DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

OPERATING SYSTEM LAB

LIST OF EXPERIMENTS

1. Introduction to DOS and its external and internal commands.
2. Introduction to windows with its features.
3. Installation of operating system (windows).
4. Study of Administrative Tools provided in Control Panel (window XP,window 7,window 8, window 10).
5. Introduction to Unix /Linux Operating System with its commands.
6. Introduction to Vi Editor .
7. Introduction to Shell Programming commands.
8. Programs using shell scripts.

Experiment No 1

DOS Commands

Syntax Notes

To be functional, each DOS command must be entered in a particular way: this command entry structure is known as the command's "syntax." The syntax "notation" is a way to reproduce the command syntax in print.

For example, you can determine the items that are optional, by looking for information that is printed inside square brackets. The notation [d:], for example, indicates an optional drive designation. The command syntax, on the other hand, is how YOU enter the command to make it work.

Command Syntax Elements

1. **Command Name**

   The DOS command name is the name you enter to start the DOS program (a few of the DOS commands can be entered using shortcut names). The DOS command name is always entered first. In this book, the command is usually printed in uppercase letters, but you can enter command names as either lowercase or uppercase or a mix of both.

2. Space

Always leave a space after the command name.

3. Drive Designation

The drive designation (abbreviated here as "d:") is an option for many DOS commands. However, some commands are not related to disk drives and therefore do not require a drive designation. Whenever you enter a DOS command that deals with disk drives and you are already working in the drive in question, you do not have to enter the drive designator. For example, if you are working in drive A (when the DOS prompt A> is showing at the left side of the screen) and you want to use the DIR command to display a directory listing of that same drive, you do not have to enter the drive designation. If you do not enter a drive designation, DOS always assumes you are referring to the drive you are currently working in (sometimes called the "default" drive).

4. A Colon

When referring to a drive in a DOS command, you must always follow the drive designator with a colon (:) (this is how DOS recognizes it as a drive designation).

5. Pathname

A pathname (path) refers to the path you want DOS to follow in order to act on the DOS command. As described in Chapter 3, it indicates the path from the current directory or subdirectory to the files that are to be acted upon.

6. Filename

A filename is the name of a file stored on disk. As described in Chapter 1, a filename can be of eight or fewer letters or other legal characters.

7. Filename Extension

A filename extension can follow the filename to further identify it. The extension follows a period and can be of three or fewer characters. A filename extension is not required.

8. Switches

Characters shown in a command syntax that are represented by a letter or number and preceded by a forward slash (for example, "/P") are command options (sometimes known as "switches"). Use of these options activate special operations as part of a DOS command's functions.

9. Brackets

Items enclosed in square brackets are optional; in other words, the command will work in its basic form without entering the information contained inside the brackets.

10. Ellipses

Ellipses (...) indicate that an item in a command syntax can be repeated as many times as needed.

11. Vertical Bar

When items are separated by a vertical bar (|), it means that you enter one of the separated items. For example: ON | OFF means that you can enter either ON or OFF, but not both.

DOS Commands are divided into 2 types:

1. **Internal Commands**

   These are for performing basic operations on files and directories and they do not need any external file support.

2. **External Commands**

   These external commands are for performing advanced tasks and they do need some external file support as they are not stored in COMMAND.COM

In MS-DOS, keyboard shortcuts involving handy ones like Functional keys, arrows, pipe character (" | "), asterisk (*), ?, [] and ESC are of great help for recalling to searching to clearing command line etc., Here are few of them:

- UP (↑) and DOWN (↓) arrows recall previously entered commands.
- ESC clears the present command line. It abandons the currently construct command and the next prompt appears.
- F1 or → retypes one character at a time from the last command entry from the current cursor position.
- F2 retypes all characters from the last command entry up to the one identical to your next keystroke. It asks you to enter char to copy up to and retypes the last command up to that char.
- F3 retypes all remaining characters from the last command entry.
- F4 stores all characters beginning at the first match with your next keystroke and ending with the last command entry.
- F5 or F8 keys give all the previously typed commands.
- F6 places a special end-of-file code at the end of the currently open file. Sometimes referred to as Ctrl+z or ^z.
- F7 key displays command history and ALT+F7/ESC hides it.
- F9 is used to select a command by number. Just enter the command number and it fetches the command line for you.
- Pipe character (" | ") combines several series of commands or programs inter-dependent.
- Name enclosed within [] indicate a sub-directory.

- Asterisk (*) is used to represent zero or more any characters.
- ? is used to present zero or single character.

MS-DOS commands perform tasks like:

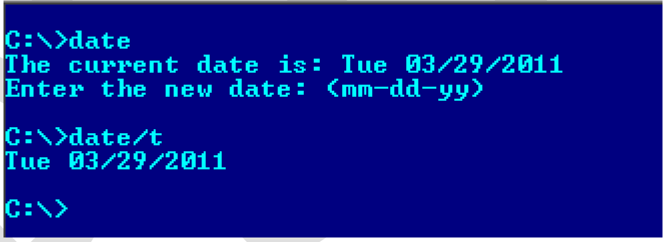
- Manage files and directories
- Maintain Disks
- Configure Hardware and Networking
- Optimize the use of memory
- Customize MS-DOS

Commonly Used Internal DOS Commands

1. DATE

This command is used to display the system current date setting and prompt you to enter a new date.

The syntax is: **DATE [/T | date]**



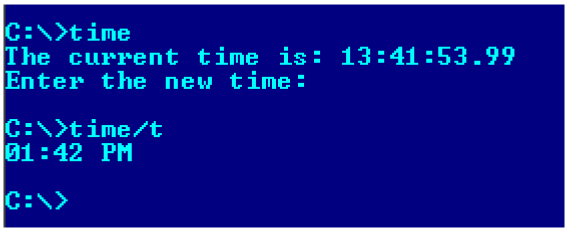
```
C:\>date
The current date is: Tue 03/29/2011
Enter the new date: (mm-dd-yy)

C:\>date/t
Tue 03/29/2011

C:\>
```

If you type DATE without parameters then it displays current date and prompts to enter new date. We should give new date in mm-dd-yy format. If you want to keep the same date just Press ENTER. DATE command with /T switch tells the command to just output the current system date, without prompting for a new date.

2. TIME

This command is used to displays or set the system time.

The syntax is: **TIME [/T | time]**



```
C:\>time
The current time is: 13:41:53.99
Enter the new time:

C:\>time/t
01:42 PM

C:\>
```

Same as DATE command, typing TIME with no parameters displays the current time and a prompt for a new one. Press ENTER to keep the same time. TIME command used with /T switch tells the command to just output the current system time, without prompting for a new time.

1. **COPY CON**

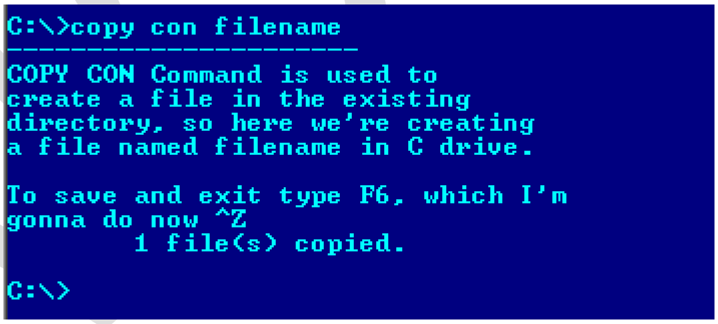
It is used to create a file in the existing directory. Here CON is a DOS reserved word which stands for console.

Syntax is: **COPY CON filename** after that press Enter and start typing your text and after you're done typing your text, to save and exit hit F6 key.

2. **TYPE**

This command is used to display the contents of a text file or files. The syntax is:**TYPE [drive:][path]filename**

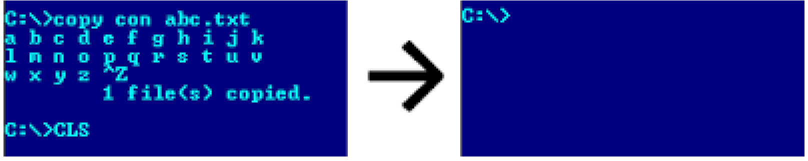
Now, lets try to display the contents of the file named filename we've created earlier using COPY CON command.



```
C:\>copy con filename
----------------------
COPY CON Command is used to
create a file in the existing
directory, so here we're creating
a file named filename in C drive.

To save and exit type F6, which I'm
gonna do now ^Z
        1 file(s) copied.

C:\>
```

3. **CLS**

It is used to clear the screen. Syntax is **CLS**



```
C:\>copy con abc.txt
a b c d e f g h i j k
l m n o p q r s t u v
w x y z ^Z
        1 file(s) copied.

C:\>CLS
```

```
C:\>
```

4. **REN**

This command is used to change/modify the name of a file or files.

Syntax is: **REN [drive:] [path] filename1 filename2.**

Here, filename1 is source file for which you wanted to change the name, and filename2 will obviously becomes your new file name. Also note that you cannot specify a new drive or path for your destination file.

5. **DIR**

This command displays a list of files and subdirectories in a directory. Syntax is:**DIR [drive:] [path] [filename] [/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N] [/O[[:]sortorder]] [/P] [/Q] [/S] [/T[[:]timefield]] [/W] [/X] [/4]**

Here,

[drive:][path][filename]	Specifies drive, directory, and/or files to list.
/A:attributes	Displays files with specified attributes. The possible attributes are as follow: D → Directories, R → Read-only files, H → Hidden files, A → Files ready for archiving, S → System files, – Prefix meaning not
/B	display in bare format with no heading information or summary
/C	Using this attribute with dir by default displays the thousandseparator in file sizes. To disable display or separator use /-C
/D	Displays file list sorted by column.
/L	Uses lowercase in listing file names and sub-directories.
/N	Display in new long list format where filenames are on the far right.
/O:sortorder	Displays list by files in sorted order. The sortorder attributes are as follow: N → By name (alphabetic), S → By size (smallest first), E → By extension (alphabetic), D → By date/time (oldest first), G → Group directories first, – Prefix to reverse order
/P	Display page wise pausing after each screenful of information and prompts to press any key to continue.

/Q	Displays the owner of a file or files.
/S	Displays files in specified directory and all subdirectories. Bear caution in using this in your root directory as you may end up in overflowing information. To stop the screen overflow at any point hit Pause-Break key.
T:timefield	This sorts and displays the list based on time field specified. C for Creation, A for Last Access, W for Last Written
/W	Displays list width wise or wide list format.
/X	This is used to display the short names generated for non-8dot3 file names.

Note that switches may be different in the DIRCMD environment variable, in which case just override present switches by prefixing any switch with – (hyphen), for example instead of using /P use /-P

6. PATH

This command displays the path that how we have come to the present position or sets a search path for executable files.

Its Syntax is **PATH [[drive:]path[;…][;%PATH%]]**

Typing PATH without any parameters displays the current path under current directory. Typing PATH ; clears all search-path settings and direct cmd.exe to search only in the current directory. And including %PATH% in the new path setting causes the old path to be appended to the new setting.

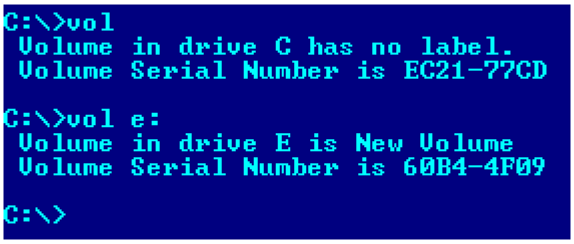
7. VER

This command displays the version of the Microsoft Windows running on your computer.

8. VOL

It displays the disk volume label and serial number, if they exist for the drive specified. If no drive is specified it displays for the active drive.

Syntax is **VOL [drive:]**



```
C:\>vol
 Volume in drive C has no label.
 Volume Serial Number is EC21-77CD

C:\>vol e:
 Volume in drive E is New Volume
 Volume Serial Number is 60B4-4F09

C:\>
```

9. DEL/ERASE

Used to delete one or more files.

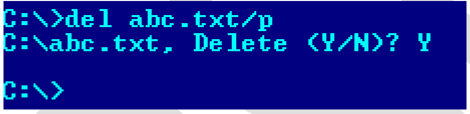
Syntax is **DEL [/P] [/F] [/S] [/Q] [/A[[:]attributes]] names**

Here,

tr>

names — Specifies a list of one or more files or directories. Wildcards * and ? may be used to delete multiple files. * indicates group of unknown characters whereas using wildcard ? in file-names is for single unknown character. And using this command if a directory is specified, all files within the directory will be deleted.

/P — Prompts for (Y)es/(N)o confirmation before deleting each file.



```
C:\>del abc.txt/p
C:\abc.txt, Delete (Y/N)? Y

C:\>
```

/F — Used to force delete read-only files.

/S — Delete specified files from all subdirectories. If Command Extensions are enabled DEL and ERASE change while using /S switch such that it shows you only the files that are deleted, not the ones it could not find.

/Q — Delete in quite mode and do not ask if ok to delete on global wildcard

/A:attributes — Delete files based on specified attribute. The attributes are: R for Read-only files, S for System files, H for Hidden files, A for files ready for archiving and – Prefix meaning not.

10. COPY

This command is useful in copying one or more files to another file or location. Syntax is **COPY [/D] [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] source [/A | /B] [+ source [/A | /B] [+ …]] [destination [/A | /B]]**

The different switches that can be used with this command as follow along with their use.

source	It specifies the file or files to be copied.
/A	Indicates an ASCII text file.
/B	This switch indicates a binary file.
/D	This allows the destination file to be created with decryption.
destination	This specifies the directory and/or filename for the new file or files.
/V	Helps to verify new files to be written correctly.
/N	Specifying this switch uses short filename, if available, when copying a file with a non-8dot3 file name.
/Y	If destination file already exists, this switch suppresses prompting to confirm you want to overwrite it and does it asap.
/-Y	Contrary to above switch, this causes prompting to confirm you want to overwrite an existing destination file.
/Z	Copies networked files in restartable mode.

For appending multiple files for source use wildcard or file1+file2+file3 format and make sure to specify a single file for destination.

11. MD, CD and RD

1. **MD (or MKDIR)** command stand for make directory and it is used to create a directory. Syntax is **MD [drive:]path**
2. **CD (or CHDIR)** stands for create or change directory and it allows to display the name of or change the current directory or rather we can say come out of a directory. Syntax is **CD [/D] [drive:][path]**

   → Typing *CD drive:* displays the current directory in the specified drive. This CD (or CHDIR) command does not treat spaces as delimiters due to which it allows to CD into a subdirectory name that contains a space without surrounding the name with quotes.

   For example:

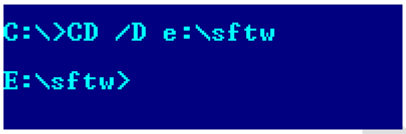
   CHDIR program filesmozilla firefox

   is the same as:

   CHDIR “program filesmozilla firefox”

→ If you type *CD* without any parameters it displays current drive and directory. *CD*.. specifies that you want to change to the higher directory in the current path. Whereas, using *CD* you can directly change to parent/root directory from any location in the current drive.

→Using /D switch changes current drive in addition to current directory for a drive.



```
C:\>CD /D e:\sftw

E:\sftw>
```

3. **RD (or RMDIR)** command removes or deletes a directory. There are two conditions to remove any directory – (1) Directory to be removed should be empty. and (2) We should be outside the directory we are commanding to delete.

   Syntax is **RD [/S] [/Q] [drive:]path**

   Here, using the switch /S removes a directory tree meaning it removes all directories and files in the specified directory in addition to the directory itself. And using /Q is the quiet mode that doesn't asks for ok approval to remove a directory tree.

PROMPT

This changes the cmd.exe command prompt. By default the prompt is always set to the name of current drive followed by > sign.

Customize the Prompt

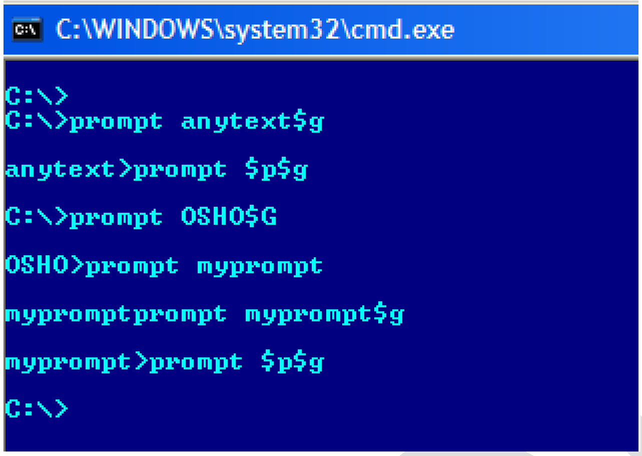
To customize the prompt to display any text of your choice, use the syntax
prompt anytext and this will change the prompt to new command prompt anytext.

Prompt with Options (or Special Codes)

You can use prompt with options. To let the prompt display the current working directory use **prompt pg**

$p in the above signifies the current drive and path.

$g signifies the greater than sign >



```
C:\>
C:\>prompt anytext$g

anytext>prompt $p$g

C:\>prompt OSHO$G

OSHO>prompt myprompt

mypromptprompt myprompt$g

myprompt>prompt $p$g

C:\>
```

The other options used with prompt command are as follows.

Any combination of these can be used with prompt command.

$A & (Ampersand)

$B | (pipe character)

$C ((Left parenthesis)

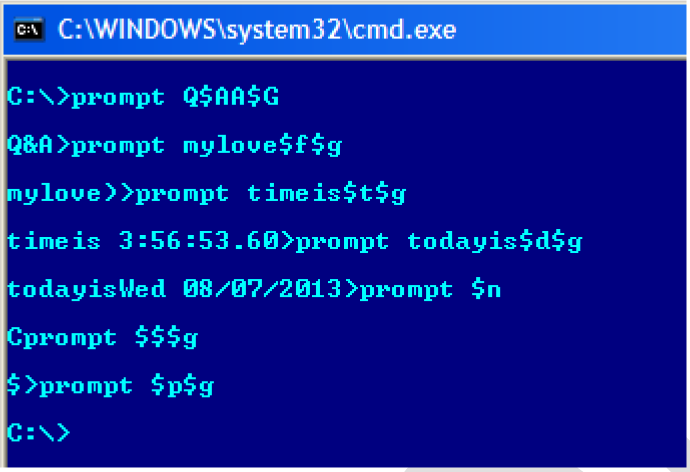
$D Current date

$E Escape code (ASCII code 27)

$F) (Right parenthesis)

$H Backspace (erases previous character)

$L < (less-than sign) $N Current drive $Q = (equal sign) $T Current time $V Windows XP version number $_ Carriage return and linefeed $$ $ (dollar sign)



```
C:\>prompt Q$AA$G
Q&A>prompt mylove$f$g
mylove)>prompt timeis$t$g
timeis 3:56:53.60>prompt todayis$d$g
todayisWed 08/07/2013>prompt $n
Cprompt $$$g
$>prompt $p$g
C:\>
```

Most Commonly Used External DOS Commands

1. **EDIT**

This command is used to modify or change the data of a file.

Syntax is **EDIT [/B] [/H] [/R] [/S] [filename(s)]**

Using switch /B you can force the edit in monochrome mode. /H displays the maximum number of lines possible for your system hardware. Whereas using /R and /S one can load files in read-only mode and force the use of short filenames respectively. [filename(s)] is used to specify file(s) to go edit. You can use wildcards (* and ?) to specify multiple files.

2. **XCOPY**

This command is used to copy files and directory trees from one disk to another disk.

Syntax is **XCOPY source [destination] [/A | /M] [/D[:date]] [/P] [/S [/E]] [/V] [/W] [/C] [/I] [/Q] [/F] [/L] [/G] [/H] [/R] [/T] [/U] [/K] [/N] [/O] [/X] [/Y] [/-Y] [/Z] [/EXCLUDE:file1[+file2][+file3]…]**

3. **LABEL**

It is used to create, change, or delete the volume label of a disk.

Syntax is **LABEL [drive:] [label]**

LABEL [/MP] [volume] [label]

Here, [drive:] is for secifying the drive letter of a drive to be labelled and [label] specifies the label of the volume disk. [/MP] is used to specify that the volume should be created as a mount point and [volume] is used to specify volume name, usually mentioned after drive letter followed by colon and then giving volume name required.

4. **DISKCOPY**

This command copies the contents of one floppy from the source drive to a formatted or un-formatted floppy disk in the destination drive. This command copies the data from particular position on the source disk to exactly the same position on the destination disk. Syntax **Diskcopy A: B:**
copies contents of A: to B: drive. This command can be used with /V switch which verifies that the disk is copied correctly.

5. **CHKDSK**

This command is used to check a disk and display a status report with properties of disk like serial number, volume label, memory and other properties along with errors on the disk if any.
Syntax is **CHKDSK [volume path] [/F] [/V] [/R] [/X] [/I] [/C] [/L[:size]]**
[volume path] is where you specify the drive letter followed by a colon and volume name to be checked. using /F switch allows you to fix errors on the disk. /V display full path and/or cleanup message if any. /R is used in tandem with /F and used to locate bad sectors and recover readable information. If you wanted to perform a less vigorous check of index entries on the disk then the right option is to use /I or /C rather then /R as they skip checking of cycles on the volume and helps in reducing the amount of time required to run chkdsk. Using /X forces the volume to dismount first before checking is performed. /L:size is all about specifying the log file size in kilobytes.

6. **TREE**

This command is very useful to view the list of directories and subdirectories present on the disk in graphical form. If you wanted to include files also with directories and subdirectories, then you'll have to give the command line as tree/f which presents the tree view of all the content on your disk. Here is the syntax for this command with allowed switches:

TREE [drive:path] [/F] [/A]

In case you wanted use ASCII instead of extended characters, then go ahead include /A in the command line.

7. **DELTREE**

This command is used to remove a directory along with its contents.

Syntax is **deltree [drive:path]**

here, [drive:path] specifies the directory name to be deleted. All the subdirectories and files in this directory will be deleted without prompt and there’s not getting back. So, keep caution while using this command.

8. **DOSKEY**

This command is generally used to edits command lines and recalls commands.

Syntax is **DOSKEY [/REINSTALL] [/HISTORY] [text]**

Here, /REINSTALL installs new copy of doskey, /HISTORY is used to display all previously given commands stored in memory. And [text] specifies the commands you want to record.

9. **FIND**

This command searches for a specific text string in a file or files. Syntax is **FIND [/V] [/C] [/N] [/I] [/OFF] “string” [[drive:][path]filename[…]]**

The basic essential elements in the command line for find are – the string enclosed in ” ” and [[drive:][path]filename(s)]. String specifies the text string to find in the file and [[drive:][path]filename(s)] specifies the file or files where the text string search is to be done. If a path is not specified, FIND searches the text typed at the prompt or piped from another command. When you append /OFF in the command line, it searches and finds even those files with offline attribute set. Apart from searching the text string, this command is useful in:

1. Displaying all lines not containing the specified string @ /V
2. Displaying only the number count of lines containing the text string @ /C
3. Displaying line numbers with the displayed lines @ /N

SORT

This command is used to arrange the data of a file in alphabetical order (A-Z, 0-9) or reverse alphabetical order.

Syntax is **SORT [/R] [[drive1:][path1]filename1] [/T [drive2:][path2]] [/O [drive3:][path3]filename3]**

/R in command line reverses the sort order; that is, the data of the specified file sorts sorts Z to A, then 9 to 0. [drive1:][path1]filename1 specifies the file to be sorted. /T [drive2:][path2] is used in cases of data overflow in main memory and it specifies the path of the directory to hold the sort's working storage. And /O [drive3:][path3]filename3 specifies the file where the sorted input is to be stored.

FORMAT

This command creates a new root directory and a File Allocation Table (FAT) for the disk. In order for MS-DOS to be able to use a new disk you must use this command to format the disk.

FORMAT with /S switch

When the disk is formatted with **/s** option, the disk can be used as a booting disk.**C:>DOS>Format A: /s**

The above command copies the OS files MSDOS.SYS, IO.SYS and COMMAND.COM which are required for booting the machine from your system startup drive to the newly formatted disk. The disk can then be used for booting.

FORMAT with /U switch

Here's the command **C:DOS>Format A: /U**

This command specifies an Unconditional Format which destroys all existing data and prevents you from later unformatting the disk.

FORMAT with /Q switch

This can be used only with the previously formatted disk. This deletes FAT, Root directory and data of disk but doesn't scan for the bad errors. This is generally used for Quick formatting.

Warning As Format command deletes all existing data, use this command with extreme caution. Any disk formatted (except with /U switch) may be later unformatted using the UNFORMAT command.

BACKUP

The Backup command backs up one or more files from one disk to another. You can backup files onto either a hard disk or on a floppy disk. Syntax is

BACKUP Source Destination

Here source specifies the location of files to be backed up and destination drive specifies the drive on which you want to store the backup files. The backed-up files are stored in backup.nnn and control.nnn files where nnn represents the backup disk number.

Backup with Switches

0. The /S switch can be used to backup the contents of all files in the source including the contents of sub-directories.
1. The /N switch can be used to backup only those files that have changed since the last backup.
2. Backup command with /D:mm-dd-yyyy switch will backup files that have changed since the data specified.

RESTORE

The RESTORE command restores files that were backed up by using BACKUP command.

Syntax: **RESTORE drive1 drive2:path**

Here drive1 specifies the drive on which backup files are stored.

drive2:path specifies the path to which those backup files will be restored.

► Using backup command with /S switch is used to restore all backup files to their original directories and sub-directories.

Experiment No 2

Aim: Introduction to windows with features.

To learn and practice basic Windows functions, it is recommended to read some Windows Help; use for instance the Index tab in the Windows Help to find topics on Introduction and Welcome.

Some common Windows features:

Moving a window:

A window can be moved over your desktop by dragging the title bar of a window to a new position: position the mouse pointer on the title bar, press the left mouse button and hold it down, and move the mouse pointer to a new position, then release the left mouse button.

Changing the size and shape of a window:

A window can be sized by dragging its borders. Select the window you want to resize, point to a border or a corner of a window, the cursor changes into a two-headed arrow, drag (press the left mouse button and hold it down) the border or corner until the window has the size you want, then release the mouse button.

A window cannot be resized when a map or table is already entirely displayed in a window.

Resizing a window is possible when the border of a window consists of double lines.

Maximize:

A window can be enlarged to the full size of your screen:

- by clicking the maximize button (the delta-up button at the upper-right corner of a window), or
- by clicking the Control-menu box in the upper-left corner of a window to open the Control menu, select the Maximize command, or
- by double-clicking the title bar of a window.

To restore a window, double-click the title bar again. When two or more windows are open and you maximize one of them, all the others are covered up by the maximized one.

Restoring a window to its previous size: (after maximize)

A window is restored to its previous size by clicking the restore button, i.e. the double-delta button at the upper-right corner of a maximized window. You can also restore a window to its previous size by double-clicking the title bar. Another way is to click the Control-menu box and select Restore from the menu.

Minimize:

A window can be reduced to an icon on the desktop if you want the window or application available for later use. To reduce a window to an icon:

- click the minimize button (the delta-down button at the upper-right corner of a window), or
- click the Control-menu box in the upper-left corner of the window to open the Control menu, select the Minimize command.

After a window is reduced to an icon, you can select and move the icon in the same way you select and move a window. See also: icons of minimized windows.

Restoring an icon: (after minimize)

An icon can be restored to a window by double-clicking it. You can also click the minimized icon and select Restore from the Control menu.

Using scroll bars:

When a map or table cannot be entirely displayed in a window, scroll bars appear at the right and/or bottom of the window. You may have both a horizontal and vertical scroll bar or only one of them, depending on the size of your map/table in the window. You can use the scroll bars to view information that does not fit in the window.

The buttons with the arrows are called scroll arrows, the area between the scroll arrows is called the scroll bar, and the button in the scroll bar is called the scroll box.

The best ways to use the scroll bars in windows are:

- Click above or below the scroll box on the vertical scroll bar: the contents of the window will scroll vertically.
- Click to the left or right of the scroll box on the horizontal scroll bar: the contents of the window will scroll horizontally.
- Drag the scroll box in a scroll bar to the position you want. The section of the map/table that moves into view depends on where you position the scroll box. For example, if you position the scroll box half way down the vertical scroll bar, the contents halfway the map/table appear.

Using scroll bars can be convenient when you have zoomed in on a map.

Closing a window:

A window can be closed:

- from the File menu, choose Exit,
- by double-clicking the Control-menu box,

- from the Control menu, choose Close, or
- by pressing Alt+F4 on the keyboard.

Experiment No 3

AIM: Installation of Operating System (Windows)

PLEASE READ THIS BEFORE YOU START

The process detailed in this document is only for those who want to perform a clean install of Windows 7. This means that any previous version of Windows (XP or Vista) and all programs and files will be completely removed from the system. This is OK for a public computer system, but if you're planning on following this process on a staff or circulation computer, you will want to backup any important information first.

Windows 7 Installation Overview

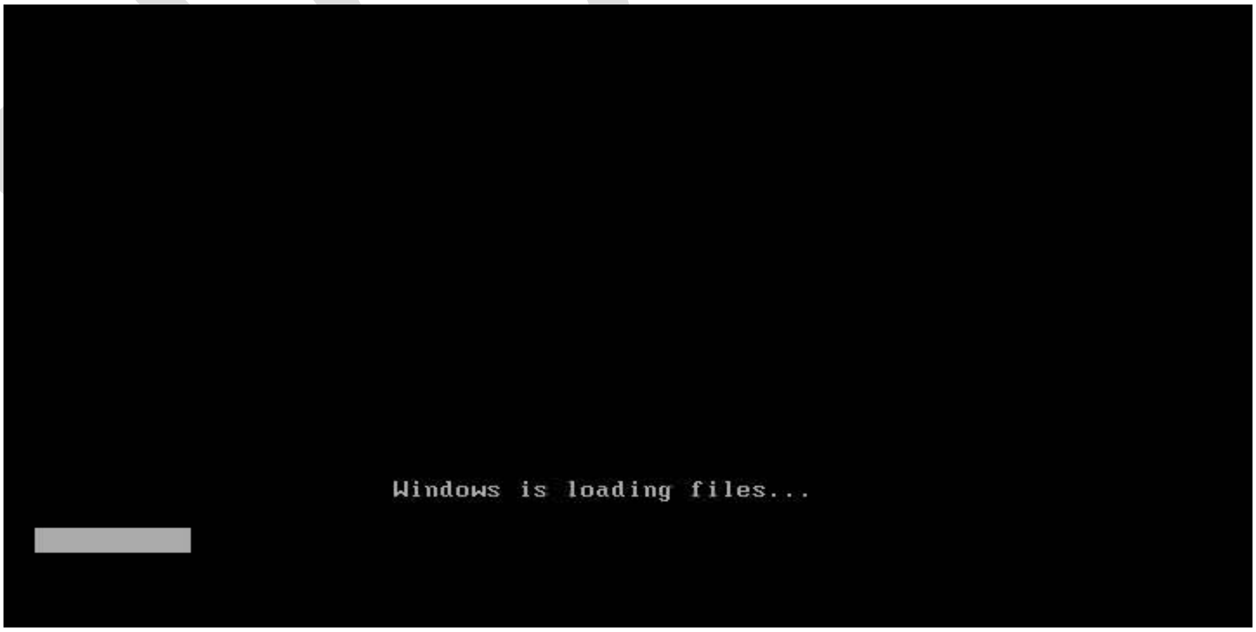
You should have to get your System Specifications and then search for it on Google. If Windows 7 Supports your system then begin this tutorial step by step. In this tutorial you will learn How To Install Windows 7.

Things You'll Need Before Windows 7 Installation

If you already have boot-able CD/DVD then you can skip this part. Otherwise if you need to have Windows 7 ISO download then click here.Also if you want to install windows 7 with USB then make flash drive boot-able by following this method.

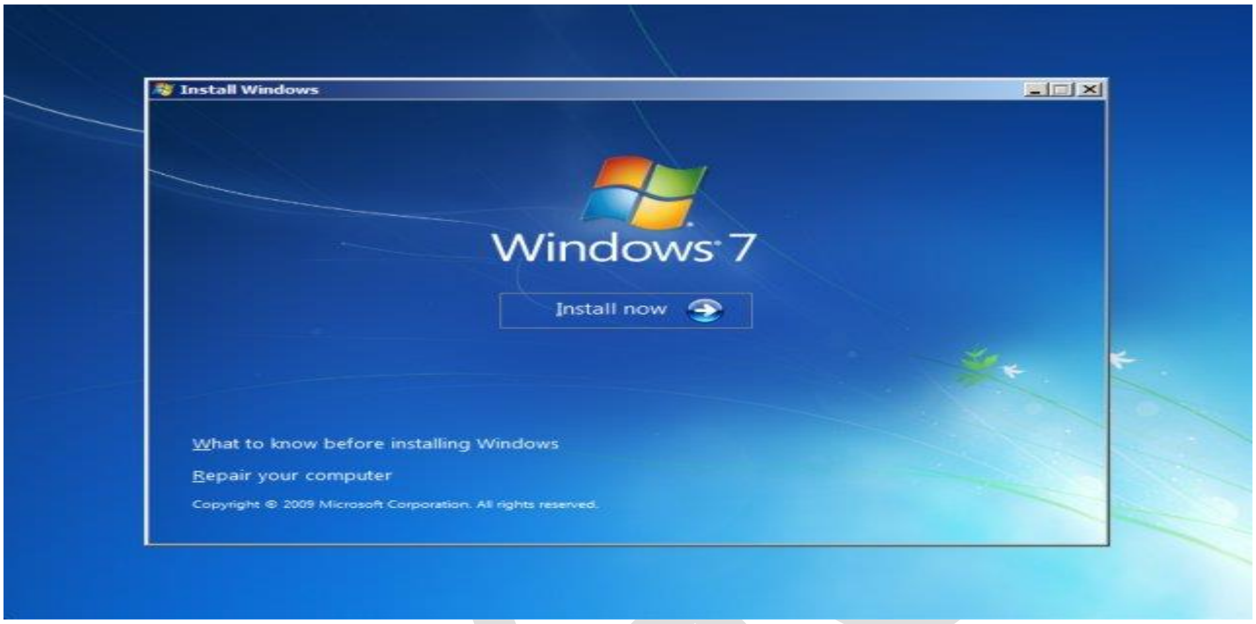
How To Install Windows 7 – Steps

Step#1 Turn ON your PC and Press 'F2′ Continuously. There will come up and option to boot through CD/DVD. Select that option. Windows will start loading its files.

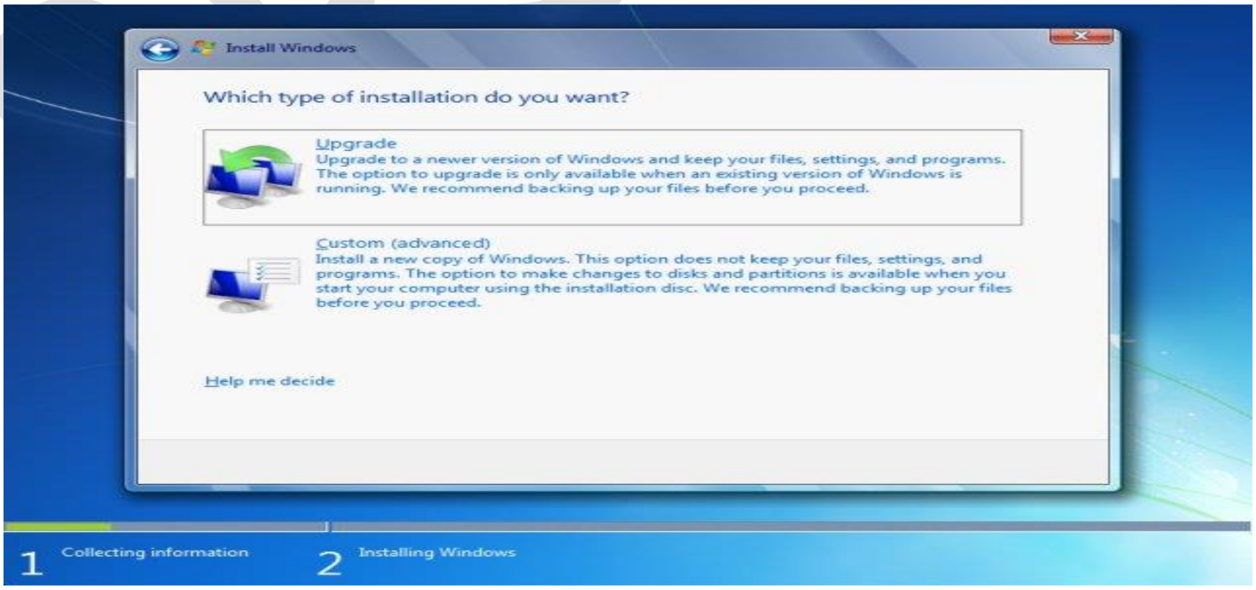


```
Windows is loading files...
```

Step#2 Now you will get the Windows Setup Window. This is the part to select Language for your windows. Select ‘English’ and click Next. Also there will be a ‘INSTALL NOW’ button. Click on it and proceed to next step.


Install Windows
Windows 7
Install now
What to know before installing Windows
Repair your computer
Copyright © 2009 Microsoft Corporation. All rights reserved.

Step#3 There will be a license agreement. Check on ‘I ACCEPT’ and proceed to NEXT. After that there will be an option to install windows. ‘UPGRADE’ and ‘CUSTOM’. Right now we are installing a clean version so Click on CUSTOM.

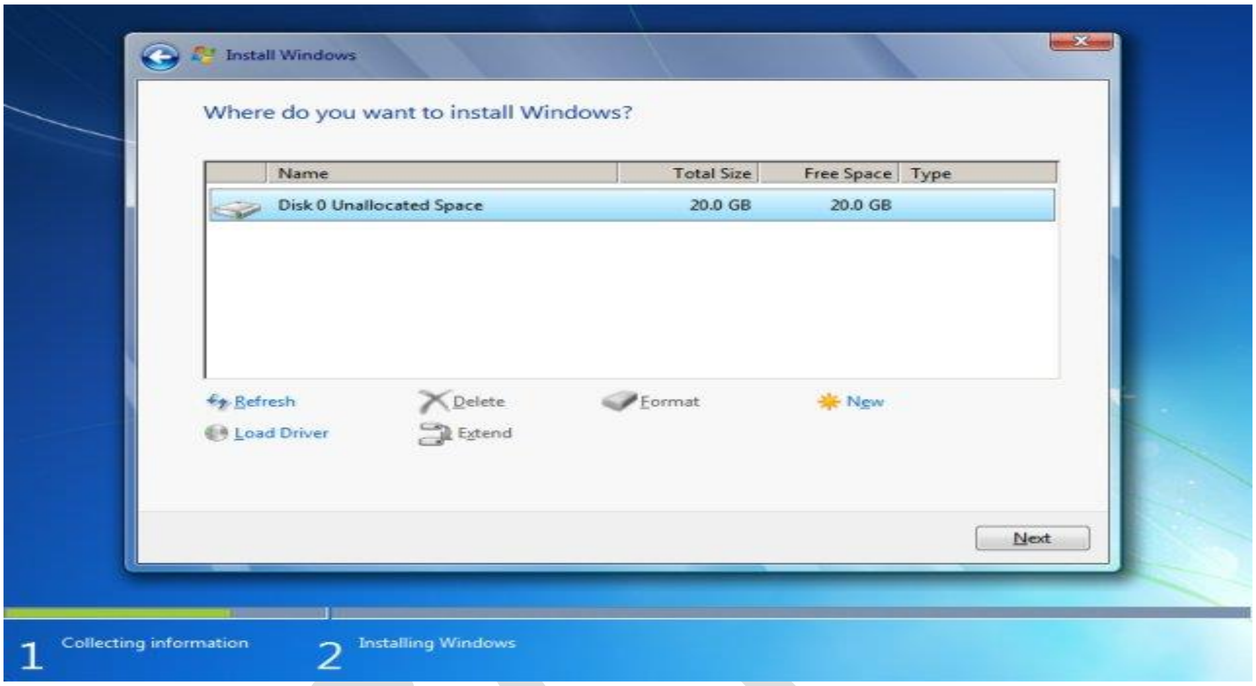


Which type of installation do you want?

Upgrade
Upgrade to a newer version of Windows and keep your files, settings, and programs. The option to upgrade is only available when an existing version of Windows is running. We recommend backing up your files before you proceed.

Custom (advanced)
Install a new copy of Windows. This option does not keep your files, settings, and programs. The option to make changes to disks and partitions is available when you start your computer using the installation disc. We recommend backing up your files before you proceed.

Step#4 In this step you will do partitioning of your drive. Be careful, this is the most important part of the Installation. In this you will allocate spaces to your drive. If you want to create a new drive, simply click on a drive and then click ‘NEW’. A new drive will be created.

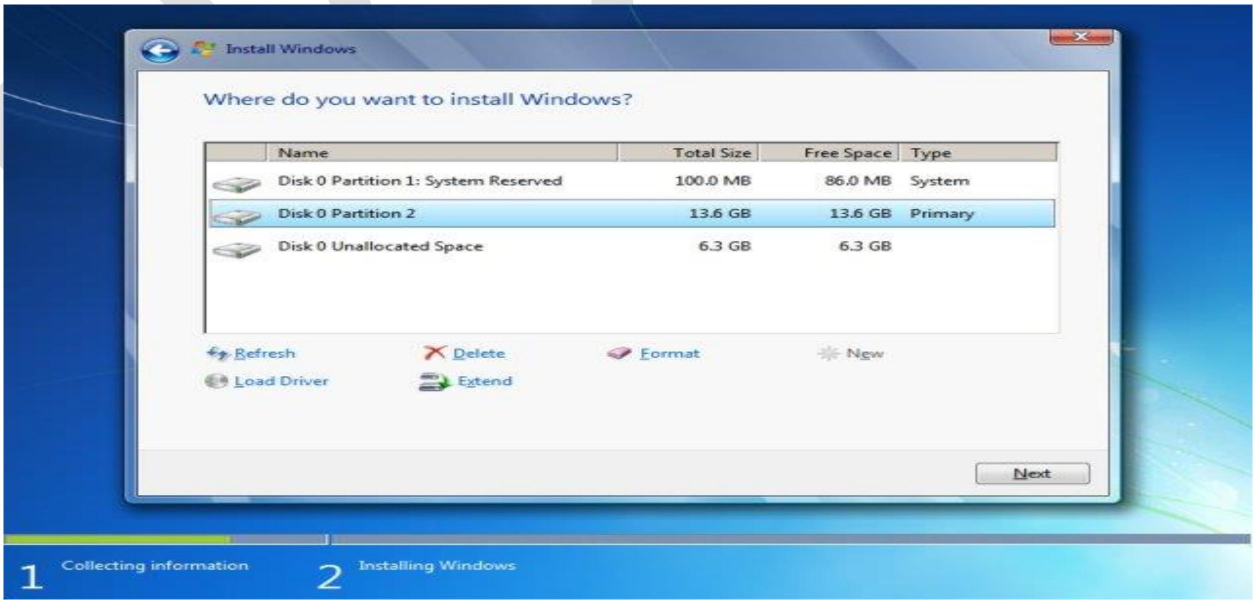


Install Windows

Where do you want to install Windows?

Name	Total Size	Free Space	Type
Disk 0 Unallocated Space	20.0 GB	20.0 GB	

Step#5 When you have created the drives, Simply Select the drive in which you want to install windows. Click ‘Format’, this will erase all the previous data on you that drive. Click on ‘NEXT’ to proceed.

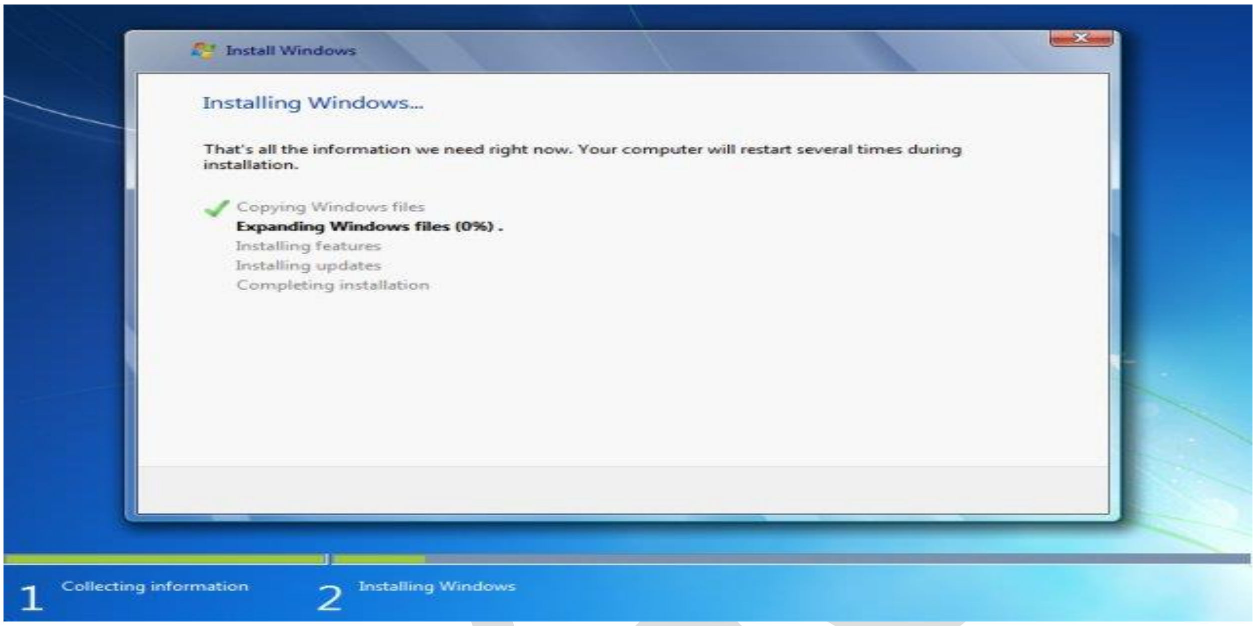


Install Windows

Where do you want to install Windows?

Name	Total Size	Free Space	Type
Disk 0 Partition 1: System Reserved	100.0 MB	86.0 MB	System
Disk 0 Partition 2	13.6 GB	13.6 GB	Primary
Disk 0 Unallocated Space	6.3 GB	6.3 GB	

Step#6 Now you windows will start installing its files. Grab a cup of coffee and wait for a few minutes while it install. During this process don’t plug in or off your device. It might cause interruption and you might loose your data and have to begin the process all over again.

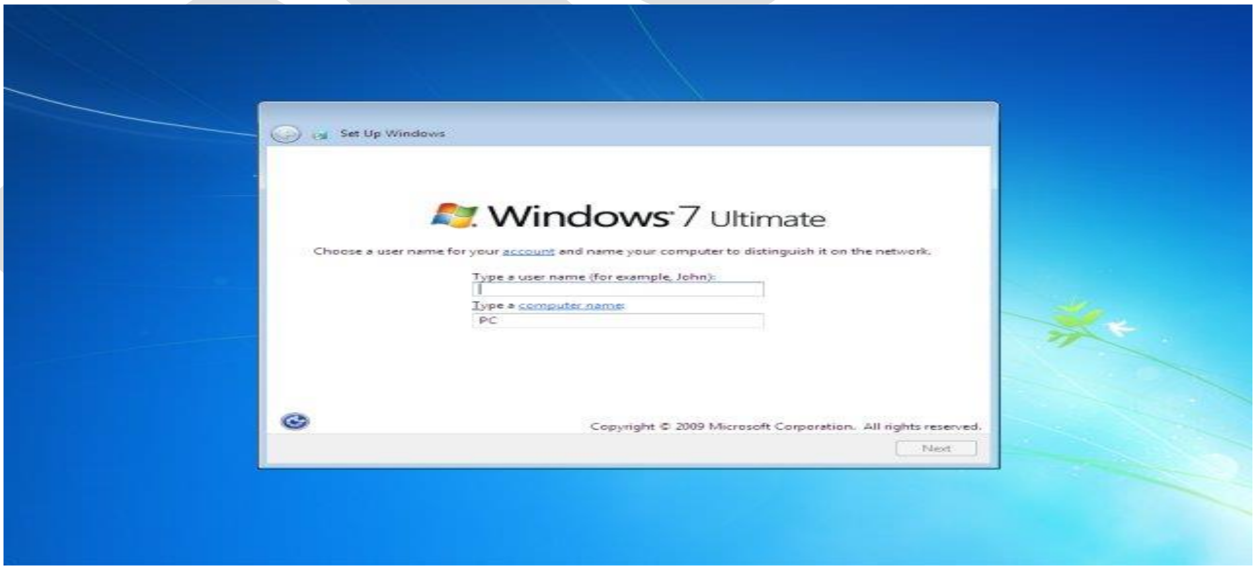

Install Windows

Installing Windows...

That's all the information we need right now. Your computer will restart several times during installation.

✓ Copying Windows files
Expanding Windows files (0%) .
Installing features
Installing updates
Completing installation

Step#7 Now when you files are installed. Your PC will be rebooted and now you will see is a ‘User Settings’ Screen. Simply add your Name and password and proceed to ‘NEXT’


Set Up Windows

Windows 7 Ultimate

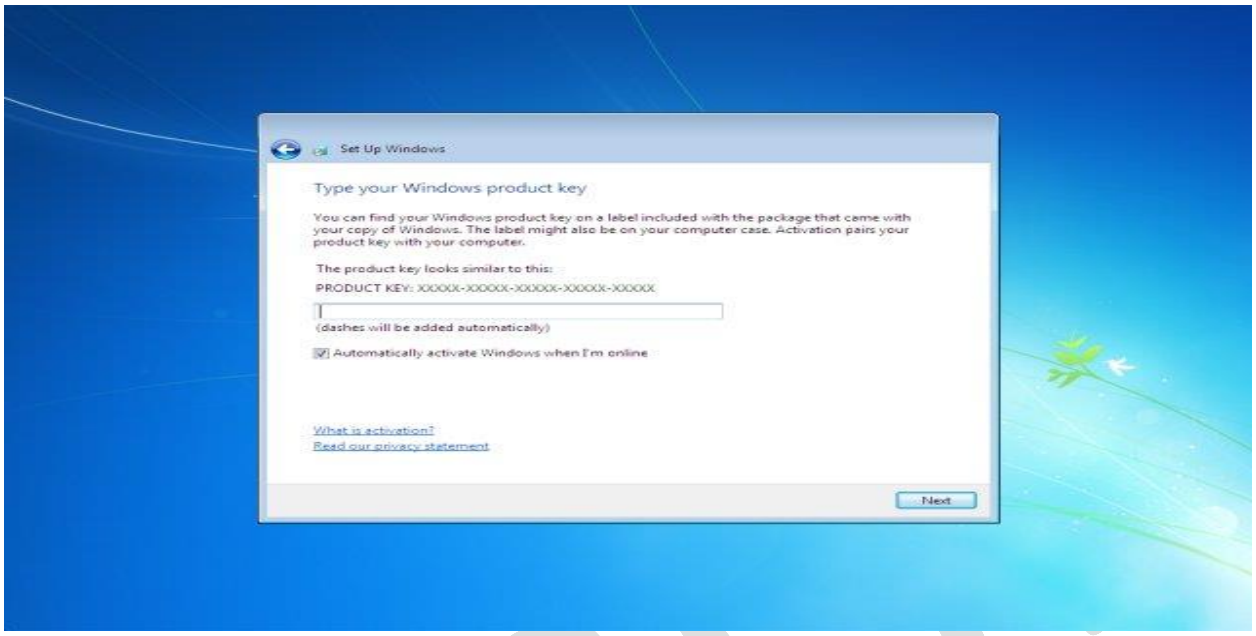
Choose a user name for your account and name your computer to distinguish it on the network.

Type a user name (for example, John):

Type a computer name:
PC

Copyright © 2009 Microsoft Corporation. All rights reserved.

Step#8 In this step you have to activate your windows. Simply look at the back of your Windows CD/DVD cover there will be a PRODUCT KEY. Add this key into your PC and Click ‘NEXT’.



Type your Windows product key

You can find your Windows product key on a label included with the package that came with your copy of Windows. The label might also be on your computer case. Activation pairs your product key with your computer.

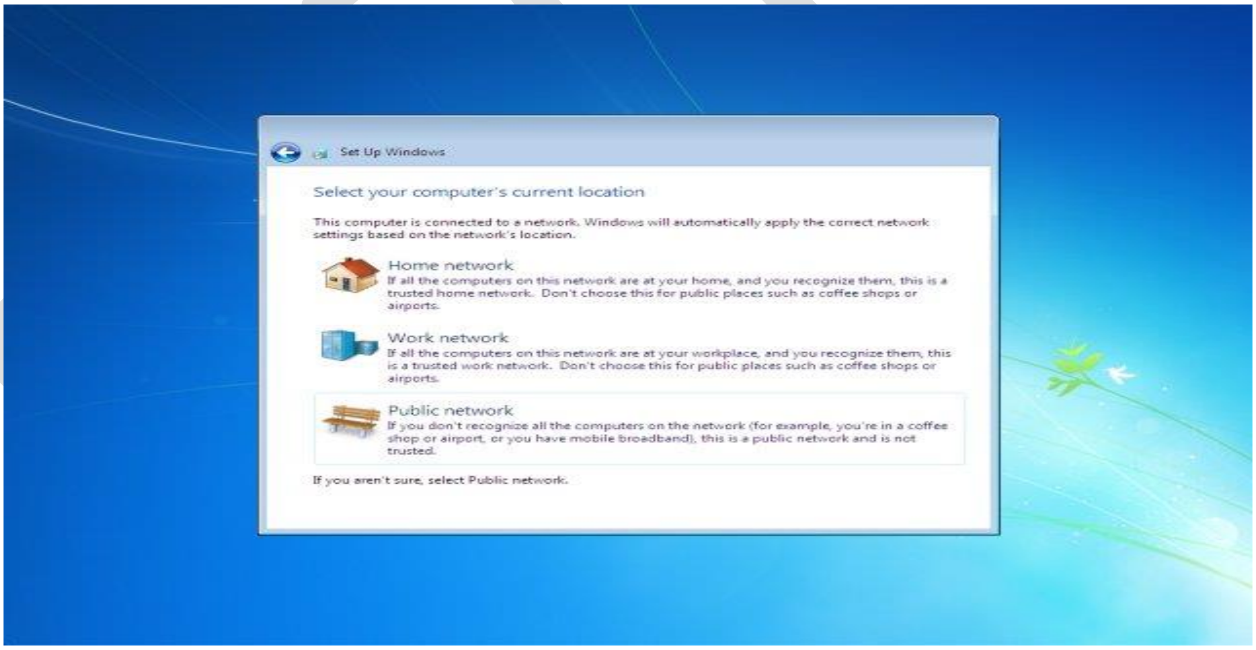
The product key looks similar to this:

PRODUCT KEY: XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

(dashes will be added automatically)

☑ Automatically activate Windows when I'm online

Step#9 Now you have Installed you windows. Give the desired information the Windows Step guide will ask, like Time Zone, Update Timing and Your Computers location. There are three types of location. ‘WORK’, ‘PUBLIC’ and ‘HOME’. Select on anyone of the them according to your location. It only add sharing security according to your location.



Select your computer's current location

This computer is connected to a network. Windows will automatically apply the correct network settings based on the network's location.

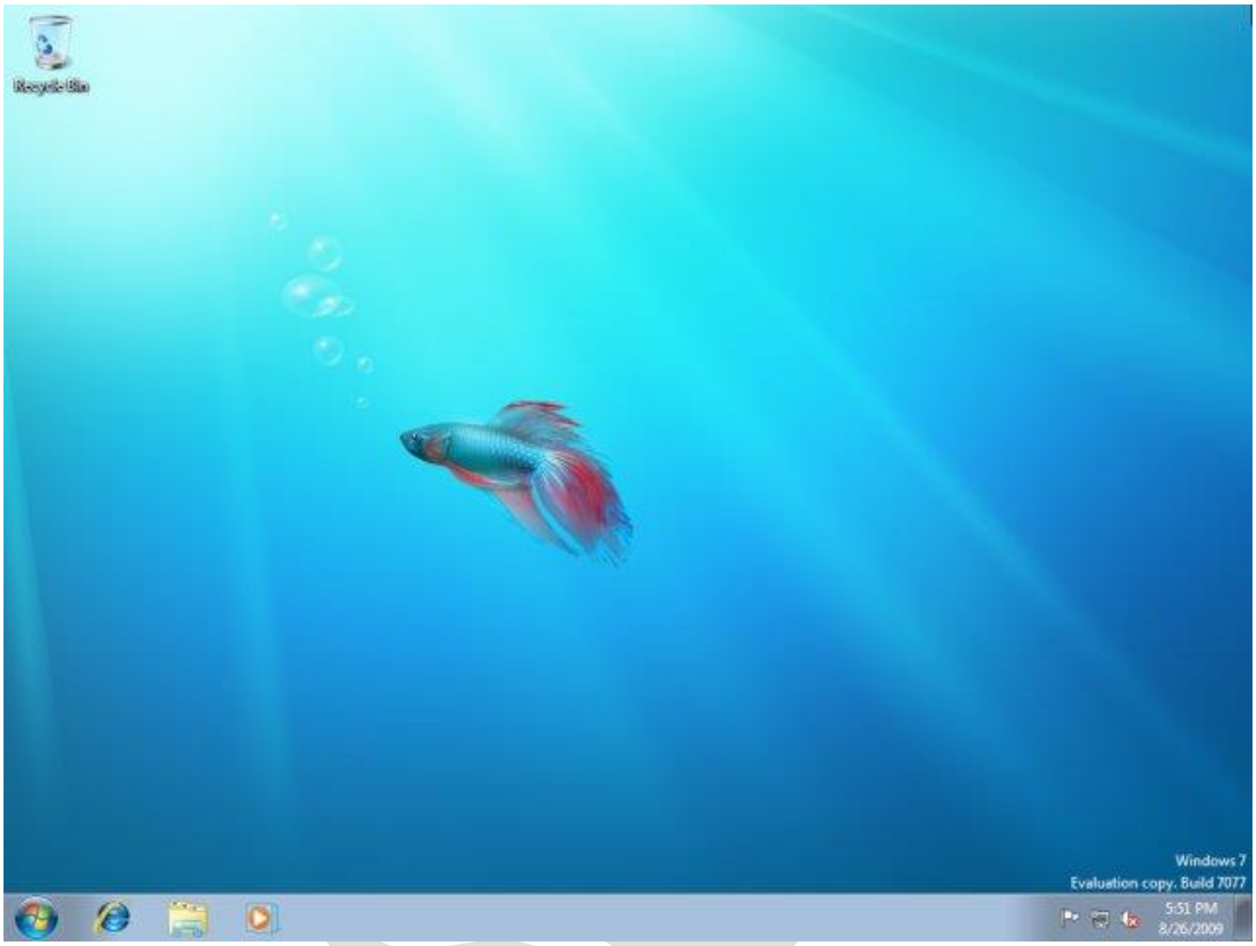
Home network
If all the computers on this network are at your home, and you recognize them, this is a trusted home network. Don't choose this for public places such as coffee shops or airports.

Work network
If all the computers on this network are at your workplace, and you recognize them, this is a trusted work network. Don't choose this for public places such as coffee shops or airports.

Public network
If you don't recognize all the computers on the network (for example, you're in a coffee shop or airport, or you have mobile broadband), this is a public network and is not trusted.

If you aren't sure, select Public network.

Last Step – Congratulations:- You have installed you windows. Now you can see is your desktop. It is simple to use, setup your desktop and enjoy!


Recycle Bin
Windows 7
Evaluation copy. Build 7077
5:51 PM
8/26/2009

Experiment No 4

STUDY OF ADMINISTRATIVE TOOLS

Administrative Tools is a folder in Control Panel that contains tools for system administrators and advanced users. The tools in the folder might vary depending on which version of Windows you are using.

- Open Administrative Tools by clicking the Start button , and then clicking Control Panel. In the search box, type administrative tools, and then click Administrative Tools.

Many of the tools in this folder, such as Computer Management, are Microsoft Management Console (MMC) snap-ins that include their own help topics. To view specific help for an MMC tool, or to search for an MMC snap-in that you don't see in the following list, open the tool, click the Help menu, and then click Help Topics.

Some common administrative tools in this folder include:

- **Component Services.** Configure and administer Component Object Model (COM) components. Component Services is designed for use by developers and administrators.
- **Computer Management.** Manage local or remote computers by using a single, consolidated desktop tool. Using Computer Management, you can perform many tasks, such as monitoring system events, configuring hard disks, and managing system performance.
- **Data Sources (ODBC).** Use Open Database Connectivity (ODBC) to move data from one type of database (a data source) to another. For more information, see what is ODBC?
- **Event Viewer.** View information about significant events, such as a program starting or stopping, or a security error, which are recorded in event logs.
- **iSCSI Initiator.** Configure advanced connections between storage devices on a network. For more information, see What is Internet Small Computer System Interface (iSCSI)?
- **Local Security Policy.** View and edit Group Policy security settings.
- **Performance Monitor.** View advanced system information about the central processing unit (CPU), memory, hard disk, and network performance.

- **Print Management.** Manage printers and print servers on a network and perform other administrative tasks.
- **Services.** Manage the different services that run in the background on your computer.
- **System Configuration.** Identify problems that might be preventing Windows from running correctly.
- **Task Scheduler.** Schedule programs or other tasks to run automatically. For more information, see Schedule a task.
- **Windows Firewall with Advanced Security.** Configure advanced firewall settings on both this computer and remote computers on your network.
- **Windows Memory Diagnostic.** Check your computer's memory to see if it's functioning properly.

Experiment No 5

STUDY OF LINUX

INTRODUCTION TO LINUX

Linux is a generic term referring to Unix-like computer operating systems based on the Linux kernel. Their development is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed by anyone. The name "Linux" comes from the Linux kernel, originally written in 1991 by Linus Torvalds. The rest of the system usually comprises components such as the Apache HTTP Server, the X Window System, the K Desktop Environment, and utilities and libraries from the GNU operating system (announced in 1983 by Richard Stallman). Many quantitative studies of free / open source software focus on topics including market share and reliability, with numerous studies specifically examining Linux. The Linux market is growing rapidly, and the revenue of servers, desktops, and packaged software running Linux was expected to exceed $35.7 billion by 2008.

LINUX FILE SYSTEM

A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the file system. The difference between a disk or partition and the file system it contains is important. A few programs (including, reasonably enough, programs that create file systems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a file system, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong type). Before a partition or disk can be used as a file system, it needs to be initialized, and the bookkeeping data structures need to be written to the disk. This process is called making a file system. Most UNIX file system types have a similar general structure, although the exact details vary quite a bit. The central concepts are superblock, inode, data block, directory block, and indirection block. The superblock contains information about the file system as a whole, such as its size (the exact information here depends on the file system). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first. Like UNIX, Linux chooses to have a single hierarchical directory structure. Everything starts from the root directory, represented by /, and then expands into sub-directories instead of having so-called 'drives'. In the Windows environment, one may put one's files almost anywhere: on C drive, D drive, E drive etc. Such a file system is called a hierarchical structure and is managed by the programs themselves (program directories), not by the operating system. On the other hand, Linux sorts directories descending from the root directory / according to their importance to the boot process. Linux, like Unix also chooses to be case sensitive. What this means is that the case, whether in capitals or not, of the characters becomes very important. This feature accounts for a fairly large proportion of problems for new users especially during file transfer operations whether it may be via removable disk media such as floppy disk or over the wire by way of FTP.

Unix Commands

AIM : To study and excute the commands in unix.

COMMANDS :

1.Date Command :

This command is used to display the current data and time.

Syntax : $date $date +%ch

Options : -

a = Abbrevated weekday.
A = Full weekday.
b = Abbrevated month.
B = Full month.
c = Current day and time.
C = Display the century as a decimal number.
d = Day of the month.
D = Day in „mm/dd/yy" format
h = Abbrevated month day.
H = Display the hour.
L = Day of the year.
m = Month of the year.
M = Minute.
P = Display AM or PM
S = Seconds
T = HH:MM:SS format
u = Week of the year.
y = Display the year in 2 digit.
Y = Display the full year.
Z = Time zone .

To change the format :

Syntax : $date „+%H-%M-%S"

2.Calender Command :

This command is used to display the calendar of the year or the particular month of calendar year.

Syntax :

a.$cal b.$cal

Here the first syntax gives the entire calendar for given year & the second Syntax gives the calendar of reserved month of that year.

3.Echo Command :

This command is used to print the arguments on the screen .

Syntax :

```
$echo
```

Multi line echo command :

To have the output in the same line , the following commands can be used.

Syntax :

```
$echo text
```

To have the output in different line, the following command can be used.

Syntax :

```
$echo “text >line2 >line3”
```

4.Banner Command :

It is used to display the arguments in „#‟ symbol .

Syntax :

```
$banner
```

5.’who’ Command :

It is used to display who are the users connected to our computer currently.

Syntax :

```
$who – option‟s
```

Options : -

H–Display the output with headers.

b–Display the last booting date or time or when the system was lastely rebooted.

6.’who am i’ Command : Display the details of the current working directory.

Syntax :

```
$who am i
```

7.’tty’ Command :

It will display the terminal name.

Syntax :

```
$tty
```

8.’Binary’ Calculator Command : It will change the „$‟ mode and in the new mode, arithematic operations such as +,- ,*,/,%,n,sqrt(),length(),=, etc can be performed . This command is used to go to the binary calculus mode.

Syntax :

$bc operations ^d $ 1 base –inputbase 0 base – outputbase are used for base conversions.

Base : Decimal = 1 Binary = 2 Octal = 8 Hexa = 16

9.'CLEAR' Command :

It is used to clear the screen.

Syntax :

$clear

10.'MAN' Command :

It help us to know about the particular command and its options & working. It is like „help" command in windows .

Syntax :

$man <command name>

11.MANIPULATION Command :

It is used to manipulate the screen.

Syntax :

$tput <argument>

Arguments :

1.Clear – to clear the screen.
2.Longname – Display the complete name of the terminal.
3.SMSO – background become white and foreground become black color.
4.rmso – background become black and foreground becomes white color.
5.Cop R C – Move to the cursor position to the specified location.
6.Cols – Display the number of columns in our terminals.

12.LIST Command :

It is used to list all the contents in the current working directory.

Syntax :

$ ls – options<arguments>

If the command does not contain any argument means it is working in the Current directory.

Options :

a– used to list all the files including the hidden files.
c– list all the files columnwise.
d- list all the directories.
m- list the files separated by commas.
p- list files include „/" to all the directories.
r- list the files in reverse alphabetical order.
f- list the files based on the list modification date.
x-list in column wise sorted order.

DIRECTORY RELATED COMMANDS :

1.Present Working Directory Command :

To print the complete path of the current working directory.

Syntax :

```
$pwd
```

2.MKDIR Command :

To create or make a new directory in a current directory .

Syntax :

```
$mkdir<directory name>
```

3.CD Command :

To change or move the directory to the mentioned directory .

Syntax :

```
$cd <directory name>
```

FILE RELATED COMMANDS :

1.CREATE A FILE :

To create a new file in the current directory we use CAT command.

Syntax :

```
$cat ><filename
```

The > symbol is redirectory we use cat command.

2.DISPLAY A FILE :

To display the content of file mentioned we use CAT command without „>‟ operator.

Syntax :

```
$cat <filename.
```

Options –s = to neglect the warning /error message.

3.COPYING CONTENTS :

To copy the content of one file with another. If file doesnot exist, a new file is created and if the file exists with some data then it is overwritten.

Syntax :

```
$ cat <filename source> >> <destination filename>
```

$ cat <source filename> >> <destination filename> it is avoid overwriting.

Options : -

-n content of file with numbers included with blank lines.

Syntax :

```
$cat –n <filename>
```

4.SORTING A FILE :

To sort the contents in alphabetical order in reverse order.

Syntax :

```
$sort <filename >
```

Option : $ sort –r <filename>

5.COPYING CONTENTS FROM ONE FILE TO ANOTHER :

To copy the contents from source to destination file . so that both contents are same.

Syntax :

```
$cp <source filename> <destination filename>
$cp <source filename path > <destination filename path>
```

6.MOVE Command :

To completely move the contents from source file to destination file and to remove the source file.

Syntax :

```
          $ mv <source filename> <destination filename>
```

7.REMOVE Command :

To permanently remove the file we use this command .

Syntax :

```
$rm <filename>
```

8.WORD Command :

To list the content count of no of lines , words, characters .

Syntax :

```
$wc<filename>
```

Options :

-c – to display no of characters.

-l – to display only the lines.

-w – to display the no of words.

9.LINE PRINTER :

To print the line through the printer, we use lp command.

Syntax :

```
          $lp <filename>
```

10.PAGE Command :

This command is used to display the contents of the file page wise & next page can be viewed by pressing the enter key.

Syntax :

```
$pg <filename>
```

11. FILTERS AND PIPES

HEAD : It is used to display the top ten lines of file.

Syntax: $head<filename>

TAIL : This command is used to display the last ten lines of file.

Syntax: $tail<filename>

PAGE : This command shows the page by page a screenfull of information is displayed after which the page command displays a prompt and passes for the user to strike the enter key to continue scrolling.

Syntax: $ls –a\p

MORE : It also displays the file page by page .To continue scrolling with more command , press the space bar key.

Syntax: $more<filename>

GREP : This command is used to search and print the specified patterns from the file. Sy**ntax:** $grep [option] pattern <filename>

SORT : This command is used to sort the datas in some order.
Syntax: $sort<filename>

PIPE : It is a mechanism by which the output of one command can be channeled into the input of another command.
Syntax: $who | wc-l

TR : The tr filter is used to translate one set of characters from the standard inputs to another.
Syntax: $tr "[a-z]" "[A-Z]"

COMMUNICATION THROUGH UNIX COMMANDS

1.Command :MESG

Description: The message command is used to give permission to other users to send message to your terminal.

Syntax: $mesg y

2.Command: WRITE

Description: This command is used to communicate with other users, who are logged in at the same time.

Syntax: $write

3.Command: WALL

Description: This command sends message to all users those who are logged in using the unix server.

Syntax: $wall

4.Command: MAIL

Description: It refers to textual information, that can be transferred from one user to another

Syntax: $mail

5.Command: REPLY

Description: It is used to send reply to specified user.

Syntax: $reply

Experiment No 6

AIM : To study the various commands operated in vi editor in UNIX.

DESCRIPTION :

The Vi editor is a visual editor used to create and edit text, files, documents and programs. It displays the content of files on the screen and allows a user to add, delete or change part of text. There are three modes available in the Vi editor , they are

1.Command mode

2.Input (or) insert mode.

Starting Vi :

The Vi editor is invoked by giving the following commands in UNIX prompt.

Syntax : $vi <filename> (or)

$vi

This command would open a display screen with 25 lines and with tilt (~) symbol at the start of each line. The first syntax would save the file in the filename mentioned and for the next the filename must be mentioned at the end.

Options :

1.vi +n <filename> - this would point at the nth line (cursor pos).

2.vi –n <filename> - This command is to make the file to read only to change from one mode to another press escape key.

INSERTING AND REPLACING COMMANDS :

To move editor from command node to edit mode, you have to press the <ESC> key. For inserting and replacing the following commands are used.

1.ESC a Command :

This command is used to move the edit mode and start to append after the current character.

Syntax : <ESC> a

2.ESC A COMMAND :

This command is also used to append the file , but this command append at the end of current line.

Syntax : <ESC> A

3.ESC i Command :

This command is used to insert the text before the current cursor position.

Syntax : <ESC> i

4.ESC I Command :

This command is used to insert at the beginning of the current line.

Syntax : <ESC> I

5.ESC o Command :

This command is insert a blank line below the current line & allow insertion of contents.

Syntax : <ESC> o

6.ESC O Command :

This command is used to insert a blank line above & allow insertion of contents.

Syntax : <ESC> O

7.ESC r Command :

This command is to replace the particular character with the given characters.

Syntax : <ESC> rx Where x is the new character.

8.ESC R Command :

This command is used to replace the particular text with a given text.

Syntax : <ESC> R text

9.<ESC> s Command :

This command replaces a single character with a group of character .

Syntax : <ESC> s

10.<ESC> S Command :

This command is used to replace a current line with group of characters. **Syntax :** <ESC> S

CURSOR MOVEMENT IN vi :

1.<ESC> h :

This command is used to move to the previous character typed. It is used to move to left of the text . It can also used to move character by character (or) a number of characters.

Syntax : <ESC> h – to move one character to left.

<ESC> nh – tomove „n‟ character to left.

2.<ESC> l :

This command is used to move to the right of the cursor (ie) to the next character. It can also be used to move the cursor for a number of character.

Syntax : <ESC> l – single character to right.

<ESC> nl - „n‟ characters to right.

3.<ESC> j :

This command is used to move down a single line or a number of lines.

Syntax :

<ESC> j – single down movement.

<ESC> nj – „n‟ times down movement.

4.<ESC> k :

This command is used to move up a single line or a number of lines.

Syntax :

<ESC> k – single line above.

<ESC> nk – „n‟ lines above.

5.ENTER (OR) N ENTER :

This command will move the cursor to the starting of next lines or a group of lines mentioned.

Syntax :

<ESC> enter <ESC> n enter.

6.<ESC> + Command :

This command is used to move to the beginning of the next line.

Syntax :

<ESC> + <ESC> n+

7.<ESC> - Command :

This command is used to move to the beginning of the previous line.

Syntax :

<ESC> - <ESC> n-

8.<ESC> 0 :

This command will bring the cursor to the beginning of the same current line.

Syntax :

<ESC> 0

9.<ESC> $:

This command will bring the cursor to the end of the current line.

Syntax :

<ESC> $

10.<ESC> ^ :

This command is used to move to first character of first lines.

Syntax :

<ESC> ^

11.<ESC> b Command :

This command is used to move back to the previous word (or) a number of words.

Syntax :

<ESC> b <ESC>nb

12.<ESC> e Command :

This command is used to move towards and replace the cursor at last character of the word (or) no of words.

Syntax :

<ESC> e <ESC>ne

13.<ESC> w Command :

This command is used to move forward by a single word or a group of words.

Syntax :

<ESC> w <ESC> nw

DELETING THE TEXT FROM Vi :

1.<ESC> x Command :

To delete a character to right of current cursor positions , this command is used.

Syntax :

<ESC> x <ESC> nx

2.<ESC> X Command :

To delete a character to left of current cursor positions , this command is used.

Syntax :

<ESC> X <ESC> nX

3.<ESC> dw Command :

This command is to delete a single word or number of words to right of current cursor position.

Syntax :

<ESC> dw <ESC> ndw

4.db Command :

This command is to delete a single word to the left of the current cursor position.

Syntax :

<ESC> db <ESC> ndb

5.<ESC> dd Command :

This command is used to delete the current line (or) a number of line below the current line.

Syntax :

<ESC> dd <ESC> ndd

6.<ESC> d$ Command :

This command is used to delete the text from current cursor position to last character of current line.

Syntax : <ESC> d$

SAVING AND QUITING FROM vi :-

1.<ESC> w Command :

To save the given text present in the file.

Syntax : <ESC> : w

2.<ESC> q! Command :

To quit the given text without saving.

Syntax : <ESC> :q!

3.<ESC> wq Command :

This command quits the vi editor after saving the text in the mentioned file.

Syntax : <ESC> :wq

4.<ESC> x Command :

This command is same as „wq‟ command it saves and quit.

Syntax : <ESC> :x

5.<ESC> q Command :

This command would quit the window but it would ask for again to save the file.

Syntax : <ESC> : q

Experiment No 7

AIM :

To study about the Unix Shell Programming Commands.

INTRODUCTION :

Shell programming is a group of commands grouped together under single filename. After logging onto the system a prompt for input appears which is generated by a Command String Interpreter program called the shell. The shell interprets the input, takes appropriate action, and finally prompts for more input. The shell can be used either

interactively - enter commands at the command prompt, or as an interpreter to execute a shell script. Shell scripts are dynamically interpreted, NOT compiled.

Common Shells.

C-Shell - csh : The default on teaching systems Good for interactive systems Inferior programmable features

Bourne Shell - bsh or sh - also restricted shell - bsh : Sophisticated pattern matching and file name substitution

Korn Shell : Backwards compatible with Bourne Shell Regular expression substitution emacs editing mode

Thomas C-Shell - tcsh : Based on C-Shell Additional ability to use emacs to edit the command line Word completion & spelling correction Identifying your shell.

01. SHELL KEYWORDS :

echo, read, if fi, else, case, esac, for , while , do , done, until , set, unset, readonly, shift, export, break, continue, exit, return, trap , wait, eval ,exec, ulimit , umask.

02. General things SHELL

The shbang line The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script. The shbang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behavior of the shell.

E*XAMPLE*

```
#!/bin/sh
```

Comments Comments are descriptive material preceded by a # sign. They are in effect until the end of a line and can be started anywhere on the line.

E*XAMPLE*

```
# this text is not # interpreted by the shell
```

Wildcards There are some characters that are evaluated by the shell in a special way. They are called shell metacharacters or "wildcards." These characters are neither numbers nor letters. For example, the *, ?, and [ ] are used for filename expansion. The <, >, 2>, >>, and | symbols are used for standard I/O redirection and pipes. To prevent these characters from being interpreted by the shell they must be quoted.

E*XAMPLE*

Filename expansion:

```
rm *; ls ??; cat file[1-3];
```

Quotes protect metacharacter:

```
echo "How are you?"
```

03. SHELL VARIABLES :

Shell variables change during the execution of the program .The C Shell offers a command "Set" to assign a value to a variable.

For example:

```
% set myname= Fred
% set myname = "Fred Bloggs"
% set age=20
```

A $ sign operator is used to recall the variable values.

For example:

% echo $myname will display Fred Bloggs on the screen

A @ sign can be used to assign the integer constant values.

For example:

```
%@myage=20
%@age1=10
%@age2=20
%@age=$age1+$age2
%echo $age
```

List variables

```
% set programming_languages= (C LISP)
% echo $programming _languages
C LISP
% set files=*.*
% set colors=(red blue green)
% echo $colors[2]
blue
% set colors=($colors yellow)/add to list
```

Local variables Local variables are in scope for the current shell. When a script ends, they are no longer available; i.e., they go out of scope. Local variables are set and assigned values.

E*XAMPLE*

```
variable_name=value name="John Doe" x=5
```

Global variables Global variables are called environment variables. They are set for the currently running shell and any process spawned from that shell. They go out of scope when the script ends.

E*XAMPLE*

```
        VARIABLE_NAME=value export VARIABLE_NAME PATH=/bin:/usr/bin:. export PATH
```

Extracting values from variables To extract the value from variables, a dollar sign is used.

E*XAMPLE*

```
echo $variable_name echo $name echo $PATH
```

Rules : -

1.A variable name is any combination of alphabets, digits and an
underscore („-„);

2.No commas or blanks are allowed within a variable name.

3.The first character of a variable name must either be an alphabet or an
underscore.

4.Variables names should be of any reasonable length.

5.Variables name are case sensitive . That is , Name, NAME, name,
NAme, are all different variables.

04. EXPRESSION Command :

To perform all arithematic operations .

Syntax :

```
Var = „expr$value1‟ + $ value2‟
```

Arithmetic The Bourne shell does not support arithmetic. UNIX/Linux commands must be used to perform calculations.

E*XAMPLE*

```
n=`expr 5 + 5` echo $n
```

Operators The Bourne shell uses the built-in test command operators to test numbers and strings.

E*XAMPLE*

Equality:

= *string* != *string* -eq *number* -ne *number*

Logical:

-a *and* -o *or* ! *not*

Logical:

AND &&

OR ||

Relational:

-gt *greater than* -ge *greater than, equal to*

-lt *less than* -le *less than, equal to*

Arithmetic **:**

+, -, *, /, %

Arguments (positional parameters) Arguments can be passed to a script from the command line. Positional parameters are used to receive their values from within the script.

E*XAMPLE*

At the command line:

$ scriptname arg1 arg2 arg3 ...

In a script:

echo $1 $2 $3 *Positional parameters* echo $* *All the positional paramters* echo $# *The number of positional parameters*

05.READ Statement :

To get the input from the user.

Syntax :

read x y

(no need of commas between variables)

06. ECHO Statement :

Similar to the output statement. To print output to the screen, the echo command is used. Wildcards must be escaped with either a backslash or matching quotes.

Syntax :

Echo “String” (or) echo $ b(for variable).

E*XAMPLE*

echo "What is your name?"

Reading user input The read command takes a line of input from the user and assigns it to a variable(s) on the right-hand side. The read command can accept muliple variable names. Each variable will be assigned a word.

E*XAMPLE*

echo "What is your name?" read name read name1 name2 ...

6. CONDITIONAL STATEMENTS :

The if construct is followed by a command. If an expression is to be tested, it is enclosed in square brackets. The then keyword is placed after the closing parenthesis. An if must end with a fi.

Syntax :

1.if

This is used to check a condition and if it satisfies the condition if then does the next action , if not it goes to the else part.

2.if…else

Syntax :

If cp $ source $ target
Then
Echo File copied successfully
Else
Echo Failed to copy the file.

3.nested if

here sequence of condition are checked and the corresponding performed accordingly.

Syntax :

```
if condition
then
command
if condition
then
command
else
command
fi
fi
```

4.case ….. esac

This construct helps in execution of the shell script based on Choice.

E*XAMPLE* The if construct is:

```
if command
then
block of statements
fi
-------------------------------------
if [ expression ]
then
block of statements
fi
-------------------------------------
```

The if/else/else if construct is:

```
if command
then
block of statements
elif command
then
block of statements
```

The case command construct is:

```
case variable_name in
pattern1)
statements
;;
pattern2)
statements
;;
pattern3)
;;
*) default value
;;
esac
case "$color" in
blue)
echo $color is blue
;;
green)
```

```
elif command
then
```

```
echo $color is green
;;
```

```
block of statements
else
block of statements
fi
------------------------------
if [ expression ]
then
block of statements
elif [ expression ]
then
block of statements
elif [ expression ]
then
block of statements
else
block of statements
fi
```

```
red|orange)
echo $color is red or orange
;;
*) echo "Not a color" # default
esac
```

The if/else construct is:

```
if [ expression ]
then
block of statements
else
block of statements
fi
--------------------------------------
```

```
block of statements
else
block of statements
fi
------------------------------
if [ expression ]
then
block of statements
```

```
red|orange)
echo $color is red or orange
;;
*) echo "Not a color" # default
esac
```

The if/else construct is:

```
if [ expression ]
then
```

```
elif [ expression ]
then
block of statements
elif [ expression ]
then
block of statements
else
block of statements
fi
--------------------------------------
```

07. LOOPS

There are three types of loops: while,until and for. The while loop is followed by a command or an expression enclosed in square brackets, a do keyword, a block of statements, and terminated with the done keyword. As long as the expression is true, the body of statements between do and done will be executed.

The until loop is just like the while loop, except the body of the loop will be executed as long as the expression is false.

The for loop used to iterate through a list of words, processing a word and then shifting it off, to process the next word. When all words have been shifted from the list, it ends. The for loop is followed by a variable name, the in keyword, and a list of words then a block of statements, and terminates with the done keyword.

The loop control commands are break and continue.

E*XAMPLE*

while command

```
block of statements
else
block of statements
fi
--------------------------------------
```

```
do
block of statements
done
------------
while [ expression ]
do
block of statements
done
until command for variable in word1 word2 word3
...
do do
block of statements block of statements
done done
------------
until [ expression ]
do
        block of statements
done
------------
until control command
do
        commands
done
```

08. Break Statement :

This command is used to jump out of the loop instantly, without waiting to get the control command.

09. ARRAYS

(positional parameters) The Bourne shell does support an array, but a word list can be created by using positional parameters. A list of words follows the built-in set command, and the words are accessed by position. Up to nine positions are allowed.The built-in shift command shifts off the first word on the left-hand side of the list. The individual words are accessed by position values starting at 1.

E*XAMPLE*

```
set word1 word2 word3
echo $1 $2 $3            Displays word1, word2, and word3
set apples peaches plums
shift                    Shifts off apples
echo $1                  Displays first element of the list
 echo $2                 Displays second element of the list
 echo $*                 Displays all elements of the list
```

Command substitution To assign the output of a UNIX/Linux command to a variable, or use the output of a command in a string, backquotes are used.

E*XAMPLE*

```
variable_name=`command`
echo $variable_name
now=`date`
echo $now
echo "Today is `date`"
```

10. FILE TESTING

The Bourne shell uses the test command to evaluate conditional expressions and has a built-in set of options for testing attributes of files, such as whether it is a directory, a plain file (not a directory), a readable file, and so forth.

E*XAMPLE*

-d *File is a directory*
-f *File exists and is not a directory*
–r *Current user can read the file*
–s *File is of nonzero size*
–w *Current user can write to the file*
–x *Current user can execute the file*

```
#!/bin/sh
1 if [ –f file ]
then
echo file exists
fi
2 if [ –d file ]
then
echo file is a directory
fi
3 if [ -s file ]
then
echo file is not of zero length
fi
4 if [ -r file -a -w file ]
then
echo file is readable and writable
fi
```

11. EXECUTION OF SHELL SCRIPT :

1.By using change mode command
2.$ chmod u + x sum.sh
3.$ sum.sh
or
$ sh sum.sh

EXPERIMENT NO.8

SHELL PROGRAMMING

Ex.No :8a CONCATENATION OF TWO STRINGS

Aim:

To write a shell program to concatenate two strings.

Algorithm:

Step1: Enter into the vi editor and go to the insert mode for entering the code

Step2: Read the first string.

Step3: Read the second string

Step4: Concatenate the two strings

Step5: Enter into the escape mode for the execution of the result and verify the output

Program:

```
echo “enter the first string”
read str1
echo “enter the second string”
read str2
echo “the concatenated string is” $str1$str2
```

Sample I/P:

```
Enter first string: Hello
Enter first string: World
```

Sample O/P:

```
The concatenated string is HelloWorld
```

Result:

Thus the shell program to concatenate two strings is executed and output is verified successfully.

Ex.No. :8b COMPARISON OF TWO STRINGS

Aim:

To write a shell program to compare the two strings.

Algorithm:

Step1: Enter into the vi editor and go to the insert mode for entering the code

Step2: Read the first string.

Step3: Read the second string

Step4: Compare the two strings using the if loop

Step5: If the condition satisfies then print that two strings are equal else print two strings are not equal.

Step6: Enter into the escape mode for the execution of the result and verify the output

Program:

```
echo “enter the first string”
read str1
echo “enter the second string”
read str2
if [ $str1 = $str2 ]
then
echo “strings are equal”
else
echo “strings are unequal”
fi
```

Sample I/P:1

```
Enter first string: hai
Enter second string: hai
```

Sample O/P:1

```
The two strings are equal
```

Sample I/P:2

```
Enter first string: hai
Enter second string: cse
```

Sample O/P:2

```
The two strings are not equal
```

Result:

Thus the shell program to compare the two strings is executed and output is verified successfully.

Ex.No:8c MAXIMUM OF THREE NUMBERS

Aim:

To write a shell program to find greatest of three numbers.

Algorithm:

Step1: Declare the three variables.

Step2: Check if A is greater than B and C.

Step3: If so print A is greater.

Step4: Else check if B is greater than C.

Step5: If so print B is greater.

Step6: Else print C is greater.

Program:

```
echo "enter A"
read a
echo "enter B"
read b
echo "enter C"
read c
if [ $a -gt $b -a $a -gt $c ]
then
echo "A is greater"
elif [ $b -gt $a -a $b -gt $c ]
then
echo "B is greater"
else
echo "C is greater"
fi
```

Sample I/P:

```
Enter A:23
Enter B:45
Enter C:67
```

Sample O/P:

```
C is greater
```

Result:

Thus the shell program to find the maximum of three numbers is executed and output is verified successfully.

Ex.No:8d FIBONACCI SERIES

Aim:

To write a shell program to generate fibonacci series.

Algorithm :

Step 1 : Initialise a to 0 and b to 1.

Step 2 : Print the values of 'a' and 'b'.

Step 3 : Add the values of 'a' and 'b'. Store the added value in variable 'c'.

Step 4 : Print the value of 'c'.

Step 5 : Initialise 'a' to 'b' and 'b' to 'c'.

Step 6 : Repeat the steps 3,4,5 till the value of 'a' is less than 10.

Program :

```
echo enter the number
read n
a=-1
b=1
i=0
while [ $i –le $n ]
do
t=`expr $a + $b`
echo $t
a=$b
b=$t
i=`expr $i + 1
done
```

Sample I/P :

Enter the no: 5

Sample O/P:

```
0
1
```

```
1
2
3
5
```

Result :

Thus the shell program to find the fibonacci series is executed and output is verified successfully.

Ex.No:8e ARITHMETIC OPERATIONS USING CASE

Aim:

To write a shell program to perform the arithmetic operations using case

Algorithm :

Step 1 : Read the input variables and assign the value

Step 2 : Print the various arithmetic operations which we are going to perform

Step 3 : Using the case operator assign the various functions for the arithmetic operators.

Step 4 : Check the values for all the corresponding operations.

Step 5 : Print the result and stop the execution.

Program :

```
echo 1.Addition
echo 2.Subraction
echo 3.Multiplication
echo 4.Division
echo enter your choice
read a
echo enter the value of b
read b
echo enter the value of c
read c
echo b is $b c is $c
case $a in
```

```
1)d=`expr $b + $c`
echo the sum is $d
;;
2)d=`expr $b - $c`
echo the difference is $d
;;
3)d=`expr $b \* $c`
echo the product is $d
;;
4)d=`expr $b / $c`
echo the quotient is $d
;;
esac
```

Sample I/P :

```
1.Addition
2.Subraction
3.Multiplication

Division
Enter your choice:1
Enter the value of b:3
Enter the value of c:4
b is 3 c is 4
the sum is 7
```

Sample O/P:

```
b is 3 c is 4
the sum is 7
```

Result :

Thus the shell program to perform arithmetic operations using case is executed and output is verified successfully.