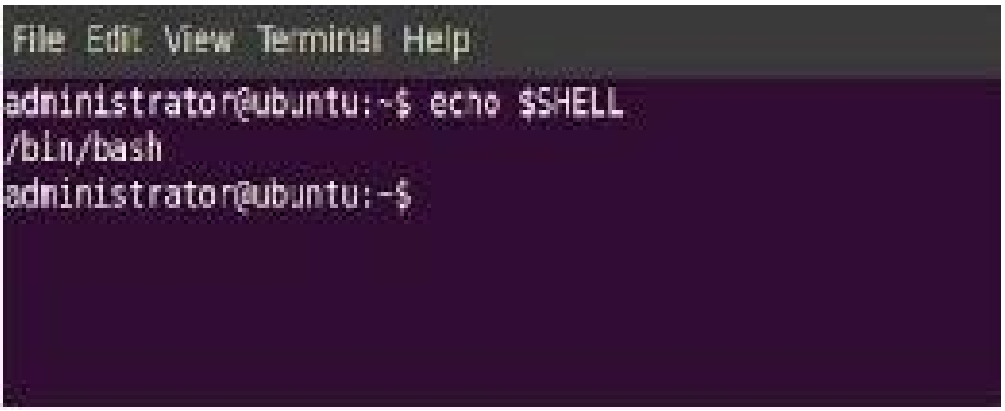
How to display default shell in the system

Type the following command in the terminal

$ echo $SHELL

And press Enter key. The default shell will be displayed as on Fig 3.

Fig 3



The Linux directory layout

Directory	Description
	The nameless base of the file system. 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 accessed 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-file system. An interface to kernel data structures
/root	The root (super user) 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.

What are Linux commands?

Linux commands are executable binary files which can be ran to perform certain tasks, like for example listing the files in a directory running an entire graphical application. Examples of frequently used commands are ls, cd, pwd, date or cat. With the exception of executable files, there is also a category called shell built-ins, which are commands provided by the shell itself (Bash in our case). We'll deal with those later.

The general form of a Linux command is:

command options(s) filename(s)

Which specifies a command, followed by one or more parameters, and optionally one or more files to apply it on. For example:

$ echo -e 'Hello, world!\n'

Will output the text 'Hello, world!' followed by a newline character. The **-e** parameter (also called argument, or switch in this case) tells the echo command to interpret escaped characters, like the trailing **\n**, which will add a newline after the text inside the single quotes. Ignore the leading dollar sign, it just signifies the shell prompt.

A command may or may not have arguments. An argument can be an option or a filename.

Special characters in linux operating system

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

The cd command

The cd command is used to change the current directory (i.e., the directory in which the user is currently working) in Linux and other Unix-like operating systems. It is similar to the CD and CHDIR commands in MS-DOS.

cd's syntax is

cd [option] [directory]

The items in square brackets are optional. When used without specifying any directory name, cd returns the user to the previous current directory. This provides a convenient means of toggling between two directories.

When a directory name is provided, cd changes the current directory to it. The name can be expressed as an absolute pathname (i.e., location relative to theroot directory) or as a local pathname (i.e., location relative to the current directory). It is usually more convenient to use a local pathname when changing to a subdirectory of the current directory.

As an example, the following would change the current directory, regardless of where it is on the system (because it is an absolute path), to the root directory (which is represented by a forward slash):

cd /

Likewise, the following would change the current directory, regardless of its location, to the /usr/sbin directory (which contains non-vital system utilities that are used by the system administrator):

cd /usr/sbin

If a user currently in the directory /usr/local/share/man/ desired to change to the directory /usr/local/share/man/man2, which is a subdirectory of the current directory, it would be possible to change by using the absolute pathname, i.e.,

cd /usr/local/share/man/man2

However, it would clearly be much less tedious to use the relative pathname, i.e.,

cd man2

On Unix-like operating systems the current directory is represented by a singledot and its parent directory (i.e., the directory that contains it) is represented by two consecutive dots. Thus, it is possible (and often convenient) to change to the parent of the current directory by using the following:

cd ..

Another convenient feature of cd is the ability for any user to return directly to its home directory by merely using a tilde as the argument. A home directory, also called a login directory, is the directory on a Unix-like operating system that serves as the repository for a user's personal files, directories and programs. It is also the directory that a user is first in after logging into the system. A tilde is a short, wavy, horizontal line character that represents the home directory of the current user. That is, any user can return immediately to its home directory by typing the following and then pressing the Enter key:

cd ~

This is easier than typing the full name of the user's home directory, for instance, /home/josephine in the case of a user named josephine. (And it is just one of the numerous shortcuts that help make the command line on Unix-like operating systems so easy to use.)

When followed by a space and then a hyphen, cd both returns the user to the previous current directory and reports on a new line the absolute pathname of that directory. This can further enhance the already convenient toggling capability of cd. Toggling is particularly convenient when at least one of the two directories has a long absolute pathname, such as /usr/local/share/man/man2.

cd has only two options, and neither of them are commonly used. The -P option instructs cd to use the physical directory structure instead of following symbolic links. The -L option forces symbolic links to be followed.

The pwd command

The pwd command reports the full path to the current directory.

The current directory is the directory in which a user is currently operating while using a command line interface. A command line interface is an all-text display mode and it is provided via a console (i.e., a display mode in which the entire screen is text only) or via a terminal window (i.e., a text-only window in a GUI).

The full path, also called an absolute path, to a directory or file is the complete hierarchy of directories from the root directory to and including that directory or file. The root directory, which is designated by a forward slash (/), is the base directory on the filesystem (i.e., hierarchy of directories), and it contains all other directories, subdirectories and files on the system. Thus, the full path for any directory or file always begins with a forward slash.

pwd is one of the most basic commands in Linux and other Unix-like operating systems, along with ls, which is used to list the contents of the current directory, andcd, which is used to change the current directory.

pwd's syntax is

pwd [option]

Unlike most commands, pwd is almost always used just by itself, i.e.,

Pwd

That is, it is rarely used with its options and never used with arguments (i.e., file names or other information provided as inputs). Anything that is typed on the same line after pwd, with the exception of an option, is ignored, and no error messages are returned.

As an example, if a user with the username janis is in its home directory, then the above command would typically return /home/janis/ (because, by default, all home

directories are located in the directory /home). Likewise, if a user were currently working in directory /usr/share/config (which contains a number of programconfiguration files), then the same command would return /usr/share/config.

pwd is useful for confirming that the current directory has actually been changed to what the user intended after using cd. For example, after issuing the cd command to change the current directory from /home/janis to /usr/share/config, pwd could be used for confirmation; that is, the following sequence of commands would be issued:

cd /usr/share/config/

pwd

The standard version of pwd has a mere two options, both of which are employed only infrequently. The --help option is used as follows:

pwd --help

This option displays information about pwd, of which there is very little because it is such a simple command (i.e., it only has two options and accepts no arguments).

The other option is --version, which displays the version number, i.e.,

pwd --version

Although it is often thought of as standing for present working directory, pwd is actually an acronym for print working directory. The word print is traditional UNIXterminology for write or display, and it originated when computer output was typically printed on paper by default because CRT (cathode ray tube) display monitors were not yet widely available.

The echo command

echo is a built-in command in the bash and C shells that writes its arguments to standard output.

A shell is a program that provides the command line (i.e., the all-text display user interface) on Linux and other Unix-like operating systems. It also executes (i.e., runs) commands that are typed into it and displays the results. bash is the default shell on Linux.

A command is an instruction telling a computer to do something. An argument is input data for a command. Standard output is the display screen by default, but it can be redirected to a file, printer, etc.

The syntax for echo is

$ echo $USER

$ echo "Hello world"

The items in square brackets are optional. A string is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks).

When used without any options or strings, echo returns a blank line on the display screen followed by the command prompt on the subsequent line. This is because pressing the ENTER key is a signal to the system to start a new line, and thus echo repeats this signal.

When one or more strings are provided as arguments, echo by default repeats those stings on the screen. Thus, for example, typing in the following and pressing the ENTER key would cause echo to repeat the phrase This is a pen. on the screen:

echo This is a pen.

It is not necessary to surround the strings with quotes, as it does not affect what is written on the screen. If quotes (either single or double) are used, they are not repeated on the screen.

Fortunately, echo can do more than merely repeat verbatim what follows it. That is, it can also show the value of a particular variable if the name of the variable is preceded directly (i.e., with no intervening spaces) by the dollar character ($), which tells the shell to substitute the value of the variable for its name.

For example, a variable named x can be created and its value set to 5 with the following command:

x = 5

The value of x can subsequently be recalled by the following:

echo The number is $x.

Echo is particularly useful for showing the values of environmental variables, which tell the shell how to behave as a user works at the command line or in scripts(short programs).

For example, to see the value of HOME, the environmental value that shows the current user's home directory, the following would be used:

echo $HOME

Likewise, echo can be used to show a user's PATH environmental variable, which contains a colon-separated list of the directories that the system searches to find the executable program corresponding to a command issued by the user:

echo $PATH

echo, by default, follows any output with a newline character. This is a non-printing (i.e., invisible) character that represents the end of one line of text and the start of the next. It is represented by \n in Unix-like operating systems. The result is that the subsequent command prompt begins on a new line rather than on the same line as the output returned by echo.

The -e option is used to enable echo's interpretation of additional instances of the newline character as well as the interpretation of other special characters, such as a horizontal tab, which is represented by \t. Thus, for example, the following would produce a formatted output:

echo -e "\n Projects: \n\n\tplan \n\tcode \n\ttest\n"

(The above command should be written on a single line, although it may render as two lines on smaller display screens.) The -n option can be used to stop echo from adding the newline to output.

By making use of output redirection, echo provides a very simple way of creating a new file that contains text. This is accomplished by typing echo followed by the desired text, the output redirection operator (which is a rightward pointing angle bracket) and finally the name of the new file. The file can likewise be formatted by using special characters. Thus, for example, the formatted output from the above example could be used to create a new file called project1:

echo -e "\n Project1: \n\n\tplan \n\twrite \n\ttest\n" > project1

The contents of the new file, including any formatting, can be verified by using a command such as cat or less, i.e.,

less project1

echo can likewise be a convenient way of appending text to the end of a file by using it together with the the append operator, which is represented by two consecutive rightward pointing angle brackets. However, there is always the risk of accidentally using a single bracket instead of two, thereby overwriting all of the contents of the file, and thus, this feature is best reserved for use in scripts.

echo can also be used with pattern matching, such as the wildcard character, which is represented by the star character. For example, the following would return the phrase The gif files are followed by the names of all the .gif image files in the current directory:

echo -e The gif files are *.gif

The cal command

Displays calendar of current month.

$ cal

July 2012

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

'cal ' will display calendar for specified month and year.

$ cal 08 1991

August 1991

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Date command

Display current time and date.

$ date

Fri Jul 6 01:07:09 IST 2012

If you are interested only in time, you can use 'date +%T' (in hh:mm:ss):

$ date +%T

01:13:14

tty command

Displays current terminal.

$ tty

/dev/pts/0

whoami command

This command reveals the current logged in user.

$ whoami

raghu

id command

This command prints user and groups (UID and GID) of current user.

$ id

uid=1000(raghu) gid=1000(raghu)

groups = 1000 (raghu), 4(adm), 20(dialout), 24(cdrom), 46(plugdev), 112(lpadmin), 120(admin),122(sambashare)

By default information about current user is displayed. If another username is provided as an argument, information about that user will be printed:

$ id root

uid=0(root) gid=0(root) groups=0(root)

Clear command

This command clears the screen.

Getting help command

For all its advantages, a big disadvantage of command line is that there are a lot of commands and even more are their options and usage. But nobody can remember all commands. There are some smarter ways of using command line. Linux provides us with several such resources discussed here:

--help option

With almost every command, '--help' option shows usage summary for that command.

$ date --help

Usage: date [OPTION]... [+FORMAT]

or: date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

Display the current time in the given FORMAT, or set the system date.

The whatis command

The whatis command provides very brief descriptions of command line programs (i.e., all-text mode programs) and other topics related to Linux and other Unix-like operating systems.

It accomplishes this by searching the short descriptions in the whatis database for each keyword provided to it as an argument (i.e., input data). This database contains just the title, section number and description from the NAME section of each page in the man manual that is built into most Unix-like systems.

The syntax for whatis is:

whatis keyword(s)

For example, the following provides a single line summary of the headcommand (which by default displays the first ten lines of each file that is provided to it as an argument):

whatis head

whatis can be used to simultaneously search for information about multiple topics. For example, the following would provide information about both head and tail (which by default reads the final ten lines of files):

whatis head tail

The output of whatis is limited by the fact that it provides only a single line for each keyword found in the database; thus it supplies incomplete information about even moderately complex commands. For example, the following use of whatis to obtain information about the cat command generates the output "concatenate files and print on the standard output":

whatis cat

However, this omits some important information about cat, particularly the facts that it is very convenient to use for reading files and that it is also used to create and write to files.

whatis is similar to the apropos command. However, apropos is more powerful in that its arguments are not limited to complete words but can also be strings (i.e., any finite sequences of characters) which comprise parts of words. Both commands are unusual in that they have no options.

The man command (which is used to read the built-in manual pages), when used with its -f option, produces the same output as whatis. Thus, for example,

man -f cat

is equivalent to

whatis cat

Info pages

Info documents are sometimes more elaborated than man pages. But for some commands, info pages are just the same as man pages. These are like web pages. Internal links are present within the info pages. These links are called nodes. info pages can be navigated from one page to another through these nodes.

$ info date

Word processors in the Linux environment

Text editors are used by many different types of people. Casual users, writers, programmers, and system administrators will all use a text editor at one time or another in Linux.

Use of text editor

A text editor is just like a word processor without a lot of features. All operating systems come with a basic text editor. Linux comes with several. The main use of a text editor is for writing something in plain text with no formatting so that another program can read it. Based on the information it gets from that file, the program will run one way or another.

vi Editor

"vi" (pronounced "vee eye") is a text editor with a deceptively simple appearance that belies its great power and efficiency. New users soon realize that there is far more to this little program than meets the eye.

vi, or one of its clones, is found in almost every version of Linux and Unix, and, in fact, it is the only editor that is available in virtually every Unix installation.

History of vi

The vi editor was developed starting around 1976 by Bill Joy, who was then a graduate student at the University of California at Berkeley. Joy later went on to help found Sun Microsystems and became its Chief Scientist.

"ed" was the original Unix text editor. Like other early text editors, it was line oriented and used from dumb printing terminals. Joy first developed "ex" as an improved line editor that supported a superset of ed commands. He then developed vi as a "visual interface" to ex. That is, it allows text to be viewed on a full screen rather than only one line at a time. vi takes its name from this fact.

vi remains very popular today in spite of the development and widespread availability of GUI (graphical user interface) mode text editors which are far more intuitive and much easier for beginners to use than text-mode text editors such as vi. GUI-mode text editors include gedit and Emacs, both of which have become very common on Linux and other Unixes today.

Features of vi

- It is present in almost every Linux Unix system, even the most minimal.
- It is very small. In fact, some versions have a total code size of less than 100KB. This makes it easy to include vi on even the tiniest versions of Linux, such as those in embedded systems and those that run from a single floppy disk.
- It is typist-friendly, at least once you get used to it. For example, the commands are very short, usually just a few keystrokes. And because vi does not use the mouse, there is never any need to remove one's hands

from the keyboard. This can speed up editing substantially.

- It is very powerful, as just a few very short commands can make sweeping changes to large documents. In fact, vi is more powerful than most of its users realize, and few of them know more than just fraction of all the commands.

Opening and closing file

vi can be used both when your system is in text mode (the entire screen is devoted to text and there are no images) and when your system is in GUI mode (the screen contains windows, images and menus). When it is in GUI mode (usually KDE or Gnome), vi runs in a terminal window. A terminal window is a text-only window, and it can usually be opened by clicking on an icon (small image) of a computer screen.

(In the case of Red Hat Linux, the terminal window can be opened by clicking on the icon of a red hat in the lower left hand corner of the screen, opening the System Tools menu and then selecting Terminal from that menu. It can be convenient to add the icon for the terminal window to the launcher panel along the bottom of the screen, if it is not already there.)

There are at least two ways to use vi to simultaneously create and open a new file. One is by just typing vi at the command line, like this:

vi

This creates an empty file that will not have a name until you save its contents to disk (i.e., transfer the text you typed into it to your hard disk, floppy disk, etc. for long term storage).

A second way to open a new file is by typing vi followed by the name of the file to be created, for example:

vi apple

This creates a new file named "apple" in the current directory (the directory or folder which is currently open on your all-text screen or your terminal window).

If you want, it could create the same file with an extension such as ".txt" added to the end of the file name. In Linux this is merely a matter of convenience (or habit), and it generally makes no real difference for the file because it remains a plain text file in either case. For example:

vi apple.txt

To close a file to which no changes have been made, hit ESC (the Esc key, which is located in the upper left hand corner of the keyboard), then type :q (a colon followed by a lower case "q") and finally press ENTER. (The term "hit" is used here instead of "press" to emphasize that it is not necessary to keep the ESC key held down but just to press it momentarily.)

To close a file to which changes have been made (such as text having been added or removed) without saving the changes, hit ESC, type :q! and then press ENTER. This is sometimes referred to as a "forced quit."

vi works with a buffer (a block of memory in the RAM chips). When you open an existing file, vi copies that file from the hard disk (or floppy, CDROM, etc.) to a buffer. All changes that you make to a file are initially made only to the copy in the buffer, and they are only made to the file itself when you "save" your changes. "Saving" a file means writing (i.e., transferring) the contents of the buffer to the hard disk (or floppy disk).

Likewise when you open a new file. All text you enter (and subsequent edits you make to it) exists only in the buffer until you save the file to disk.

To save the changes that have been made to a file, hit ESC, type :qw and then press ENTER. The "w" stands for "write." An alternative, and perhaps easier, way to save a file and quit at the same time is to hit ESC and then type ZZ (two capital Z's in succession).

After you have created a new text file and closed it, you might want to confirm that nothing went wrong and that the file actually exists. Probably the simplest way to do this is to use the standard Unix ls command, which displays a list of all of the files in the current directory.

Entering text

vi has two basic modes of operation: command mode and text insert mode. How to switch back and forth between them is probably the most confusing thing about vi for beginners. But it is actually very simple, and once you get used to it you might also find it quite efficient.

Command mode is the default mode when a file (existing or new) is opened. (This is the opposite of most text and word processors and therefore may seem counter-intuitive.) Because every file opens initially in command mode, you can not immediately begin typing text. That is, everything that is typed on the keyboard is interpreted by vi to be a command.

Examples of the many types of commands can perform on a file while in command modes are:-

- Switching to text insert mode.
- Moving the cursor around the file.
- Deleting characters or lines.
- Transposing characters.
- Changing case.
- Appending the contents of the file to another (closed) file.
- Setting vi options.
- Saving the file to disk.
- Closing the file and quitting vi.

The other mode, text insert mode, is also referred to as simply "insert mode" or "input mode." It is used for entering text into the buffer memory (and simultaneously onto the screen). In this mode everything that is typed on the keyboard is added to the text and does not become a command (although you can perform some command operations in text mode with vi clones).

The most common way to switch from command mode to the input mode is to use the i (which stands for "insert" or "input") command. This is accomplished by simply typing the letter i while in command mode. Now you are ready to start typing text.

Unlike word processors and even most word editors, there is no automatic word wrap in the traditional version of vi (although you will notice it in some clones). New lines are started by pressing ENTER.

When finished typing text or need to perform some other operation such as moving to a different position in the text or deleting some of it, hit ESC in order to return to the command mode.

Once you have typed some text, you can use the four basic commands for moving the cursor around the text. These commands enable you to go to any desired location in order to modify the text, including making insertions and deletions. The four basic cursor positioning commands are:

h move cursor one character to left

j move cursor one line down

k move cursor one line up

l move cursor one character to right

Each of these commands can be either used by itself or modified by typing an integer in front of it to indicate the number of characters or lines to move. For example, typing (in command mode, of course)

3j - will move the cursor down three lines. Or typing 2h will move it two characters to the left.

These commands can be repeated by holding the key down. If attempting an impossible movement, such as pressing k when the cursor is on the top line, the screen might flash or a beeping sound might be made (depending on how your computer is set up).

The cursor can be moved directly to any desired line by using the G command preceded by the line number. For example, typing

5G - moves the cursor to the fifth line from the top of the text. Just typing G without any number moves the cursor to the final line of text.

When you switch from command mode to input mode with the i command and then start typing text, each character you type is placed to the left of the character covered by the cursor. This causes the character covered by the cursor as well as everything to its right to be shifted to the right.

There will be times when it need to place a character to the right of the character under the cursor. This is particularly useful when the cursor is over the last character in a line and you want to append the line. To do this, simply use the a (lower case "a," which stands for "append") command instead of the i command to switch from command mode into insert mode.

After it have saved a file that have created or modified using vi, might want to verify that its contents are really what you had intended. One way to do this is to use cat, the Unix concatenation utility. (No, this has no relationship to the popular domesticated animal whose name has the same spelling). For example, type:

cat /home/john/fruit/lemon

Editing Text

vi offers a rich assortment of commands for editing text. Among the most basic are those used for deleting or erasing.

The x (lower case "x") command deletes the character immediately under (i,e., covered by) the cursor. To delete any desired character, just switch to the command mode (if you are not already there) and then use an appropriate combination of the h, j, k and l commands (of course, one at a time) to move the cursor to that character. Then type x and the character is deleted.

By pressing x continuously instead of just hitting it once, the cursor continuously moves to the right and each character under it is successively deleted.

The X (upper case "X") command is similar except that it deletes the character to the left of the cursor rather than the character under it.

There are several additional commands for deleting text. The D (upper case "D") command removes the text on the current line from the character under the cursor to the end of the line.

The d (lower case "d") command is very flexible because it can be modified to delete any number of characters, words or lines. Typing d by itself will not do anything, but typing dw causes the character the cursor is resting on and the remaining characters to the right of it in the same word to be deleted. (The "w" stands for "word.")

Typing 2dw causes the character under the cursor, the remaining characters to the right of it in the same word and all of the characters in the next word to be deleted. For example, typing 2dw with the cursor on the "a" of the string "pineapple plantation" causes the string "apple plantation" to be deleted.

As another example, typing 3dw with the cursor on the "j" of the string "the bluejay flew south" causes the string "jay flew south" to be deleted. That is, "jay" and two words to the right of it are deleted.

Deleting an entire line can be accomplished with the dd command. This command can also be used to delete multiple lines by preceding it with an integer representing the number of lines to be removed. For example, typing

2dd will delete two consecutive lines beginning with the current line.

With some terminals, deletion of a line causes it to be replaced on the screen with an "@" character. This character merely represents an empty line and is not inserted into the text. Its purpose is to relieve the processor from having to redraw the screen (i.e., change the whole screen). This character can be removed if desired by typing r (or l on some terminals) while holding down the CTRL key.

The change command c (lower case "c") differs from the delete command in that it not only deletes a section of text but also activates insert mode to allow you to type in replacement text. After you have completed typing in the replacement text, be sure to press ESC to return to the command mode.

As is the case with d, the c command is not used by itself but is only used in combination with another letter after it and an optional integer before it.

For example, the command cw (which stands for "change word") deletes the characters in the current word under and to the right of the cursor and then switches vi to the insert mode so that you can enter text to replace the deleted characters. The number of new characters typed in can be the same as, fewer or more than the number deleted.

The amount of text to be changed can be increased by preceding the command with a number. For instance, typing 2cw will additionally remove the next word for replacement with whatever is typed in. The space between the words is not preserved.

The d and c commands can also be modified by other characters in addition to "w." For example they can be used with "b," which stands for "back." Thus, typing3bd will delete the characters to the left of the cursor in the current word together with the two words to the left of the current word.

The cc command erases the current line, leaving it blank and awaiting replacement text. Preceding this command with an integer will delete that number of lines, beginning with the current line. For example, typing 5cc will allow you to change five consecutive lines starting with the current line.

Another change command, R, differs from the c commands in that it does not initially delete anything. Rather, it activates insert mode and lets you replace the characters under the cursor one at a time with characters that you type in.

vi supports several types of transposition. Transposing the order of two adjacent characters is easy with the xp command. Just place the cursor on the left-most of the two characters, type x to erase the left character and then type p for the deleted character to be put to the right of the cursor.

Two adjacent words can be transposed with the deep command. To use it, position the cursor in the space just to the left of the word on the left and type deep. Two adjacent lines can be transposed with the ddp command by placing the cursor on the upper line and typing ddp.

It is also a simple matter to change the case of a letter. When the cursor is over the desired letter, hit the "~" (tilde) key. This will change a capital letter to a small letter and visa versa.

The J (upper case "J") command is used to join the next line to the current line. The opposite operation, splitting a line, is accomplished in insert mode by merely positioning the cursor over what will be the first character of the new line and then hitting ENTER.

vi also has an undo capability. The u (lower case "u") command is used to reverse the effects of an already issued command that has changed the buffer, but which is not yet written to disk. U (upper case "U") undoes all of the changes that have been made to the current line during your current visit to it

Searching Text

vi also has powerful search and replace capabilities. To search the text of an open file for a specific string (combination of characters or words), in the command mode type a colon (:), "s," forward slash (/) and the search string itself. What you type will appear on the bottom line of the display screen. Finally, press ENTER, and the matching area of the text will be highlighted, if it exists. If the matching string is on an area of text that is not currently displayed on the screen, the text will scroll to show that area.

The formal syntax for searching is:

:s/string

For example, suppose you want to search some text for the string "cherry." Type the following and press ENTER:

:s/cherry

The first match for "cherry" in your text will then be highlighted. To see if there are additional occurrences of the same string in the text, type n, and the highlight will switch to the next match, if one exists.

The syntax for replacing one string with another string in the current line is

:s/pattern/replace/

Here "pattern" represents the old string and "replace" represents the new string. For example, to replace each occurrence of the word "lemon" in a line with "orange," type:

:s/lemon/orange/

The syntax for replacing every occurrence of a string in the entire text is similar. The only difference is the addition of a "%" in front of the "s":

:%s/pattern/replace/

Thus repeating the previous example for the entire text instead of just for a single line would be:

:%s/lemon/orange/

Working with multiple files

It is easy to insert text into an open file from another file. All that is necessary is to move the cursor to the location where you want the text inserted, then type

:r filename

where "filename" is the name of the file to insert.

For example, if you want to copy the contents of the file "peach" into the file "fruit," you would first position the cursor to the desired line in "fruit" and then type

:r peach

Notice that this operation causes no change to the file "peach."

You can also append text from the currently open file to any other file. This is accomplished using the :w (colon + "w") command followed without a space by >>. For example, to append the contents of a currently open file named "pear" to the file named "apple," type

:w>> apple

At times it can be convenient to open multiple files simultaneously. This is efficiently accomplished by just listing all of the files to be opened after the vi command. For example, to simultaneously open files about three kinds of fruit, type:

vi apple pear orange

This allows you to edit "apple" first. After saving "apple," typing :n calls up "pear" for editing.

If you want to simultaneously open all files in the current directory, just type vi * (vi + space + asterisk).

Additional operations

As you have learned, creating and opening files in vi can be a very simple matter. However, many combinations of options are available that can add much power and flexibility for these tasks, as can be seen by looking at the full syntax for opening files:

vi [flags] [cmd] [filename]

The square brackets ([ ]) around each section of arguments (modifiers) of the command indicates that they are optional. (That is, a file can be opened by just typingvi alone or by typing it with any combination of the three arguments. For instance, the example of vi dog contains only the mandatory vi and the optional third argument, which is the name of the file to open.)

As only one of many possible examples of adding options for opening files, an existing file can be opened with the cursor appearing on any desired line instead of just on the first line. (One situation in which this can be particularly useful is if your file is part of a program which you are writing and the compiler reports an error on a specific line in that file.) This is accomplished by adding the + (plus sign) command followed the desired line number. For example, to open the file "apple" with the cursor located on the third line, type:

vi +3 apple

Use of the + command without any modifying number opens a file with the cursor positioned on the last line of text. This can save some keystrokes when you want to open a file just to append data to the end of it. For example:

vi + apple

You have already learned several commands for switching from command mode to insert mode, including i for inserting to the left of the cursor position, a for inserting to the right of the cursor position and the c commands for changing text. A more complete list is as follows:

a	appends after current cursor position.
A	appends at end of current line.
c	starts a change option.
C	starts a change option from current position to end of current line.
i	inserts to the left of the cursor position.
I	inserts at start of line.
o	cursor moves to new, blank line below its current position.
O	cursor moves to new, blank line above its current position.
R	replaces characters one at a time.

A simple way to obtain basic information about any file that is currently open, including name, size and the current line number, is to hold down CTRL and type g. This data appears on the bottom line of the display.

Summary of commands

The following list contains the basic commands presented in the first eight pages of this tutorial along with occasional examples of usage (shown in parenthesis). They are presented in roughly the same order in which they appear in the tutorial. (All commands that begin with a colon are followed by ENTER.)

vi	typed at the command line to open one or more files in the same directory (vi tomato.txt opens a file named "tomato.txt" in the current directory) (vi parsley sage rosemary opens the three files "parsley," "sage" and "rosemary" in the current directory)
vi *	typed at the command line to open every file in the current directory
:q	closes (quits) a file to which no changes have been made
:q!	quits without saving any changes
:w	writes (i.e., saves) the current file to disk
:wq	writes the buffer contents to disk (i.e., saves changes) and quits
ZZ	same as **:wq**
i	activates text insert mode, inserting text immediately under the current position of the cursor.
h	moves the cursor one character to the left (2h moves the cursor two characters to the left)
j	moves the cursor one line down (3j moves the cursor three lines down)
k	moves the cursor one line up
l	moves the cursor one character to the right
G	moves the cursor to the desired line; moves the cursor to the last line of text if not preceded by a modifying integer (5G moves the cursor to the fifth line)
a	switches to insert mode and allows insertion of text immediately to the right of the cursor.
x	deletes the character immediately under the cursor (xxx deletes the character immediately under cursor and then deletes the two characters to its right)
X	deletes a single character to the left of cursor
D	removes the text on the current line from the character under the cursor to the end of the line
dw	deletes the character immediately under the cursor and the remaining characters to the right of it in the same word (2dw deletes the character immediately under the cursor, the remaining characters to the right of it in same word and all of the next word)
dd	deletes the entire line containing the cursor, and the cursor then moves to the next line (2dd deletes two consecutive lines beginning with the current line)
cw	deletes the character under the cursor and to its right in the same word and allows new characters to be typed in to replace them (2cw deletes the character under the cursor and to its right in the same word and in the next word, and then allows replacement characters to be typed in)

cc	erases the current line and allows replacement text to be typed in (2cc erases the current line and the next line and allows replacement text to be typed in for both lines)
cb	deletes the characters to the left of the cursor in the current word and allows replacement characters to be typed in (3cb deletes the characters to the left of the cursor in the current word together with the two words to its left and then allows replacement text to be typed in)
R	activates text input mode allowing text under and to the right of the cursor to be overwritten one character at a time
xp	transposes two adjacent characters
deep	transposes two adjacent words
ddp	transposes two adjacent lines
~	changes case of the character under the cursor
J	joins the current line with the next line
u	reverses the effects of the most recent command that has changed the buffer
U	undoes all changes made to the current line during the current visit to it
:s/	searches the text for the first instance of a designated string (:s/cucumber searches the text for the first instance of the string "cucumber")
n	searches the text for the next instance of a designated string
:s/ / /	replaces the first instance of a designated string (:s/cucumber/radish/ replaces the first instance of the string "cucumber" with the string "radish")
:%s/ / /	replaces every instance of a designated string (:%s/cucumber/radish/ replaces every instance of the string "cucumber" with the string "radish")
:r	inserts text into the currently open file from another file (:r lettuce.txt inserts text into the currently open file from the file named "lettuce.txt")
:w>>	appends the text from the currently open file into another file (:w>> cabbage appends the text from the currently open file into the file named "cabbage")

pico editor

pico is a simple text editor in the style of the pine composer.

Syntax

pico [options] [file]

Description

pico is a simple, display-oriented text editor based on the pine message composer. As with pine, commands are displayed at the bottom of the screen, and context-sensitive help is provided. As characters are typed they are immediately inserted into the text.

Editing commands are entered using control-key combinations. As a work-around for communications programs that swallow certain control characters, you can emulate a control key by pressing ESCAPE twice, followed by the desired control character. For example, "ESC ESC c" would be equivalent to entering a ctrl-c. The editor has five basic features: paragraph justification, searching, block cut/paste, a spelling checker, and a file browser.

Paragraph justification (or filling) takes place in the paragraph that contains the cursor, or, if the cursor is between lines, in the paragraph immediately below. Paragraphs aredelimited by blank lines, or by lines beginning with a space or tab. Unjustification can be done immediately after justification using the control-U key combination.

String searches are not sensitive to case. A search begins at the current cursor position and wraps around the end of the text. The most recent search string is offered as the default in subsequent searches.

Blocks of text can be moved, copied or deleted with creative use of the command for mark (Ctrl-^), delete (Ctrl-k) and undelete (Ctrl-u). The delete command will remove text between the "mark" and the current cursor position, and place it in the "cut" buffer. The undelete command effects a "paste" at the current cursor position.

The spell checker examines all words in the text. It then offers each misspelled word for correction while highlighting it in the text. Spell checking can be cancelled at any time. Alternatively, pico will substitute for the default spell checking routine a routine defined by the SPELL environment variable. The replacement routine should read standard input and write standard output.

The file browser is offered as an option in the "Read File" and "Write Out" command prompts. It is intended to help in searching for specific files and navigating directory hierarchies. Filenames with sizes and names of directories in the current working directory are presented for selection. The current working directory is displayed on the top line of the display while the list of available commands takes up the bottom two. Several basic file manipulation functions are supported: file renaming, copying, and deletion.

Movement commands:

Depending on your system, the arrow keys or the backspace key may not work. Instead, you can use these commands to perform the same tasks.

To	Hold down Ctrl key and press	Instead of
Delete a character	backspace	backspace
Move up a line	p	up arrow
Move down a line	n	down arrow
Move left one space	b	left arrow
Move right one space	f	right arrow
Move to the end of line	e	end

Some pico editor options

^C Cancel allows you to stop a process at any time. If you make a mistake, just hold down the Ctrl key and press c.

^G get help

Get clear and concise assistance from the Pico help, in case something unexpected happens or you need additional information about a command.

^X Exit

Exit Pico at anytime. If made changes to a file or worked on a new file, but you havent saved the changes, you see this message:

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) (y/n)?

Answering no (press n) will close Pico and bring you back to the prompt without saving your file.

Answering yes (press y) will allow you to save the file you've been working on (see Write Out section below for details).

^O WriteOut

Save the file without hassles or worries. Fill in the name of the file beside the File Name to write: prompt. If the file already has a name, then press enter.

^T To Files option lets to save the text over a file that exists in the directory. By choosing the To Files option, Pico takes you to a directory Browser.

^R Read File

Insert text from another file into your current text file. This option allows you to search through your directories for a file that you would like to add to your text. This option is especially handy if you've saved a document and would like to add its content to the new file you're working on. Text from the file you select is placed on the line directly above your cursor.

At the Insert file : prompt you may either type a file name or use the Browser options.

^T To Files option lets you import a text file directly into the file you're currently typing. By choosing the To Files option, Pico takes you to a directory Browser.

^Y Prev Pg

Move quickly to the previous page. Although you could just as easily press the up arrow key several times, this command quickly jumps your cursor up one page.

^V Next Pg

Move quickly to the next page. Although you could just as easily press the down arrow key several times, this command quickly jumps your cursor down one page.

^K Cut text

Cut a line of text. This option allows you to cut a full line of text. By using the uncut command and your arrow keys, you can then paste the cut text at another location in your document. To cut specific text in a line or to cut several lines of text, first select the text (see Selecting Text on the next page).

Selecting text

To select text for cutting and pasting use the following steps:

Move the cursor to the beginning of the text to select

Hold down the Ctrl key and press ^

Use the right arrow key or hold down Ctrl and press f to highlight text

When you have highlighted the appropriate text, hold down the Ctrl key and press k to cut it.

Paste the text you cut, anywhere in your document, using UnCut Text

^U UnCut Text

Paste text that previously cut. if use this option to undo an accidental cut of text or place cut text at another location in the document. The text you cut is pasted on the line directly above the cursor.

^C Cur Pos

Indicate the current position of the cursor, relative to the entire document. This is a helpful option if you'd like to check exactly where in the document. The status line indicates the following items:

[line 8 of 18 (44%), character 109 of 254 (42%)]

^J Justify

Even out lines of text. This command is handy when accidentally type extra spaces between words or press the key before reaching the end of a line. The option evens the length of text lines automatically.

^U UnJustify

UnJustify lines of text. For the messy line look you can always select the UnJustify option.

^W Where is

Find a particular string of text quickly. This option allows you to do a word search in your text. This option is especially handy for longer documents. If the word you designated at the Search: prompt is found, it places the cursor beside it.

^T To Spell

Check for spelling errors. The spell check option allows to correct spelling errors throughout the document. If spell

checker finds a misspelled word or a word it doesn't recognize (don't worry, this rarely happens), it will correct the word. At the Edit a replacement: prompt, type in the correct spelling of a word. However, if you don't want to make any changes, simply press the enter key.

Any words that have corrected but re-occur in the document can be automatically replaced. At the Replace a with b? [y]: prompt press y to replace all occurrences of the misspelled word or n to ignore.

Pine Editor

pine is a program for accessing email and newsgroups.

Syntax

pine [options] [address, address]

Description

pine is a screen-oriented message-handling tool. In its default configuration, pine offers an intentionally limited set of functions geared toward the novice user, but it also has a growing list of optional power-user and personal-preference features. pine's basic feature set includes:

- View, Save, Export, Delete, Print, Reply and Forward messages.
- Compose messages in a simple editor (pico) with word-wrap and a spelling checker. Messages may be postponed for later completion.
- Full-screen selection and management of message folders.
- Address book to keep a list of long or frequently-used addresses. Personal distribution lists may be defined. Addresses may be taken into the address book from incoming mail without retyping them.
- New mail checking and notification occurs automatically.
- Context-sensitive help screens.

pine supports MIME (Multipurpose Internet Mail Extensions), an Internet Standard for representing multipart and multimedia data in email. pine allows you to save MIME objects to files, and in some cases, can also initiate the correct program for viewing the object. It uses the system's mailcap configuration file to determine what program can process a particular MIME object type. pine's message composer does not have multimedia capability itself, but any type of data file (including multimedia) can be attached to a text message and sent using MIME's encoding rules. This allows any group of individuals with MIME-capable mail software to exchange formatted documents, spread-sheets, image files, etc, via Internet email.

pine uses the "c-client" messaging API to access local and remote mail folders. This library provides a variety of low-level message-handling functions, including drivers for a variety of different mail file formats, as well as routines to access remote mail and news servers, using IMAP (Internet Message Access Protocol) and NNTP (Network News Transport Protocol). Outgoing mail is usually handed off to the send mail program but it can optionally be posted directly via SMTP.

Examples

Pine

Launch **pine.**

pine address@example.com

Launch pine, and immediately begin composing an email addressed to address@example.com.

Joe editor

'joe'- sounds like a comic strip. Actually, they are two other text editors that I like and I think are a little easier to manage. They're like 'vi' in that you use them to create and edit non-formatted text, but they're a little more user-friendly. Using 'joe' 'joe' was created by Joseph Allen, so that's why it's called Joe.

The majority of joe's commands are based on the CTRL-K keys and a third key. The most important of these is CTRL-K-H which gets 'help'. Help shows the key combinations to use with 'joe'.

The most important thing about 'joe' is the logical concept that you can just start writing if you want. Try writing anything you want.

To save it, press CTRL-K-D. To save and quit, CTRL-K-X.

To quit without saving, CTRL-C, (without the K).

The feature of 'joe' is that if edit a file again, it will save the previous file with a tilde on the end, like 'tryjoe~' That little tilde file has saved times. 'joe' is a very good option for writing those short text files.