**Experiment:01**

**OS INSTALLATION**

1. Types of OS installation

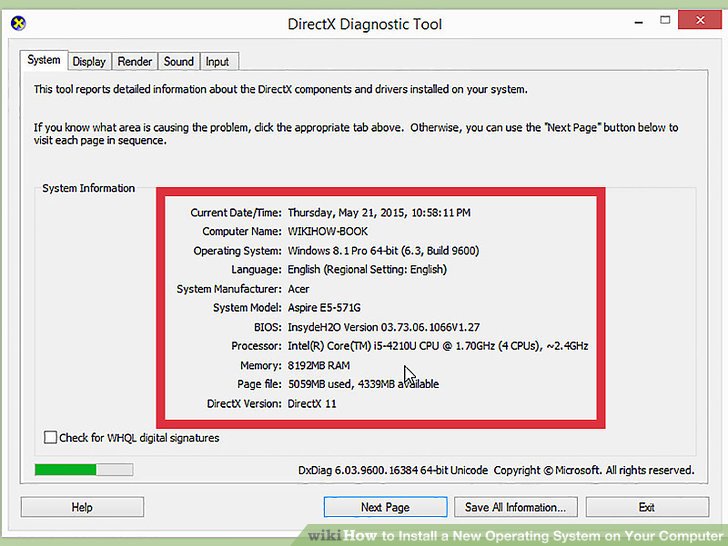
2. Boot methods

3. File System and formatting

4. Post installation tasks

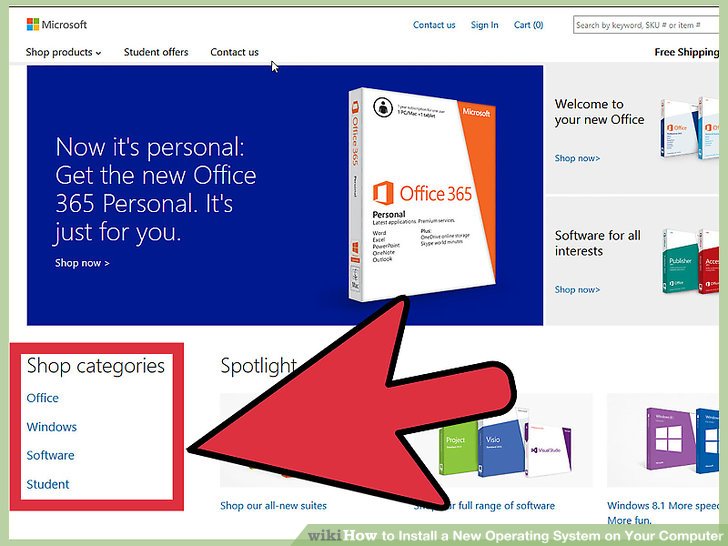
# [How to Install a New Operating System on Your Computer](https://www.wikihow.com/Install-a-New-Operating-System-on-Your-Computer)

**Method 1:determining which os to install**



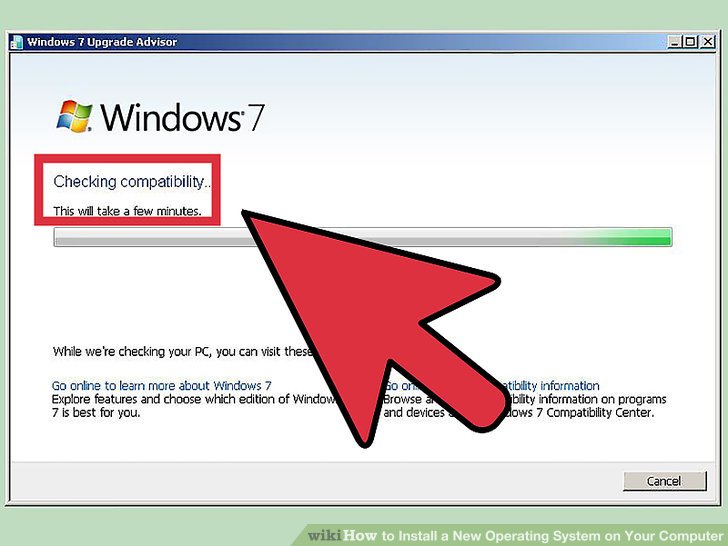
**1)Check the system requirements.** If you've decided that you want to install a new operating system, you'll first need to figure out which one you want to use. Operating systems have varying system requirements, so if you have an older computer, make sure that you can handle a newer operating system.

* Most Windows installations require at least 1 GB of RAM, and at least 15-20 GB of hard disk space. Also, your CPU needs to be powerful enough to run the OS you want to run. Make sure that your computer can accommodate this. If not, you may need to install an older operating system, such as Windows XP.
* Linux operating systems typically don't require as much space and computing power as Windows operating systems. The requirements vary depending on the distribution you choose (Ubuntu, Fedora, Mint, etc.).



**2**

**Decide whether to purchase or download.** Windows licenses need to be purchased. Each license comes with a key good for one installation. Most Linux distributions are free to download and install as much as you'd like, though some Enterprise versions are closed and require purchase (Red Hat, SUSE, etc.).



**3**

**Research your software compatibility.** Make sure that the operating system you want to install supports the programs that you want to use. If you use Microsoft Office for work, you won't be able to install it on a Linux machine. There are substitute programs available, but the functionality may be limited.

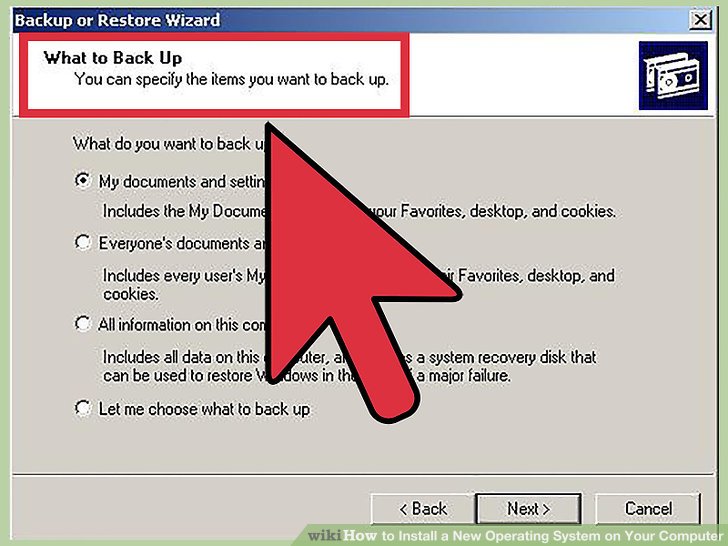
* Many games that work on Windows will not work on Linux. The number of supported titles is growing, but be aware that if you are an avid gamer your library might not transfer over smoothly.



**4**

**Obtain your new operating system.** If you've purchased a copy of Windows from a store, you should have received an installation disc along with your product code. If you don't have the disc but have a valid code, you can download a copy of the disc online. If you are installing Linux, you can download an ISO of the distribution from the developer's website.

* An ISO file is a disc image that needs to be [burned to a disc](https://www.wikihow.com/Burn-ISO-Files-to-DVD) or copied to a [bootable USB drive](https://www.wikihow.com/Make-a-USB-Bootable)



**5**

**Backup your data.** When you are installing a new operating system, you are most likely going to be wiping the hard drive in the process. This means that you will lose all of your files on the computer unless you back them up. Always make sure that any important files are copied to a backup location before starting the installation process. Use an external hard drive or burn the data to DVDs.

* If you are installing the operating system alongside your existing one, you most likely will not have to delete any data. It is still wise to backup important files just in case.
* You cannot backup programs; they will need to be reinstalled once you have finished installing your new operating system.

**Method 2: installing your new operating system**



**1**

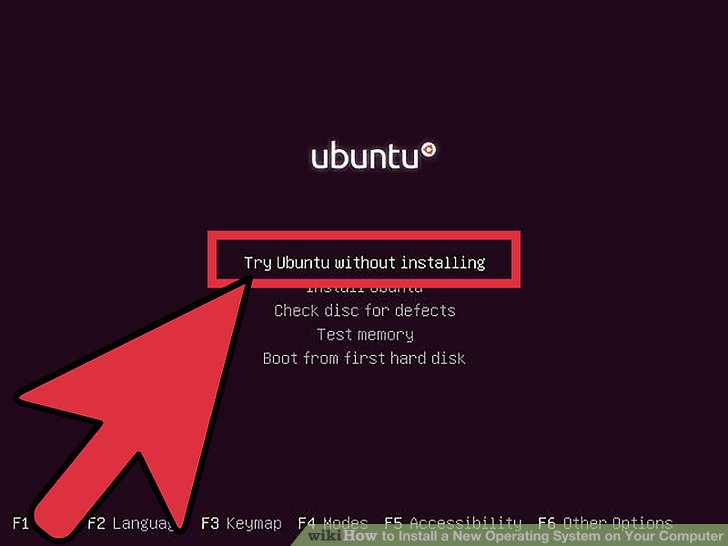
**Determine your installation order.** If you are installing a Linux distribution that you want to run alongside Windows, you need to install Windows first and then Linux. This is because Windows has a very strict boot loader that needs to be in place before Linux is installed, otherwise Windows won't load.



**2**

**Boot from your installation disc.** Insert the installation disc into your optical; drive, and reboot your computer. Normally a computer boots from the hard drive first, so you will need to adjust some settings in your BIOS in order to boot from the disc drive. You can enter the BIOS by hitting the designated Setup key during the boot process. The key will be displayed on the same screen as your manufacturer's logo.

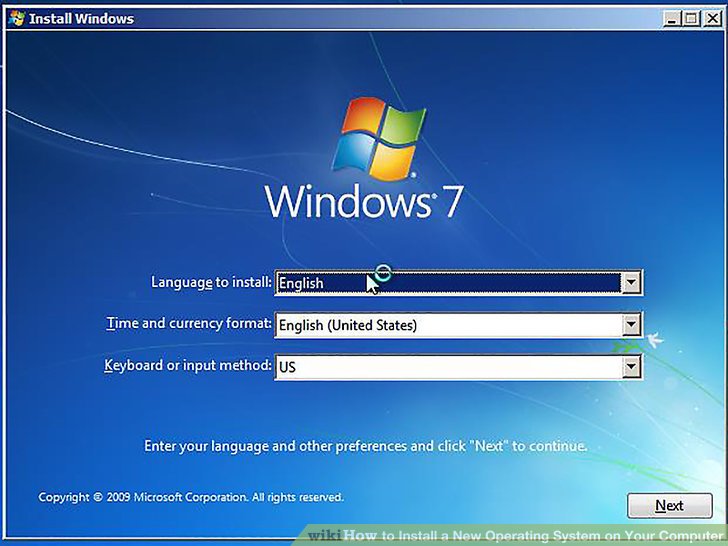
* Common Setup keys include F2, F10, F12, and Del/Delete.
* Once you are in the Setup menu, navigate to the Boot section. Set your DVD/CD drive as the first boot device. If you are installing from a USB drive, make sure that the drive is inserted and then select it as the first boot device.
* Once you've selected the correct drive, save your changes and exit Setup. Your computer will reboot.



**3**

**Try your Linux distribution before installing.** Most Linux distributions come with a copy that can be loaded directly from the installation disc. This will allow you to "test drive" your new operating system before you commit to the installation process. Once you are ready to install, click the Installation program on the desktop.

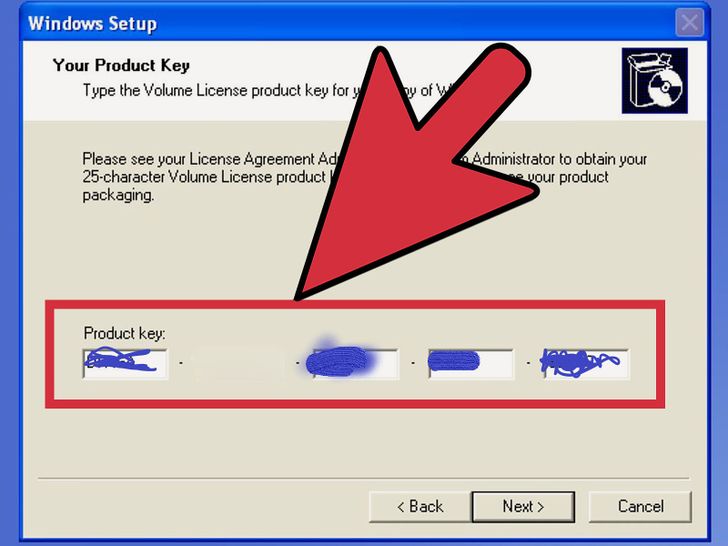
* This is only possible with Linux distributions. Windows does not allow you to test out the operating system before you install.



**4**

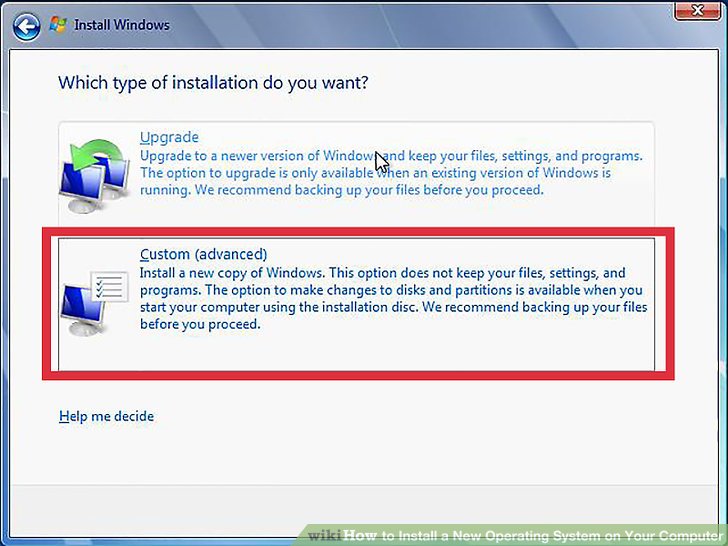
**Wait for the Setup program to load.** No matter which operating system you choose, the setup program will need to copy some files to your computer before it can continue. This can take several minutes, depending on the speed of your computer's hardware.

* You will most likely need to choose some basic options, such as language and keyboard layout.



**5**

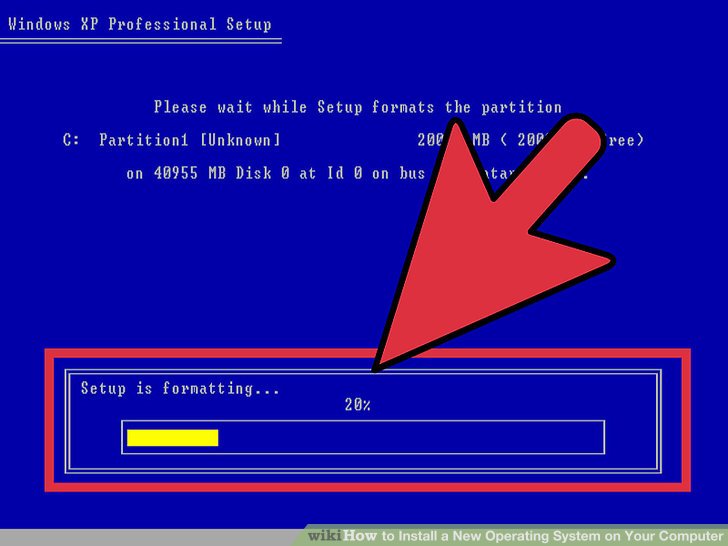
**Enter your product key.** If you are installing Windows 8, you will need to enter your product key before you can begin the installation. Older Windows versions will ask for the product key after installation is complete. Linux users will not need a product key unless it is a purchased version such as Red Hat.



**6**

**Choose your installation type.** Windows will give you the option of Upgrading or performing a Custom installation. Even if you are upgrading an older version of Windows, it is highly recommended that you choose Custom and start from scratch. This will minimize problems that may arise later from combining old settings and new ones.

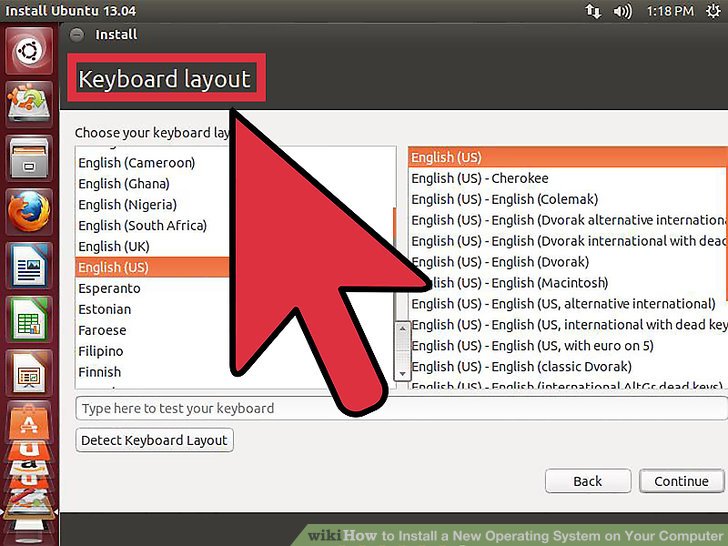
* If you are installing Linux, you will be given the option to install alongside your existing operating system (Windows), or to erase the disk and install Linux by itself. Choose the option that best meets your needs. If you choose to install alongside Windows, you will be given the option to choose how much hard disk space you want to designate for Linux.



**7**

**Format your partitions.** If you are installing Windows, you will need to choose which hard drive partition you want to install it on. Deleting partitions will wipe the data on the partition and return the space to the Unallocated section. Select the unallocated space and create a new partition.

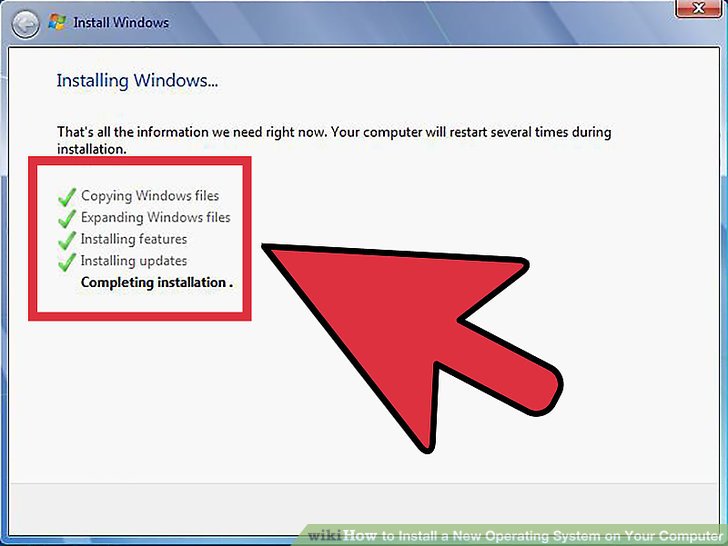
* If you are installing Linux, the partition needs to be formatted in the Ext4 format.



**8**

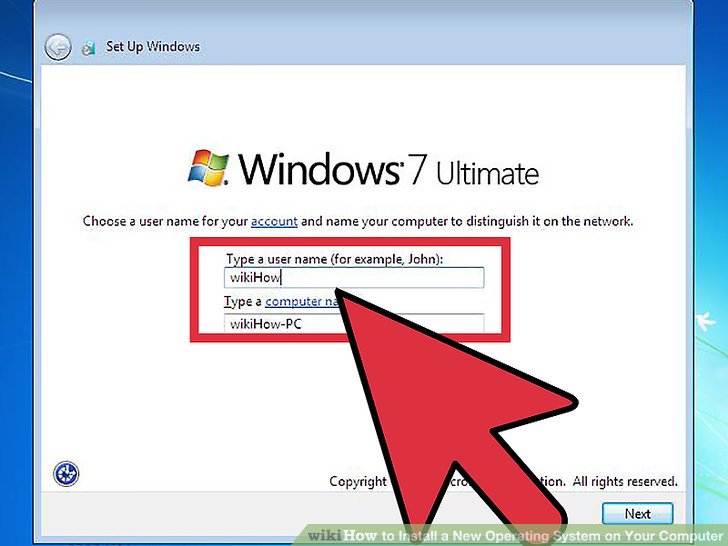
**Set your Linux options.** Before installation begins, your Linux installer will ask you for your timezone, and you will need to create a username and password. You will use this to log in to your Linux distribution as well as authorize system changes.

* Windows users will fill out personal information after the installation is complete.



**9**

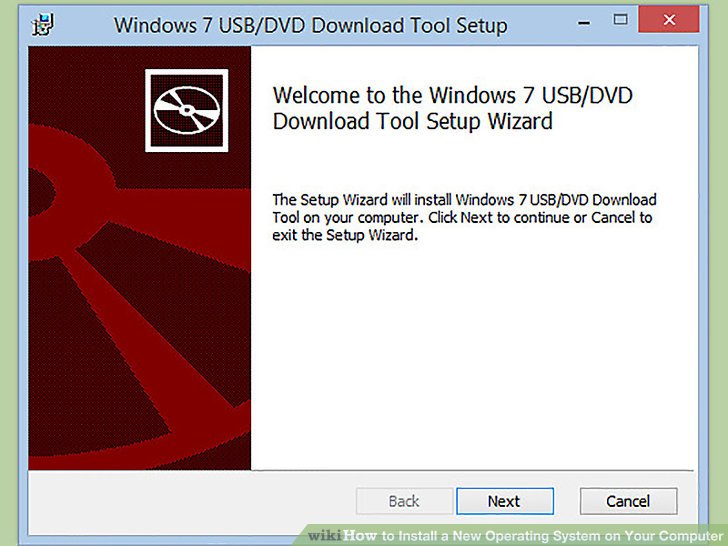
**Wait for the installation to complete.** Depending on the speed of your computer, this can take up to an hour to finish. Most installations are hands-off at this point. Your computer may reboot several times during the installation process.



**10**

**Create your Windows login.** Once your Windows installation is complete, you will need to create a username. You can also choose to create a password, though this is not necessary. After creating your login info, you will be asked for your product key.

* In Windows 8, you will be asked to customize the colors first. After that, you can choose to either log in with a Microsoft account or use a more traditional Windows username.



**11**

**Install your drivers and programs.** Once the installation is complete, you will be taken to your new desktop. From here, you can begin installing your programs and make sure that your [drivers are installed and up to date](https://www.wikihow.com/Find-and-Update-Drivers). Make sure to install an [antivirus program](https://www.wikihow.com/Install-an-Antivirus) if you are going to be connecting to the internet.

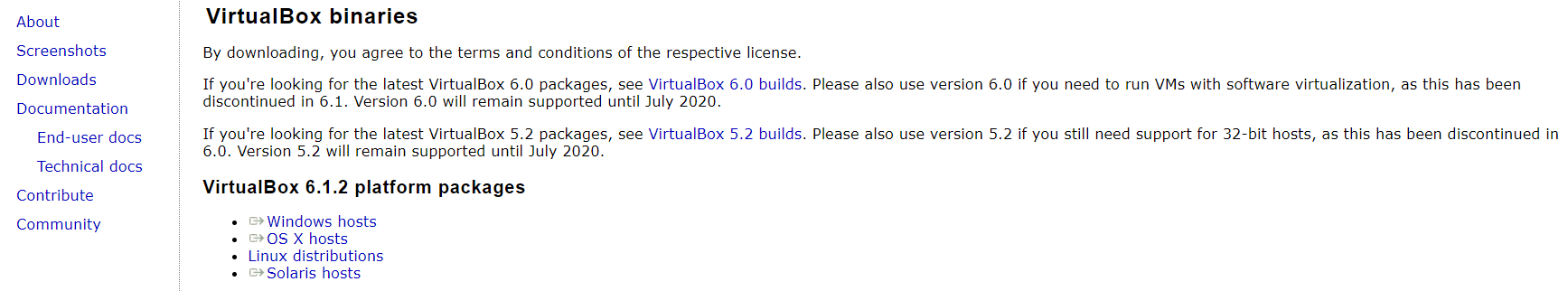
**Experiment:02**

1. **Install and configure virtual machine / virtual box on host machine**

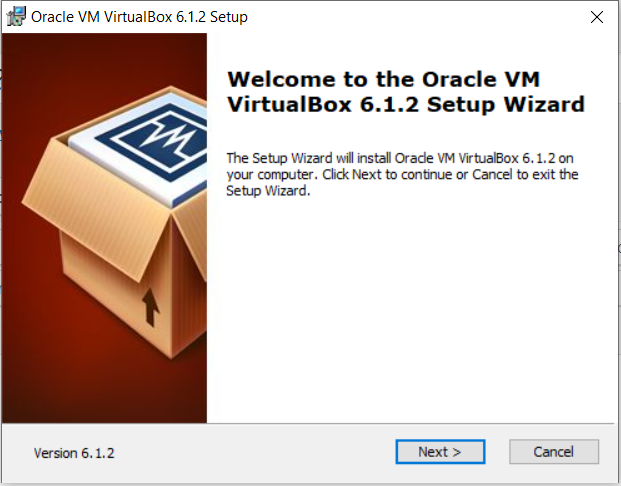
**Virtual Machine** abstracts the hardware of our personal computers such as CPU, disk drives, memory, NIC (Network Interface Card), etc, into many different execution environments as per our requirements, hence giving us a feeling that each execution environment is a single computer.Forexample,VirtualBox.  
We can create a virtual machine for several reasons, all of which are fundamentally related to the ability to share the same basic hardware yet can also support different execution environments, i.e., different operating systems simultaneously.

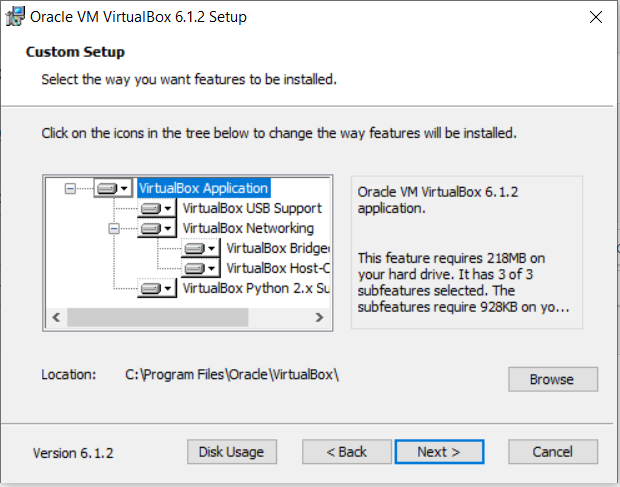
**Downloading and Installing VirtualBox**

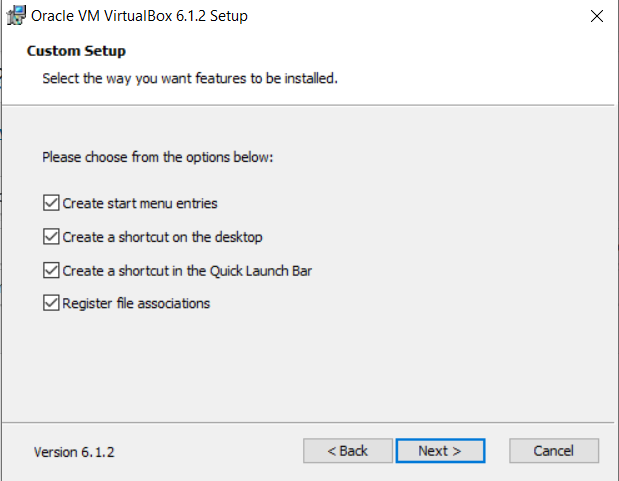
To download VirtualBox, go to the official site [**virtualbox.org**](https://www.virtualbox.org/wiki/Downloads) and download the latest version for windows.

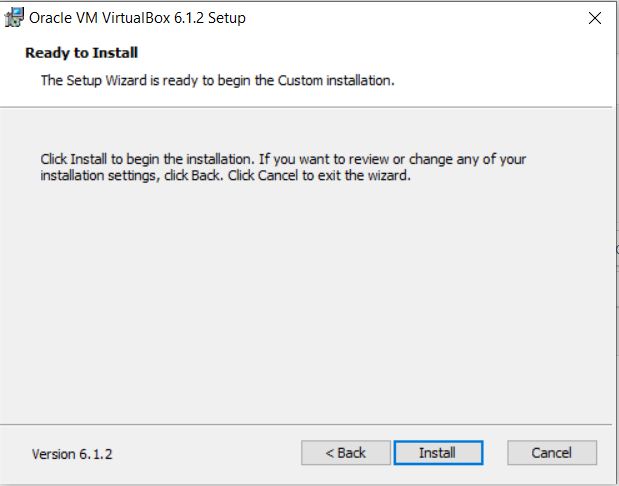


**Beginning with the Installation:**

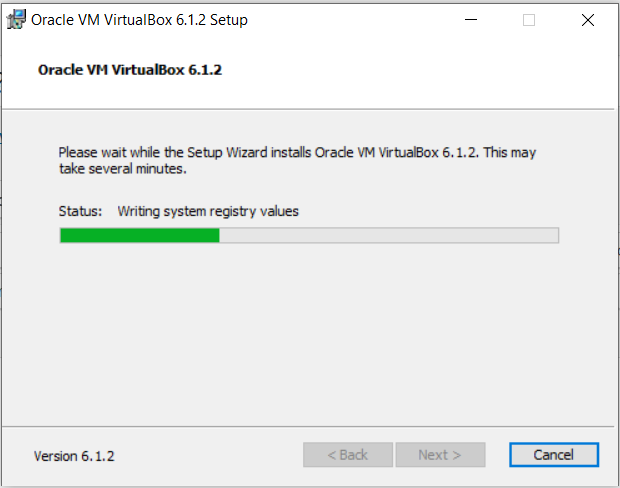
* **GettingStarted:**  
  
* **Select Installation Location:**



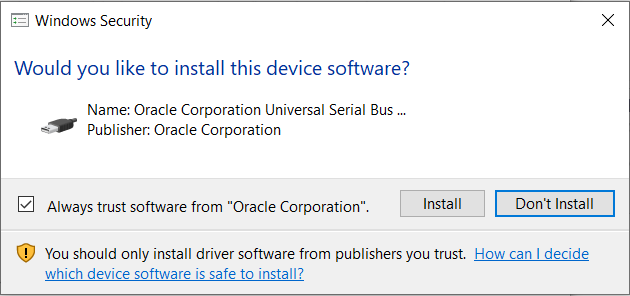
* **Creating-Entries-and-Shortcuts:**  
  
* **Ready to Install:**



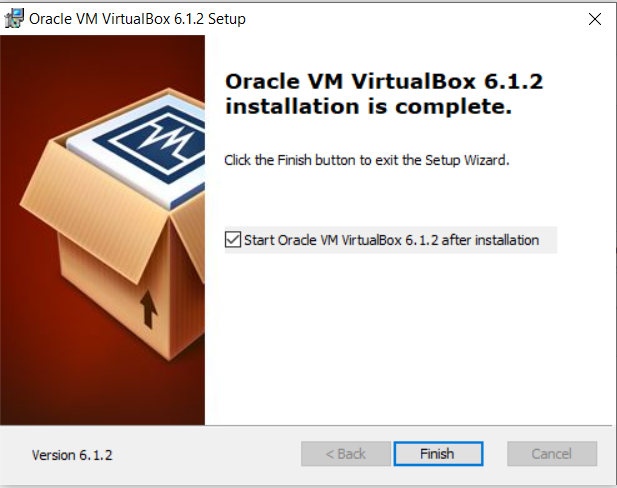
* **Installing Files and packages:**



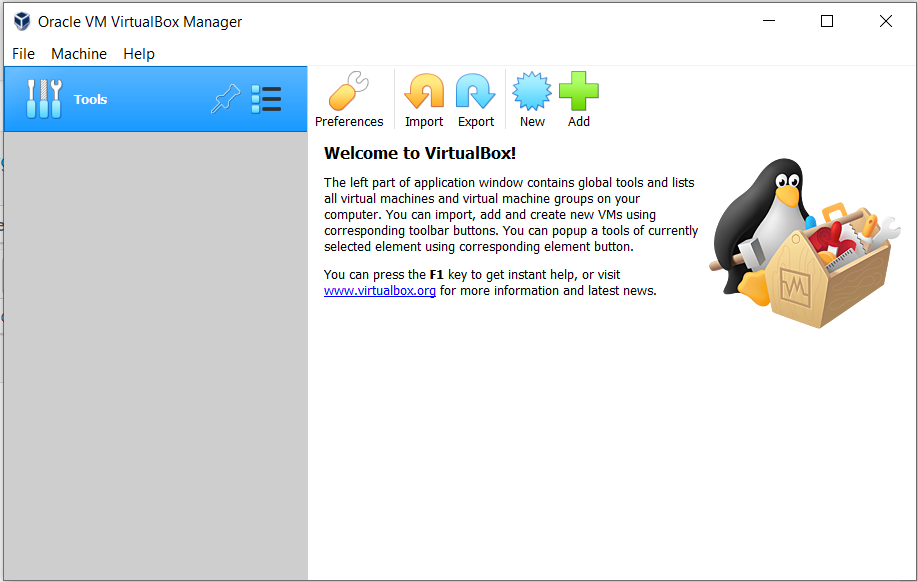
* **Installing Certificates:**



* **Finished Installation:**



When you will open virtualbox it will look like as shown below:



**Experiment:03**

|  |  |
| --- | --- |
| ***. Internal and External command in Linux*** | |
| **Internal command** | **External command** | |
| These commands are built within the shell, they are loaded during booting. | These commands are not built within the shell, they are stored as *“separate binaries”* | |
| Shell does not start separate process to run the internal commands. | Shell starts separate sub-process to execute. | |
| Examples:**type, echo, cd, pwd** | Examples:**ls, cp, mv, rm, cat** | |

# Internal commands

|  |
| --- |
| **1.type** command |

type is a internal command that is used to check whether the given command is internal or external command.

|  |  |
| --- | --- |
| **Syntax** | $ **type** [options] *command\_name* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **type** *echo* | **echo** is a shell builtin |
| **2** | $ **type***type* | **type** is a shell builtin |
| **3** | $ **type***cat* | **cat** is /bin/cat |
| **4** | $ **type***rm* | **rm**is /bin/rm |

|  |  |  |
| --- | --- | --- |
| **SN** | **Syntax with options** | **Meaning** |
| 1 | $ **type - t** *cmd name* | Option **t** tellswhat the command is: function or builtinetc.. |
| 2 | $ **type - p** *cmd name* | It displays the absolute path of the command |
| 3 | $ **type - a** *cmd name* | It displays all path which contains command name |
| **SN** | **Example** | **Output** |
| **1** | $ **type - t** *date* | file |
| **2** | $ **type - t** *declare* | builtin |
| **3** | $ **type - p** *date* | /bin/date |
| **4** | $ **type - a** *echo* | echo is a shell builtin echo is /bin/echo |

|  |
| --- |
| **2.echo** command |

echo is an internal command that is used to display. It can display a **character, string or value of any variable.**

|  |  |
| --- | --- |
| **Syntax** | $ **echo** [options] *strings* |

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Example** | | **Output** |
| **1** | $ **echo “**j” | | j |
| **2** | $ **echo**  “*Hello”* | | Hello |
| **3** | $ **echo** “*hello world”* | | hello world |
| **4** | $ x=10  $ **echo** “the value of x is $x” | | the value of x is 10 |
| **SN** | **Syntax with options** | **Meaning** | | |
| 1 | $ **echo -e** *string* | e is interpreter for escaped characters such as \b, \n, \t etc | | |
| **SN** | **Example** | **Output** | | |
| **1** | $ **echo -e** *“linux****\b****prgmng”* | linuxprgmng [ it has removed the blank space] | | |
| **2** | $ **echo -e** *“linux****\n****prgmng”* | linuxprgmng | | |

|  |
| --- |
| **3.cd** command |

cd is an internal command that changes the current working directory. cd stands for “change directory”

|  |  |
| --- | --- |
| **Syntax** | $ **cd** [options] *directory name* |

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Example** | | **Output** |
| **1** | $ **cd** gptkgf | | ......**/**gptkgf**$** |
| **2** | $ **cd /** | | Moves back to home directory |
| **3** | $ **cd..** | | Moves back to home directory |
| **SN** | **Example** (with options) | **Output** | | |
| **1** | $ **cd -L** *gptkgf* | Changes to directory *gptkgf*& makes symbolic link to point to *gptkgf* directory | | |
| **2** | $ **cd -P** *gptkgf* | Changes to directory *gptkgf*& does not make symbolic link to point to *gptkgf* directory | | |

|  |
| --- |
| **4.pwd** command |

pwd is an internal command that is used to print the name of working directory. cd stands for “present working directory”

|  |  |
| --- | --- |
| **Syntax** | $ **pwd** [options] |

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** |  | **Example** | **Output** |
| **1** | $ **pwd** |  | /home/username |

# External commands

|  |
| --- |
| **1.ls** command |

ls is an external command that is used to list the files and directories present in the system

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **ls** [options] *filename/dirname* |
| **SN** | **Example** | | **Output** | | |
| **1** | $ **ls** | | Desktop Downloads.... Documents...... | | |
| **SN** | **Example with options** | | **Meaning** | | |
| **1** | $ **ls -**r | | Lists all files & directories in reverse order (z-a) | | |
| **2** | $ **ls -**ls | | Lists all files in long format along with file size | | |
| **3** | $ **ls -**t | | Lists all files sorted by time and date | | |
| **4** | $ **ls -**a | | Lists all files & also hidden files starting with **.**(dot) | | |
| **5** | $ **ls -**l | | Lists all files & directories in a long list format | | |

|  |
| --- |
| **2.cp**command |

cp is an external command that is used to copy the contents of files and directories to other files and directories.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **cp** [options] *source destination* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **cp**f1.c f2.c | | Copies the contents of **f1.c** to **f2.c** | | |
| **SN** | **Example with options** | | **Meaning** | | |
| **1** | $ **cp - a** f1.c gptkgf | | Copies the contents of **f1.c** to the directory **gptkgf** | | |
| **2** | $ **cp - R** f1.c gptkgf | | Copies the contents of **f1.c** to the directory **gptkgf** | | |
| **3** | $ **cp - i**f1.c gptkgf | | Same as above but there is interactive prompt before copying **cp:overwrite „gptkgf/f1.c‟ ?** y | | |

|  |
| --- |
| **3.mv** command |

mv is an external command that is used to move the files and directories.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **mv** [options] *source destination* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **mv** f1.c gptkgf | | Moves **f1.c** to the directory *gptkgf* | | |
| **SN** | **Example with options** | | **Meaning** | | |
| **1** | $ **mv - i**f1.c gptkgf | | Same as above but there is interactive prompt before moving **mv:move „gptkgf/f1.c‟ ?** y | | |

## **4. cat** command

cat is an external command that is used to **a.**display the contents of the file

**b.**modify/concatenate the contents of the file

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **cat** [options] *file name* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **cat** f1.c | | Displays the contents of the file f1.c | | |
| **2** | $ **cat** f1.c f2.c | | Concatenates the contents of the file f1.c and f2.c and displays it | | |

# Other commands

|  |
| --- |
| **1.tput clear** command |

It echo‟s the clear screen sequence for the current terminal.

|  |  |
| --- | --- |
| **Syntax** | $ **tput clear** |

|  |
| --- |
| **2.who** command |

It displays information on who is logged onto the system currently.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **who** [options] |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **who** | | It displays information on who is logged onto the system currently. | | |
| **2** | $ **who -H** | | It displays information on who is logged onto the system currently along with HEADING | | |
| **3** | $ **who -q** | | It displays all login names and count of all logged-on users | | |
| **4** | $ **who -b** | | It displays the time of last system boot. | | |

|  |
| --- |
| **3.cal**command |

It displays current month with the current day highlighted.

**Syntax**

$

**cal**

[

options]

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Meaning** |
| **1** | $ **cal** | Displays current month with the current day highlighted |
| **2** | $ **cal -h** | Displays current month without highlighting the current day |
| **3** | $ **cal -m** monthname | Displays the calendar for the specified month |
| **4** | $ **cal -y** | Displays the calendar of current year |
| **5** | $ **cal -y 2017** | Displays the calendar of the year 2017 |

## **4. date** command

It is used to display the current date and time.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **date** [options] |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **date** | | Displays the current date and time. | | |
| **2** | $ **date +%d** | | Displays the current day number. | | |

|  |
| --- |
| **5.bc**command |

It is used to perform arithmetic calculations.

|  |  |
| --- | --- |
| **Syntax** | $ **bc** [options] |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **echo** “2+3” | bc | 5 |
| **2** | $ **echo** “2-3” | bc | -1 |
| **3** | $ **echo** “scale=2; 4/2” | bc | 2.00 |
| **4** | $ **echo** “scale=3; 4/2” | bc | 2.000 |

|  |
| --- |
| **6.man** command |

It is used to display man pages. They are builtin.

man stands for manual.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **man** [options] |
| **SN** | **Example** | | **Output** | | |
| **1** | $ **man** ls | | Displays information about ls command. Press **h** for help and **q** to quit | | |
| **2** | $ **man** cp | | Displays information about cp command. Press **h** for help and **q** to quit | | |

|  |
| --- |
| **7.passwd**command |

It is used to change the password of the user account

|  |  |
| --- | --- |
| **Syntax** | $ **passwd** |

**Ex:** $passwd Changing password for abc (Current) UNIX password:

Enter new UNIX password:

Retype new UNIX password:

## **8. uname**command

It is used to print information about the linux system

|  |  |
| --- | --- |
| **Syntax** | $ **uname** [options] |
| **SN** | **Example** | **Output** | |
| **1** | $ **uname** | Prints Linux | |
| **2** | $ **uname -s** | Prints kernel name | |
| **3** | $ **uname -a** | Prints all information | |
| **4** | $ **uname -r** | Prints kernel release information | |
| **5** | $ **uname -v** | Prints kernel version information | |
| **6** | $ **uname -m**  $ **uname -i**  $ **uname -p** | Prints machine hardware name, hardware platform, processor names respectively. (all outputs are same) | |
| **7** | $ **uname -o** | Prints the OS name | |

**Experiment:04**

|  |
| --- |
| ***. Working with files & directories.*** |

**Know the categories of files.**

* **Files:** On Linux system everything is considered as “files”
* Linux classifies files into 3 main categories 1. Regular files
  1. Directory files
  2. Special files

Block file Character file Named pipe file Symbolic link file Socket file

**Regular files:-** Includes all normal files such as text, image, executable files. **Directory files:-** They are collection of many files.

### Special files:-

1. **Block files:** They are hardware files present in the directory /dev
2. **Character file:** They provide stream of input and output. Ex: Terminal
3. **Named pipe file:** Communication between 2 process is done with the help of pipe file without network connectivity.
4. **Symbolic link file:** These are files linked to other files.
5. **Socket file:** Same as *symbolic link file* but communication is done with the help of network connectivity.

**Directory related Commands**– pwd, mkdir, rmdir, cd, ls

**(**pwd, cd and ls commands already we have discussed**)**

|  |
| --- |
| **1.mkdir**command |

This command is used to create or make new directory.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **mkdir** [options] *directoryname* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **mkdir***cse* | | creates a new directory called *“cse”* | | |
| **2** | $ **mkdir***csece me* | | creates 3 new directory namely *“cse”, “ce”, “me”* | | |
| **3** | $ **mkdir – v** *cse* | | creates a new directory called *“cse”* in verbose mode(v) it prints confirmation message *directory is created* | | |

|  |
| --- |
| **2.rmdir**command |

This command is used to remove empty directories present in your system.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **rmdir** [options] *directoryname* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **mkdir***cse* | | removes the directory called *“cse”* | | |
| **3** | $ **mkdir – v** *cse* | | removes the directory called *“cse”* in verbose mode | | |

**Manipulating Absolute paths and Relative paths using cd command.**

**Path:** Location of a file or a directory in the file system of the operating system is called as “path” There are two types of path

1. absolute path
2. relative path

**Absolute path:** The path that begins with / is called as absolute path. It contains complete path. **Ex:** /var/ftp

**Relative path:** The path related from the present working directory is called as relative path. It does not contain complete path.

**Ex:** Assume that you are in **/var/log** and want to change to **/var/log/kernel** then you can use relative path. Present working directory is: **/var/log** Type **cd kernel**

so now you will go inside **/var/log/kernel**

**Ex‟s:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***CURRENT DIR*** | ***CHANGE TO*** | ***ABSOLUTE PATH*** | ***RELATIVE PATH*** |
| /etc/samba | /etc | **cd** /etc | **cd** .. or **cd** / |
| /var/ftp | /var/log | **cd** /var/log | **cd** ..**cd** log or type **cd** ../log |
| /etc/lvm | /opt/uradba | **cd** /opt/uradba | **cd** .. /../ opt/uradba |

**File related Commands – cat, cp, mv, rm, comm, cmp, diff, tar, umask, wc**

**(**cat, cp, mv commands already we have discussed**)**

|  |
| --- |
| **1.rm**command |

This command is used to remove or delete files.

|  |  |
| --- | --- |
| **Syntax** | $ **rm** [options] *filename / directoryname* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Meaning** |
| **1** | $ **rm***f1.c* | removes the file *“f1.c”* |
| **2** | $ **rm -i***f1.c* | prompt before deleting the file *“f1.c”* |
| **3** | $ **rm - d** *cse* | removes the directory *“cse”* |
| **4** | $ **rm***\*.c* | removes all the **c** files |

|  |
| --- |
| **2.comm**command |

This command is used to compare two files line by line.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **comm** [options] *filen1 filen2* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **comm***f1.txt f2.txt* | | Compares the files  *f1.txt* and *f2.txt* | | |

***Output:***

|  |  |  |
| --- | --- | --- |
| **f1.txt**  cheese curd milk |  | **f2.txt**  butter cheese milk |

|  |
| --- |
| **output has three columns butter**  **1st column** contains line unique to f1.txt  **2nd column** contains line unique to f2.txt **3rd column** contains lines common to both files |

butter

cheese

curd

milk

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | | $ **comm - 1** *f1.txt f2.txt* | Compares the files  *f1.txt* and *f2.txt* and removes the column 1 from output |
| **3** | | $ **comm - 2** *f1.txt f2.txt* | Compares the files  *f1.txt* and *f2.txt* and removes the column 2 from output |
| **3.cmp**command | | |

This command is used to compare two files byte by byte. It displays the byte and line number where first difference is found. If no difference is found no output will be displayed.

|  |  |  |
| --- | --- | --- |
| **Syntax** | | $ **cmp** [options] *file1 file2* |
| **SN** | **Example** | | **Meaning** | | |
| **1** | $ **cmp***f1.txt f2.txt* | | Compares the files  *f1.txt* and *f2.txt* ***Output:-*** *differ: byte 15, line 3* | | |

|  |
| --- |
| **4.diff** command |

This command is used to analyse two files and prints the lines that are different.

|  |  |
| --- | --- |
| **Syntax** | $ **diff** *file1 file2* |

### Ex1

|  |  |  |
| --- | --- | --- |
| **f1.txt**  cheese curd milk |  | **f2.txt**  butter cheese milk |

|  |
| --- |
| **a** = add **<**first file  **d**  = delete **>** second file **c** = change  **0 a 1** means, **0** corresponds to file1 **add** and **1** corresponds to file2 |

***Output:***

0 a 1

> butter

2 d 2

< curd

**Ex2  *Output:***

|  |  |  |
| --- | --- | --- |
| **c.txt**  apples grapes pineapple |  | **cc.txt**  apples mango pineapple |

2 c 2

< grapes

> mango

|  |
| --- |
| **7.wc** command |

This command is used to count number of lines, number of words, and number of characters (byte) present in the file.

|  |  |
| --- | --- |
| **Syntax** | $ **wc** [options] *filename* |

|  |
| --- |
| **Experiment:05**  . **Basic File attributes.** |

|  |
| --- |
| **Listing seven attributes of a file : ls and its options** |

**ls – l**

* This command is used for **listing** (displaying) file attributes.
* It displays the **seven attributes** of all files which are present in the current directory.
* Consider the following output of **ls – l** command total 48

**d**rwxrwxr-x 2 xyz xyz 4096 Feb 15 07:58 n **-**rw-rw-r- - 1 xyz xyz 6 Feb 15 08:00 w.txt

**.** . . .. . . .. .. . .. . ... . ... ..

**.** .. . . . . . .. . .. .. . .. .. .. ..

|  |  |  |
| --- | --- | --- |
|  | **total 48** | Indicates that **48** blocks are occupied by these files on the disk |
| **1** | File type and  permissions (first  column) | 1st character indicates that file is ordinary file or directory There are 3 types of permission:**read, write & execute r w x | r w x | r – x user group others** |
| **2** | Second column | Indicates number of links associated with the file |
| **3** | Third column | Ownership (owner name is **xyz** ) |
| **4** | Fourth column | Group owner file name (group owner name is **xyz** ) |
| **5** | Fifth column | File size in bytes |
| **6** | Sixth column | File modification time and date |
| **7** | Seventh column | Specifies the filename |

|  |
| --- |
| **Manipulating File permissions using “chmod” command** |

### chmodcommand

* Thiscommand is used for changing the file permissions.
* **chmod**stands for “change mode”
* We can change permission for all 3 categories of user (user, group, others)
* The command can be specified in **two** ways

#### 1.Relative permissions

When using this method, chmod changes the permissions specified in the command and leaves the other permission unchanged.

|  |  |
| --- | --- |
| **Syntax** | $ **chmod** [options] permissions *filename* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **chmod**u + x f.txt | Adds execute permission on file f.txt for the **user** |
| **2** | $ **chmod**o + x f.txt | Adds execute permission on file f.txt for the **others** |
| **3** | $ **chmod**o - x f.txt | Removes execute permission on file f.txt for the **others** |
| **4** | $ **chmod**a + x f.txt | Adds execute permission on file f.txt for the **user, group andothers (all)** |
| **5** | $ **chmod**o + x f.txtf1.txt | Adds execute permission on file f.txt and f1.txt for the **others** |

#### 2.Absolute permissions

In this method **chmod**uses 3-digit code. Digits assigned for read = 4 write = 2

execute = 1

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **chmod**777 f.txt | Adds read, write and execute permission on file f.txt for **all** |
| **2** | $ **chmod**766 f.txt | User = r w x  Group = r w  Others = r w |

|  |
| --- |
| **Manipulating Hardlink and Softlink using ln command** |

### ln command

* Thiscommand is used to create link between files.

|  |  |
| --- | --- |
| **Syntax** | $ **ln**  [options] *filen1 file2* |

* There are two types of link
  1. Hard link
  2. Soft link

### Hard link

|  |  |  |
| --- | --- | --- |
| **f1.txt**  good morning |  | **f2.txt**  good morning |

#### 

#### Create two files

* $ **ln** f1.txt f2.txt
* Here each file have separate link to the contents.
* So changing or deleting **f2.txt** does not affect **f1.txt**
* Similarly changing or deleting **f1.txt** does not affect **f2.txt**

**2. Soft link**

|  |  |  |
| --- | --- | --- |
| **f1.txt**  good morning |  | **f2.txt**  good morning |

#### Create two files

f1.txt

good morning

f2.txt

* $ **ln - s** f1.txt f2.txt
* Here **f2.txt** have link to **f1.txt**
* So changing or deleting **f1.txt** affects **f2.txt**

**Experiment:06**

|  |
| --- |
| **Learn to use vi editor** |

* vi is the file editor in UNIX and vi stands for visual instrument.
* There are 3 modes in vi editor

1) command mode 2) insert mode 3) Ex mode or last line mode **Command mode**

* When we start up vi, we are in the command mode.
* This mode allows us to use certain commands to edit the file or it allows us to change to other modes.

|  |  |
| --- | --- |
| **Command mode commands** | **Meaning** |
| x | Deletes the character under the cursor |
| dd | Deletes entire line |
| dw | Deletes the word under the cursor |
| r | Replaces the single character under the cursor |
| ~ | Changes the case of the letter under the cursor from upper to lower or lower to upper |
| h, j, k, l | To move cursor left, down, up, right |
| w | Moves the cursor to the beginning of next word |
| b | Moves the cursor to the beginning of previous word |
| 0 (zero) | Moves the cursor to the beginning of the current line |
| $ | Moves the cursor to the end of the current line |
| G | Moves the cursor to the end of the file |

### Insert or input mode

* Press i (insert) or a (append) to move to insert mode where you can type the contents of the file.
* Press Esc to return to command mode.

### Ex or last line mode

* This mode helps us to give some extended commands to vi.
* When we type : (colon) we move to last line mode.

|  |  |  |
| --- | --- | --- |
| **Ex mode commands** | **Meaning** | |
| **:wq** and press enter | Save (write) and quit | |
| **:w** and press enter | Save and don‟t quit | |
| **!q** and press enter | Quit the file without saving | |
| **Experiment:07**  **Simple Filters** – head, tail, cut, paste, sort, uniq, tr, pr. | | |

* Filters read data from standard input (keyboard) and write the data to standard output (screen). Many filters are available in UNIX such as head, tail, cut, paste, sort, uniq, tr, pr.

|  |
| --- |
| **head** command |

This command prints the FIRST 10 lines of the specified file

|  |  |
| --- | --- |
| **Syntax** | $ **head**  [options] *filename* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **head** f1.txt | prints the first 10 lines of the file **f1.txt** |
| **2** | $ **head - 4** f1.txt | prints only the first 4 lines of the file **f1.txt** |
| **3** | $ **head - c 4** f1.txt | prints only the first 4 characters of the file **f1.txt** |

|  |
| --- |
| **tail** command |

This command prints the LAST 10 lines of the specified file

|  |  |
| --- | --- |
| **Syntax** | $ **tail**  [options] *filename* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **tail** f1.txt | prints the last 10 lines of the file **f1.txt** |
| **2** | $ **tail - 4** f1.txt | prints only the last 4 lines of the file **f1.txt** |
| **3** | $ **tail - c 4** f1.txt | prints only the last 4 characters of the file **f1.txt** |

|  |
| --- |
| **cut** command |

This command removes or cuts the section of each line of a specified file.

|  |  |
| --- | --- |
| **Syntax** | $ **cut** [options] *filename* |

|  |
| --- |
| **cut.txt (use tab after each word)**  one two three four five six  apple grapes mango peach berries kiwi |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **cut - f 4** cut.txt | four peach |
| **2** | $ **cut - f 2-3** cut.txt | two three grapes mango |
| **3** | $ **cut - f 1, 4, 6** cut.txt | one four six apple peach kiwi |

In the above examples**f**stands for **field**

|  |
| --- |
| **paste** command |

This command displays the corresponding lines of multiple files side by side.

|  |  |
| --- | --- |
| **Syntax** | $ **paste** [options] *file1 file2* |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **paste***a.txt b.txt* | displays the content of **a.txt** and **b.txt** in parallel here tab is used as a delimiter |
| **2** | $ **paste** - d „:‟ *a.txt b.txt* | displays the content of **a.txt** and **b.txt** in parallel here : is used as a delimiter |
| **3** | $ **paste** - s *a.txt b.txt* | displays the content of **a.txt** and **b.txt** one after the another (in serial) |

|  |
| --- |
| **sort** command |

This command sorts the content of a file, line by line.

|  |  |
| --- | --- |
| **Syntax** | $ **sort** [options] *filename* |

|  |
| --- |
| **sort.txt (use tab after each word)**   1. joe 2. marie 3. albert 4. dave |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **sort***sort.txt* | sorts the content of the file **sort.txt**   1. joe 2. marie 3. albert 4. dave |
| **2** | $ **sort** - r *sort.txt* | sorts the content of the file **sort.txt** in **reverse order**  04dave  03 albert  02 marie  01 joe |
| **3** | $ **paste** - k 2 *sort.txt* | sorts the content of the file **sort.txt** based on the key (k) 2 represents the second coloumn   1. albert 2. dave 3. joe 4. marie |

**Experiment:08**

|  |  |
| --- | --- |
| **Process-Management commands** | . |

|  |
| --- |
| **Process creation, status, Identifying process, ps& its options** |

* Program under execution is called as **process**.
* A process is created when the user enters the command.
* Every process is given a unique number known as **Process ID (PID)**. So we can identify any process using its **PID**.
* Every process will have two ID‟s
  1. Its parent process ID called **PPID**
  2. Its own ID called **PID**
* Processes communicate with each other and share the resources such as CPU, main memory, hard disk.
* One process may **create** another process.
* Creating process is called as **“parent process”** and the created process is called as **“child process”**.

|  |
| --- |
| **ps command & its options** |

* **ps**stands for process status.
* This command is used to list the running process.

|  |  |
| --- | --- |
| **Syntax** | $ **ps** [options] |

|  |  |  |
| --- | --- | --- |
| **SN** | **Example** | **Output** |
| **1** | $ **ps** | **PID TTY TIME CMD**  1674 tty1 09:00:02 bash  17336 tty1 09:00:00 ps |
| **2** | $ **ps - e** | Displays every process running currently. |
| **3** | $ **ps - e | less** | Displays every process running currently and provides scroll to see the process and to press **q** to quit. |
| **4** | $ **ps - f** | **UID =** User ID, **PID** = process ID, **PPID** = parent process ID, **C** = CPU utilization  **UID PID PPID C**  1674 tty1 09:00:02 bash |
| **5** | $ **ps - f - u *username*** | Displays the process related only to the username specified in the command. |
| **6** | $ **pstree** | Displays the tree structure of the process. |

|  |
| --- |
| **Job control, running process in background and Process termination.** |

* **Job** is the name given to two or more process.
* **We can control the job in the following ways:**
  1. Move a job to background **(bg)**
  2. Bring the job back to foreground **(fg)**
  3. List active jobs **(jobs)**
  4. Suspend a foreground job **(ctrl+z)**
  5. Kill or terminate a job **(kill)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Suspend a foreground job **(ctrl+z)** | | | **Step1** | Execute any command | | **Step2** | Press **(ctrl+z)** | | **Step3** | $ **bg**(will display [1]+done) | | |  |  | | --- | --- | | Bring the job back to foreground **(fg)** | | | **Step1** | $ **jobs** | | **Step2** | $ **fg** | |

|  |  |
| --- | --- |
| Kill or terminate a job **(kill)** | |
| **Step1** | $ **kill** *pid* |

**Experiment:09**

|  |  |
| --- | --- |
| **Shell control structures** |  |

|  |  |
| --- | --- |
| **if, case, for, while, relational and logical operators** |  |

|  |
| --- |
| **if statement** |

**Syntax: Example: if.sh**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **if** [ expression ] **then**  statement1  **else**  statement2 **fi** | | |  | | --- | | #!/bin/bash a=10  **if** [ $a **-gt**20 ] **then**  echo “a is greater than 20”  **else**  echo “a is less than 20”  **fi** | |

**Output: a is less than 20**

|  |
| --- |
| **case statement** |

**Syntax: Example: case.sh**

|  |  |  |
| --- | --- | --- |
| **case**  $variablename**inpattern1)** command;;  **pattern2)**command;;  **pattern3)**command;;  **\*)**command;;  **esac** |  | #!/bin/bash  echo “do u agree with this?[yes or no]:”  read **ans**  **case$ans** in  [yY]) echo “agreed”;;  [nN]) echo “not agreed”;;  \*) echo “invalid”;; **esac** |
|  |

**Output:** do u agree with this?[yes or no]: y

**agreed**

|  |
| --- |
| **for statement** |

**Syntax: Example: for.sh**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **for**variablename**in**  list **do**  execute until end of list is reached **done** | | |  | | --- | | #!/bin/bash  **for** i**in** 1 2 3  **do**  echo “welcome $i times” **done** | | |
| **Output:** | | **welcome 1 times welcome 2 times welcome 3 times** |
| **while statement** | |  |

**Syntax: Example: while.sh**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **while** [ condition ] **do**  execute until condition is satisfied **done** | | |  | | --- | | #!/bin/bash  **n=1 while**  [ $n -lt 5 ] **do**  echo “Hello $n times” n = `expr $n + 1` **done** | |

**Output: Hello 1 times Hello 2 times Hello 3 times Hello 4 times**

### Relational operators Logical operators

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Operator** | **Meaning** | | -eq | = = equal too | | -ne | != not equal to | | -gt | > greater than | | -lt | < less than | | -ge | >= greater than or equal to | | -le | <= less than or equal to | | |  |  | | --- | --- | | **Operator** | **Meaning** | | ! | Logical NOT | | -a | Logical AND | | -o | Logical OR | |

**PART – B**

|  |
| --- |
| **13**. Write a shell script to display current date, time, username and directory. |

#### 13.sh

**#!/bin/bash**

**echo“**TODAYS DATE & TIME IS**:”**

**echo `**date**`**

**echo“**CURRENT USER IS**:”echo$**USER

**echo“**CURRENT WORKING DIRECTORY IS**:”**

**echo$**PWD

|  |
| --- |
| **14**. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument |

#### 14.sh

|  |
| --- |
|  OUTPUT   **Run1**  **sh** 14.sh  PLEASE ARGUMENTS TO FIND FILE  **Run2**  **sh** 14.sh 13.sh  FILE EXISTS  **Run3**  **sh** 14.sh y.txt  FILE DOES NOT EXISTS |

**#!/bin/bash**

**if[** $# **-eq** 0 **]**

**then**

**echo“**PLEASE ARGUMENTS TO FIND FILE**”**

**exit 1**

**fi**

**if[ -e** $1 **] then**

**echo“**FILE EXISTS**”**

**else**

**echo“**FILE DOES NOT EXISTS**” fi**

|  |
| --- |
| **15**. Write shell script to show various system configurations like:   1. Currently logged user name and his long name 2. Current shell 3. Your home directory |

#### 15.sh

|  |
| --- |
|  OUTPUT   **sh** 15.sh  CURRENTLY LOGGED USER NAME IS**: gptkgf**  CURRENT USER LOGIN NAME IS**: gptkgf**  CURRENT SHELL IS**:**  **/bin/bash**  CURRENT USER HOME DIRECTORY IS**:**  **/home/ gptkgf** |

**#!/bin/bash**

**echo“**CURRENTLY LOGGED USER NAME IS**:”**

**echo$**USER

**echo“**CURRENT USER LOGIN NAME IS**:”echo$**LOGNAME

**echo“**CURRENT SHELL IS**:”echo$**SHELL

**echo“**CURRENT USER HOME DIRECTORY IS**:”echo$**HOME

|  |
| --- |
| **16**. Write shell script to show various system configurations like:  **a)** Your operating system type **b)** Your current path setting  **c)** Your current working directory **d)** Show all available shells |

#### 16.sh

|  |
| --- |
|  OUTPUT   **sh** 16.sh  OPERATING SYSTEN TYPE IS**:**  **linux-gnu**  CURRENT PATH SETTING IS**:**  **/home/gptkgf/bin:/........**  CURRENT WORKING DIRECTORY IS**:**  **/home/gptkgf**  AVAILABLE SHELLS ARE**:**  **/bin/sh**  **/bin/dash**  **/bin/bash**  **/bin/rbash** |

**#!/bin/bash**

**echo“**OPERATING SYSTEN TYPE IS**:”echo$**OSTYPE

**echo“**CURRENT PATH SETTING IS**:”echo$**PATH

**echo“**CURRENT WORKING DIRECTORY IS**:”**

**echo$**PWD

**echo“**AVAILABLE SHELLS ARE**:”**

**cat** /etc/shells

|  |
| --- |
| **17**. Write a Shell script to accept any two file names and check their file permissions. |

#### 17.sh

|  |
| --- |
|  OUTPUT   **Run1**  **sh** 17.sh **u.txtv.txt**  TWO FILES HAVE **SAME** PERMISSIONS  **rw-rw-r- -**  **Run2**  **sh** 17.sh **u.txt16.sh**  TWO FILES HAVE **DIFF** PERMISSIONS **rw-rw-r- - rwxrw-r- -** |

**#!/bin/bash**

**file1=`**ls -l $1|cut -c 2-10**` file2=`**ls -l $2|cut -c 2-10**`**

**if[** $file1 = $file2 **]**

**then**

**echo“**TWO FILES HAVE SAME PERMISSIONS**” echo $**file1

**else**

**echo“**TWO FILES HAVE DIFF PERMISSIONS**”**

**echo $**file1 **echo $**file2

**fi**

|  |
| --- |
| **18**. Write a Shell script to read a file name and change the existing file permissions. |

#### 18.sh

**#!/bin/bash**

**if[** ! **-e** $1 **] then**

**echo“**PLEASE CHECK GIVEN FILE DOES NOT EXIST**” exit 1**

**fi**

**if[** ! **-x** $1 **] then**

**sudo**chmod**a**+**x “$1” && echo “**FILE IS EXECUTABLE NOW**” else**

**echo“**FILE IS ALREADY EXECUTABLE**” fi**

|  |
| --- |
|  OUTPUT   **Run1 Run2**  **sh** 18.sh **18.txt** **sh** 18.sh **18.txt**  FILE IS EXECUTABLE NOW FILE IS ALREADY EXECUTABLE |

**19.**

Write a shell script to print current month calendar and to replace the current day number by

„\*‟or „\*\*‟ respectively.

**1**

**9**

**.sh**

**#!/bin/bash**

if [ $day

–

lt

10

]

then

$pattern =

“

\*

”

else

$pattern =

“

\*\*

”

fi

cal

–

h|sed

“

s/$day/$pattern/

”









OUTPUT









**Run1**

**sh**

1

9

.sh

February

201

8

Mo Tu We Th Fr Sa Su

1

2 3

4

5

6 7 8 9 10

11

12

13 14 15 16 17

18

20 21 22 23

19

**\*\***

25

26

27 28

|  |
| --- |
| **20**. Write a C-program to fork a child process and execute the given Linux commands. |

##### 20.c

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h> #include<sys/types.h>int main(void)

**{**

**int**pid, choice=1;

**char**ucmd[10];

pid = fork();

**if**(pid>0)

**{**

**while**(choice= =1)

**{**

**printf**(“\n ENTER ANY COMMAND::”);

**scanf**(“%s”, ucmd);

**system**(ucmd);

**printf**(“\n ENTER 1 to CONTINUE and 0 to EXIT::”);

**scanf**(“%d”,&choice);

**}**

**}**

**if**(**wait(0)**= =pid)

**printf**(“\n CHILD is OVER\n”);

**}**

|  |  |
| --- | --- |
|  OUTPUT   **Run1 Run2**  **cc** 20.c **cc** 20.c  **./a.out ./a.out**  ENTER ANY COMMAND::**pwd** ENTER ANY COMMAND:: **pwd**  **/home/gptkgf /home/gptkgf**  ENTER 1 to CONTINUE and 0 to EXIT:: **1** ENTER 1 to CONTINUE and 0 to EXIT:: **0**  **CHILD is OVER**  ENTER ANY COMMAND::**date**  **Mon Feb 19 18:09:21 IST 2018**  ENTER 1 to CONTINUE and 0 to EXIT :: **0**  **CHILD is OVER** | |
| **21**. Write a C-program to fork a child process, print owner process ID and its parent process ID. | |

##### 21.c

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h> #include<sys/types.h>

int main()

**{**

**int**pid, status;

pid = fork();

**if**(pid= =0)

**{**

**printf**(“\n Hello I am CHILD \n”);

**printf**(“\n My PID = %d and PPID = %d”, **getpid()**, **getppid()** );

**}**

**else**

**{**

**wait**(&status);

**printf**(“\n Hello I am PARENT \n”);

**printf**(“\n My PID = %d and PPID = %d”, **getpid()**, **getppid()** );

**}**

**printf**(“\n Now %d terminates \n”);

**}**

|  |
| --- |
|  OUTPUT   **Run1 cc**21.c  **./a.out**    Hello I am CHILD  My PID = 2018 and PPID = 2017  Now 2018 terminates  Hello I am PARENT  My PID = 2017 and PPID = 1999  Now 2017 terminates |