

## **Practical-1**

### **AIM: Installation of Operating System.**

**Steps:** Installation of Ubuntu Operating System in Windows 11 through Oracle VM VirtualBox

#### **Step 1: Download Required Software**

Before starting the installation process, you need to download the necessary software:

1. **Oracle VM VirtualBox:** Go to the official Oracle VirtualBox website (<https://www.virtualbox.org>) and download the latest version compatible with your Windows 11 operating system.
2. **Ubuntu ISO File:** Visit the official Ubuntu website (<https://ubuntu.com>) and download the ISO file for the desired version of Ubuntu. Make sure to choose the appropriate architecture (32-bit or 64-bit) depending on your system.

#### **Step 2: Install Oracle VM VirtualBox**

**After downloading Oracle VM VirtualBox, follow these steps to install it on your Windows 11 system:**

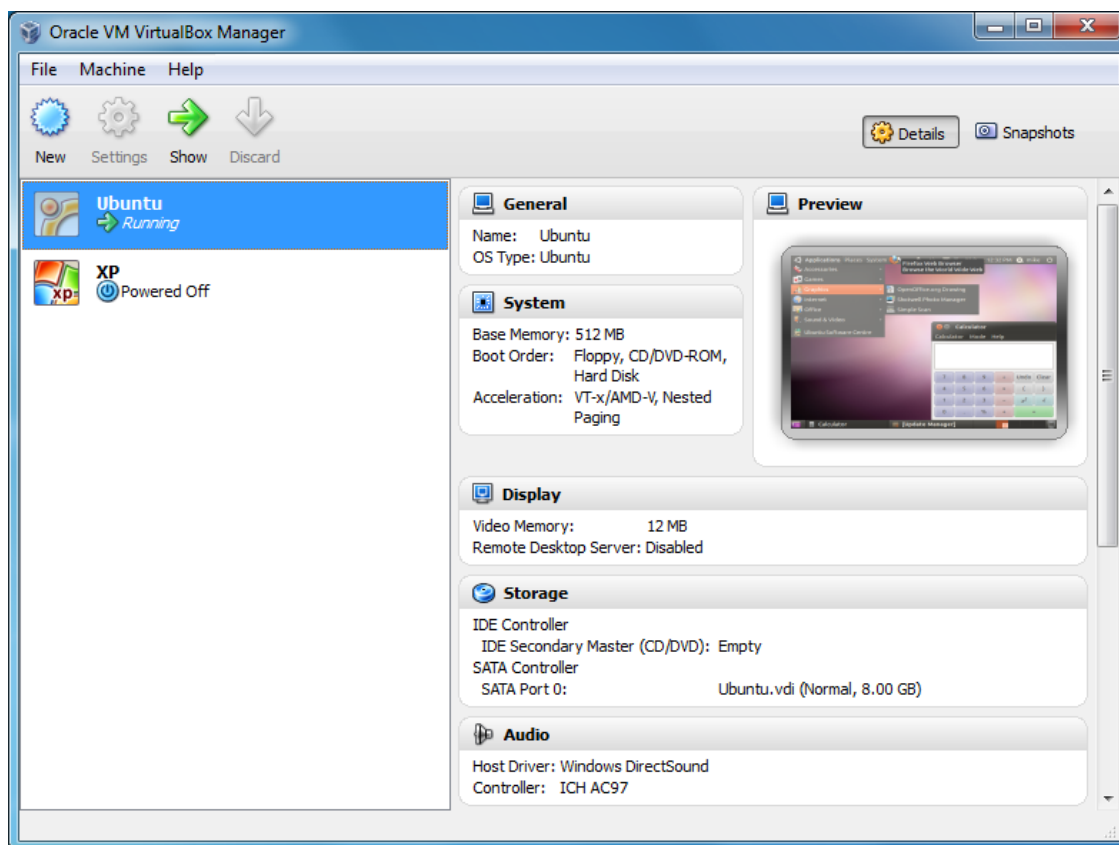
1. Locate the downloaded VirtualBox installer and double-click on it.
2. Follow the on-screen instructions provided by the installer to complete the installation process.

#### **Step 3: Create a New Virtual Machine**

Now that you have installed VirtualBox, you need to create a new virtual machine for Ubuntu:

1. Launch Oracle VM VirtualBox.
  2. Click on the "New" button in the top-left corner of the VirtualBox Manager window.
  3. In the "Create Virtual Machine" dialog box, enter a name for your virtual machine (e.g., "Ubuntu VM").
  4. Select "Linux" as the Type and choose the appropriate version (e.g., Ubuntu (64-bit)) from the Version drop-down menu.
-

5. Allocate an appropriate amount of memory (RAM) for your virtual machine. Assigning at least 2 GB of RAM to Ubuntu is recommended, but you can adjust this based on your system's resources.
6. In the "Hard Disk" section, select "Create a virtual hard disk now" and click "Create."
7. Choose the hard disk file type (VDI is recommended) and click "Next."
8. Select "Dynamically allocated" for the storage on the physical hard disk. This option allows the virtual disk to grow in size as needed.
9. Set the virtual hard disk size. It's recommended to allocate at least 20 GB of space for the Ubuntu installation, but you can adjust this based on your needs.
10. Click "Create" to finish creating the virtual machine.



#### Step 4: Configure Virtual Machine Settings

Before starting the Ubuntu installation, you need to configure a few settings for the virtual machine:

1. Select the newly created "Ubuntu VM" in the VirtualBox Manager window.
  2. Click on the "Settings" button.
-

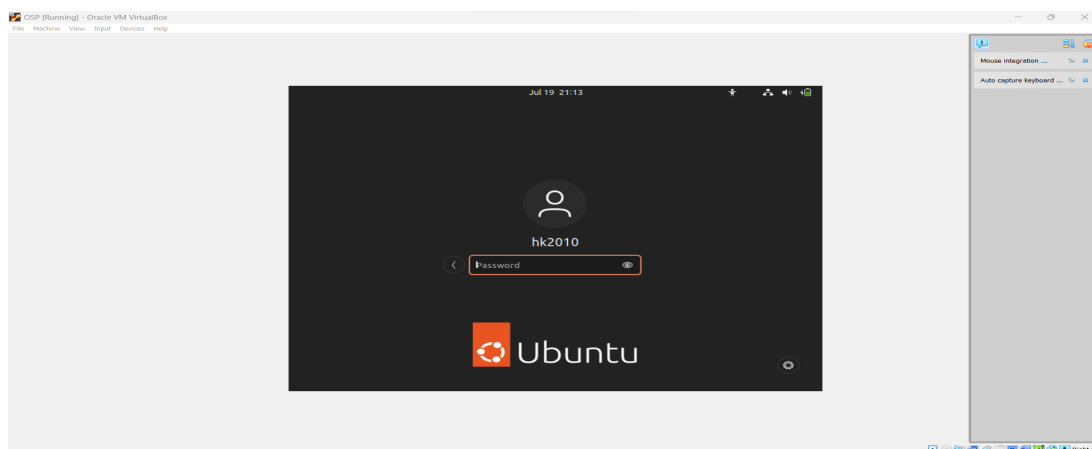
3. In the "Settings" window, go to the "Storage" tab.
4. Under "Controller: IDE," click on the empty CD/DVD icon and choose "Choose a disk file."
5. Navigate to the location where you downloaded the Ubuntu ISO file, select it, and click "Open."
6. Click "OK" to save the settings and close the "Settings" window.

### Step 5: Install Ubuntu on the Virtual Machine

Now you are ready to install Ubuntu on the virtual machine:

1. Start the virtual machine by selecting it in the VirtualBox Manager and clicking on the "Start" button.
2. The Ubuntu installation process will begin, and you will see the Ubuntu welcome screen. Select "Install Ubuntu" and follow the on-screen instructions.
3. During the installation, you'll be prompted to choose options like language, keyboard layout, and installation type. You can choose the default options for most of these settings, but make sure to select the "Install Ubuntu alongside Windows" option when asked about the installation type.
4. Create a user account and set up your preferences as prompted.
5. Once the installation is complete, the virtual machine will restart.

### Output:



### Conclusion:

---

I have successfully installed Ubuntu Operating System on your Windows 11 through Oracle VM VirtualBox. Now, you can experience Ubuntu and explore its features within the virtual environment without affecting your Windows 11 host system. Install the VirtualBox Guest Additions on the Ubuntu VM to enable better integration and performance enhancements. Enjoy using Ubuntu for your projects, testing, or learning purposes!

## **1.2 Questions and answer (given during laboratory):**

### **Q-1. List out the various ways to install OS and explain each way?**

There are several ways to install an operating system (OS) on a computer, each with its own advantages and use cases. Here are some of the common methods along with explanations for each:

#### **1. Installation from Optical Media (CD/DVD):**

This traditional method involves using a bootable installation disc, usually a CD or DVD, containing the OS setup files. To install the OS, you insert the disc into the computer's optical drive, restart the computer, and boot from the disc. The installation wizard guides you through the process, allowing you to select installation options, partition the hard drive, and configure settings. This method is becoming less common as modern computers often lack optical drives.

#### **2. USB Installation:**

With the decline of optical drives, using a bootable USB drive has become a popular alternative. You create a bootable USB drive by writing the OS's ISO file onto the USB using software like Rufus or balenaEtcher. To install the OS, you insert the bootable USB drive into the computer's USB port, restart the computer, and boot from the USB drive. The installation process is similar to using optical media.

#### **3. Network-Based Installation (PXE Boot):**

Preboot Execution Environment (PXE) allows network booting, wherein the computer boots from an OS image stored on a network server. This method is commonly used in large-scale deployments, such as in businesses or data centers, where many computers need to be provisioned with the same OS. The PXE server delivers the OS image to the client computer over the network, and the installation proceeds accordingly.

#### **4. Virtual Machine Installation:**

In this method, you can install an OS on a virtual machine using virtualization software like Oracle VM VirtualBox, VMware, or Hyper-V. Virtual machines

---

simulate a complete computer system within the host operating system. You create a virtual machine, specify its resources (CPU, RAM, storage), and then install the OS on it from an ISO file or other installation media. This approach is useful for testing new OS versions, running multiple OS environments on a single physical machine, or safely trying out software configurations.

## **5. System Image Deployment:**

A system image is a snapshot or clone of a fully configured and customized OS installation. You can create a system image of a fully set up computer and deploy it to other computers, making them identical in terms of the OS and software configuration. This method is often used in enterprise environments for mass deployment, ensuring consistency and saving time during setup.

Each method has its own advantages and is suitable for different scenarios. The choice of installation method depends on factors like the type of device, the scale of deployment, the need for customization, and the technical expertise of the user.

**Q2: From the given various ways, which way is better to use for a specific application?**

- For a single computer or a small number of computers: USB installation is typically the easiest and most convenient method.
- For large-scale enterprise deployment: Network installation or Remote Installation (PXE Boot) would be the best options.

**Q3: Which Operating system is used for various tasks/applications? And mention when to use which with its pros and cons.**

### **Windows OS:**

- Tasks: General productivity, gaming, creative work with popular software support.
- When to use: For a wide range of applications, especially in the business world and home use.
- Pros: User-friendly interface, extensive software compatibility, good gaming support.
- Cons: Can be resource-intensive, may have security vulnerabilities.

### **macOS:**

- Tasks: Creative work (graphic design, video editing, etc.), software development, general productivity.
-

- When to use: Preferred by many creatives and developers due to its integration with Apple hardware.
- Pros: Sleek and intuitive interface, optimized for Apple devices, excellent software for creative tasks.
- Cons: Limited hardware compatibility, higher-priced Apple hardware.

**Linux:**

- Tasks: Server hosting, programming, software development, data analysis, scientific computing.
- When to use: Ideal for advanced users, programmers, and server environments.
- Pros: High customizability, open-source nature, excellent for command-line tasks.
- Cons: Limited compatibility with certain proprietary software, some learning curve for beginners.

**Q4: Compare Virtual Machine VMware and Hyper-V, based on a comparison list of the specific usage of each.****VMware:**

- Specific Usage: VMware is a commercial virtualization solution suitable for both small-scale and large-scale virtualization needs.
- Advantages: Robust features, excellent performance, wide guest OS support, strong management tools.
- Disadvantages: Higher cost for the full version, steeper learning curve.

**Hyper-V:**

- Specific Usage: Hyper-V is Microsoft's virtualization platform and is typically used in Windows-based environments.
- Advantages: Integrated with Windows Server, cost-effective (included with some Windows editions), good performance.
- Disadvantages: Limited support for non-Windows OS, management tools not as extensive as VMware.

**Q5: List out the applications for each way of installation of OS.****DVD/CD Installation:**

- Used when booting from a physical disc is necessary, but it's becoming less common.

**USB Installation:**

---

- Suitable for most modern computers, especially those without an optical drive.

**Network Installation:**

- Ideal for deploying OS on multiple computers simultaneously in enterprise settings.

**Remote Installation (PXE Boot):**

- Great for large-scale OS deployment without the need for local storage media.

**Disk Imaging/Cloning:**

- Efficient for deploying a standardized OS environment across multiple machines.

**Q6: How can we convert Windows terminal to Linux terminal?**

Converting the entire Windows terminal to a Linux terminal is not possible since Windows and Linux are fundamentally different operating systems with distinct command interpreters (shells). However, you can achieve a Linux-like environment on a Windows system by using the following methods:

**Windows Subsystem for Linux (WSL):**

- WSL allows you to run a Linux distribution alongside Windows.
- It provides a compatibility layer that translates Linux system calls to Windows, allowing you to use Linux command-line tools within a Windows environment.
- To enable WSL, you need to install a Linux distribution (such as Ubuntu, Debian, or others) from the Microsoft Store.
- After installation, you can access the Linux terminal by launching the corresponding distribution from the Start menu.

**Git Bash:**

- Git Bash is a part of the Git for Windows package.
  - It provides a Bash shell environment on Windows, giving you access to many common Linux command-line tools.
  - While not a full Linux terminal, it offers a similar experience and is useful for developers familiar with Linux commands.
-

**Cygwin:**

- Cygwin is a collection of tools that provide a Linux-like environment on Windows.
- It includes a Unix-like terminal and a set of common Unix utilities.
- Cygwin allows you to compile and run many Linux software packages on a Windows system.

**Q7: Why use virtualization?**

Virtualization offers numerous benefits for both individuals and businesses, making it a popular technology in various domains. Here are some reasons why virtualization is used:

**Hardware Consolidation:**

Virtualization allows multiple virtual machines (VMs) to run on a single physical server.

By consolidating multiple workloads on a single machine, it reduces hardware costs and improves resource utilization efficiency.

**Testing and Development:**

Virtualization provides the ability to create isolated test environments, known as sandboxes.

Developers and testers can use these sandboxes to safely test new software, perform experiments, and evaluate changes without affecting the host system.

**Disaster Recovery:**

Virtual machine snapshots and backups simplify disaster recovery procedures.

Administrators can take snapshots of VMs at various stages, making it easier to revert to a previous state in case of system failures or data corruption.

**Software Isolation:**

VMs offer strong isolation between applications and operating systems running on the same physical server.

This isolation prevents conflicts and resource contention, enhancing the stability and security of each virtualized environment.

**Legacy Application Support:**

---



Virtualization allows running legacy applications that may not be compatible with the host system's operating system.

Organizations can continue using older software without the need for outdated hardware.

**Cloud Computing:**

Virtualization forms the foundation of cloud computing.

It enables cloud providers to allocate and manage resources efficiently, offering scalable and flexible services to clients.

Overall, virtualization increases flexibility, improves resource utilization, simplifies management, and enhances system reliability, making it a valuable technology for a wide range of use cases.

**Student Signature****Grade/Marks****Examiner Signature**

---

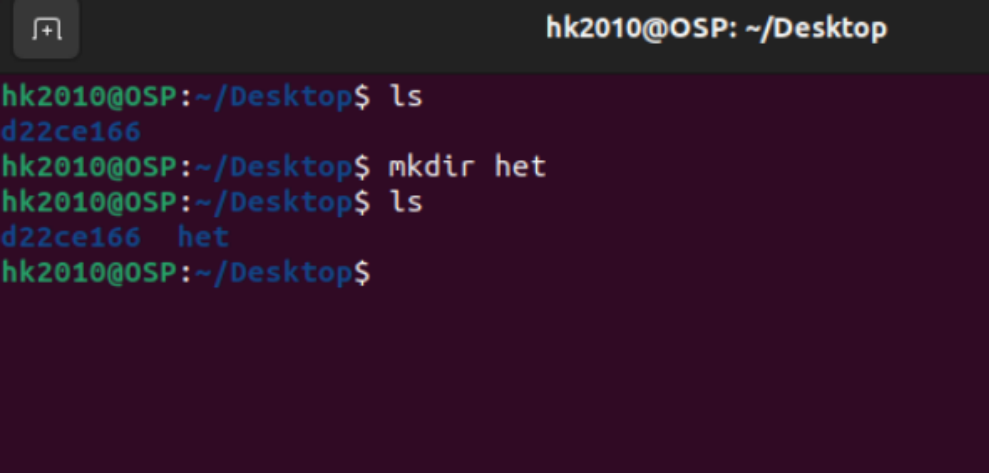
## Practical-2

### Aim: Introduction to OS and shell

#### 1. Directory Commands:

##### 1.1. mkdir

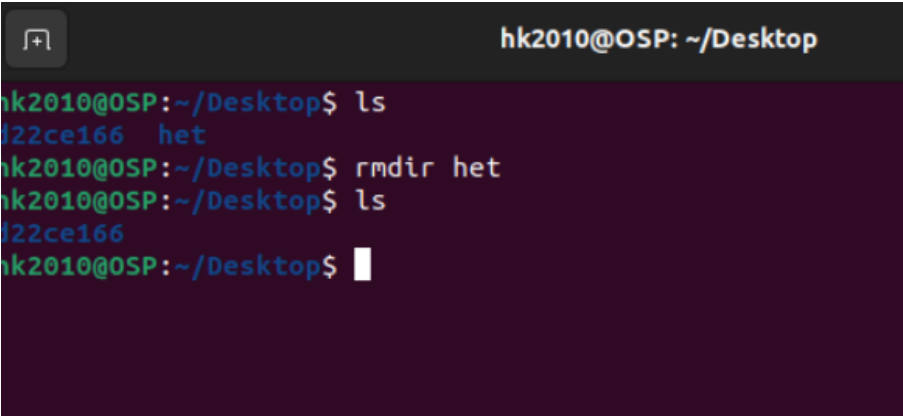
- **Command used for:** Creating a new directory.
- **Command syntax:** `mkdir directory\_name`
- **Command output:**

A terminal window titled 'hk2010@OSP: ~/Desktop' showing the execution of the 'mkdir' command. The user first runs 'ls' and sees 'd22ce166'. Then they run 'mkdir het', followed by 'ls' again, which now shows 'd22ce166' and 'het'.

```
hk2010@OSP: ~/Desktop
hk2010@OSP:~/Desktop$ ls
d22ce166
hk2010@OSP:~/Desktop$ mkdir het
hk2010@OSP:~/Desktop$ ls
d22ce166  het
hk2010@OSP:~/Desktop$
```

##### 1.2. rmdir

- **Command used for:** Removing an empty directory.
- **Command syntax:** `rmdir directory\_name`
- **Command output:**

A terminal window titled 'hk2010@OSP: ~/Desktop' showing the execution of the 'rmdir' command. The user runs 'ls' and sees 'd22ce166' and 'het'. Then they run 'rmdir het', followed by 'ls' again, which now only shows 'd22ce166'.

```
hk2010@OSP:~/Desktop$ ls
d22ce166  het
hk2010@OSP:~/Desktop$ rmdir het
hk2010@OSP:~/Desktop$ ls
d22ce166
hk2010@OSP:~/Desktop$
```

### 1.3. cd

- **Command used for:** Changing the current working directory.
- **Command syntax:** `cd directory\_name`
- **Command output:**

```
hk2010@OSP:~/Desktop$ ls
d22ce166
hk2010@OSP:~/Desktop$ cd d22ce166
hk2010@OSP:~/Desktop/d22ce166$
```

### 1.4. pwd

- **Command used for:** Printing the current working directory.
- **Command syntax:** `pwd`
- **Command output:**

```
hk2010@OSP:~/Desktop$ cd d22ce166
hk2010@OSP:~/Desktop/d22ce166$ pwd
/home/hk2010/Desktop/d22ce166
hk2010@OSP:~/Desktop/d22ce166$
```

### 1.5. ls

- **Command used for:** Listing files and directories in the current directory.
- **Command syntax:** `ls`
- **Command output:**

```
hk2010@OSP:~/Desktop$ ls
d22ce166
hk2010@OSP:~/Desktop$ cd d22ce166
hk2010@OSP:~/Desktop/d22ce166$
```

## 1.6. mv

- **Command used for:** Moving or renaming files/directories.
- **Command syntax:** `mv source destination`
- **Command output:** Moves the source file/directory to the specified destination.

```
hk2010@OSP:~/Desktop$ ls
d22ce166 sample
hk2010@OSP:~/Desktop$ mv sample.txt ~/d22ce166
hk2010@OSP:~/Desktop$ ls
d22ce166
hk2010@OSP:~/Desktop$ cd d22ce166
hk2010@OSP:~/Desktop/d22ce166$ ls
ans.txt file1 file2 sample.txt sorttext.txt
hk2010@OSP:~/Desktop/d22ce166$
```

## 2. Editor Commands:

### 2.1. vi

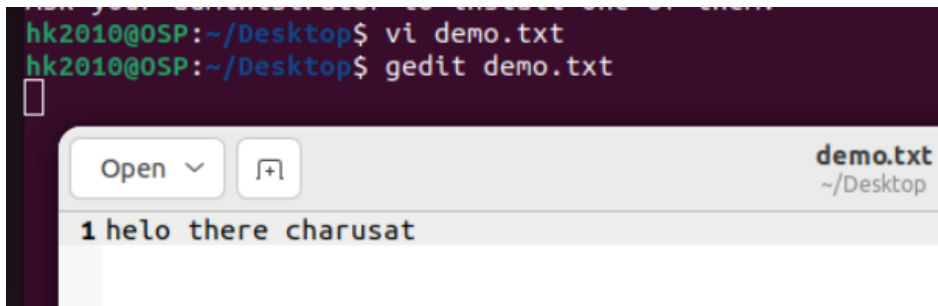
- **Command used for:** Opening the Vi text editor.
- **Command syntax:** `vi filename`
- **Command output:**

```
hk2010@OSP:~/Desktop$ vi demo.txt
hk2010@OSP:~/Desktop$
```

### 2.2. gedit

- **Command used for:** Opening the Gedit text editor.
- **Command syntax:** `gedit filename`
- **Command output:** Opens the specified file in Gedit editor

```
hk2010@OSP:~/Desktop$ vi demo.txt
hk2010@OSP:~/Desktop$ gedit demo.txt
```



The screenshot shows the Gedit text editor interface. The title bar indicates the file is 'demo.txt' located at '~/Desktop'. The editor window displays the text '1 helo there charusat' on the first line. The status bar at the bottom shows the current line and column.

### 3. File Handling/Text Commands:

#### 3.1. cp

- **Command used for:** Copying files or directories.
- **Command syntax:** `cp source destination`
- **Command output:**

```
hk2010@OSP:~/Desktop$ ls
d22ce166  demo.txt
hk2010@OSP:~/Desktop$ cp demo.txt demo1.txt
hk2010@OSP:~/Desktop$ ls
d22ce166  demo1.txt  demo.txt
hk2010@OSP:~/Desktop$
```

#### 3.2. rm

- **Command used for:** Deleting files or directories.
- **Command syntax:** `rm file\_or\_directory`
- **Command output:** Removes the specified file or directory (permanently).

```
hk2010@OSP:~/Desktop$ ls
d22ce166  demo1.txt  demo.txt
hk2010@OSP:~/Desktop$ rm demo1.txt
hk2010@OSP:~/Desktop$ ls
d22ce166  demo.txt
hk2010@OSP:~/Desktop$
```

#### 3.3. sort

- **Command used for:** Sorting lines in a text file.
- **Command syntax:** `sort filename`
- **Command output:** Displays the contents of the file with lines sorted.

```
hk2010@OSP:~/Desktop$ sort demo.txt
helo there charusat
hk2010@OSP:~/Desktop$
```

---

### 3.4. cat

- **Command used for:** Concatenating and displaying file content.
- **Command syntax:** ``cat filename``
- **Command output:** Shows the content of the file on the terminal.

```
hk2010@OSP:~/Desktop$ cat demo.txt
helo there charusat
hk2010@OSP:~/Desktop$
```

### 3.5. file

- **Command used for:** Determining file type.
- **Command syntax:** `file filename`
- **Command output:** Displays the type of the specified file.

```
hk2010@OSP:~/Desktop$ file demo.txt
demo.txt: ASCII text
hk2010@OSP:~/Desktop$
```

### 3.6. less

- **Command used for:** Viewing file content interactively (scrolling).
- **Command syntax:** ``less filename``
- **Command output:** Displays the content of the file one page at a time.

[illegible]

### 3.7. more

- **Command used for:** Viewing file content interactively (page by page).
- **Command syntax:** `more filename`
- **Command output:** Displays the content of the file one page at a time.

```
hk2010@OSP:~/Desktop$ more demo.txt
helo there charusat
hk2010@OSP:~/Desktop$
```

### 3.8. cmp

- **Command used for:** Comparing two files byte by byte.
- **Command syntax:** `cmp file1 file2`
- **Command output:** Shows the byte and line number where the first difference occurs.

```
hk2010@OSP:~/Desktop$ cmp demo.txt demo2.txt
demo.txt demo2.txt differ: byte 2, line 1
hk2010@OSP:~/Desktop$
```

### 3.9. diff

- **Command used for:** Comparing two text files line by line.
- **Command syntax:** `diff file1 file2`
- **Command output:** Displays the differences between the two files.

```
hk2010@OSP:~/Desktop$ diff demo.txt demo2.txt
1c1,2
< helo there charusat
---
> hii there
> charusat
hk2010@OSP:~/Desktop$
```

### 3.10. comm

- **Command used for:** Comparing two sorted files line by line.
- **Command syntax:** `comm file1 file2`
- **Command output:** Shows lines that are common, unique to file1, and unique to file2.

```
hk2010@OSP:~/Desktop$ comm demo.txt demo2.txt
helo there charusat
      hii there
comm: file 2 is not in sorted order
      charusat
comm: input is not in sorted order
hk2010@OSP:~/Desktop$
```

### 3.11. head

- **Command used for:** Displaying the beginning lines of a file.
- **Command syntax:** `head filename`
- **Command output:** Shows the first few lines of the file.

```
hk2010@OSP:~/Desktop$ head demo.txt
helo there charusat
hk2010@OSP:~/Desktop$ head demo2.txt
hii there
charusat
hk2010@OSP:~/Desktop$
```

### 3.12. tail

- **Command used for:** Displaying the ending lines of a file.
- **Command syntax:** `tail filename`
- **Command output:** Shows the last few lines of the file.

```
charusat
hk2010@OSP:~/Desktop$ tail demo2.txt
hii there
charusat
```

---



### 3.13. cut

- **Command used for:** Cutting sections from each line of a file.
- **Command syntax:** `cut options filename`
- **Command output:** Displays selected portions of each line from the file.

```
hk2010@OSP:~/Desktop$ vi state.txt
hk2010@OSP:~/Desktop$ cut -c -5 state.txt
Gujar
Mumba
Andra
Bihar
```

### 3.14. grep

- **Command used for:** Searching for a pattern in a file.
- **Command syntax:** `grep pattern filename`
- **Command output:** Shows lines containing the specified pattern.

```
hk2010@OSP:~/Desktop$ grep -i -n ass state.txt
4:Bihaar Assam
hk2010@OSP:~/Desktop$
```

### 3.15. touch

- **Command used for:** Creating an empty file or updating the timestamp of an existing file.
- **Command syntax:** `touch filename`
- **Command output:** Creates a new empty file or updates the timestamp of the existing file.

```
hk2010@OSP:~/Desktop$ stat demo.txt
  File: demo.txt
  Size: 20          Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d Inode: 530232       Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  hk2010)   Gid: ( 1000/  hk2010)
Access: 2023-07-21 22:03:17.158042997 +0530
Modify: 2023-07-21 22:02:23.024810140 +0530
Change: 2023-07-21 22:02:23.036811298 +0530
 Birth: 2023-07-21 22:00:22.037263522 +0530
hk2010@OSP:~/Desktop$ touch demo.txt
hk2010@OSP:~/Desktop$ stat demo.txt
  File: demo.txt
  Size: 20          Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d Inode: 530232       Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  hk2010)   Gid: ( 1000/  hk2010)
Access: 2023-07-21 22:29:07.039439850 +0530
Modify: 2023-07-21 22:29:07.039439850 +0530
Change: 2023-07-21 22:29:07.039439850 +0530
 Birth: 2023-07-21 22:00:22.037263522 +0530
hk2010@OSP:~/Desktop$
```

### 3.16. tr

- **Command used for:** Replacing characters in a file.
- **Command syntax:** `tr options set1 set2 < filename`
- **Command output:** Translates characters from set1 to set2 in the given file.

```
hk2010@OSP:~/Desktop$ cat demo.txt
helo there charusat
hk2010@OSP:~/Desktop$ cat demo.txt | tr [a-z] [A-Z]
HELO THERE CHARUSAT
hk2010@OSP:~/Desktop$
```

### 3.17. uniq

- **Command used for:** Removing duplicate lines from a sorted file.
- **Command syntax:** `uniq filename`
- **Command output:** Displays the file with adjacent duplicate lines removed.

```
hk2010@OSP:~/Desktop$ uniq demo.txt
helo there charusat
hk2010@OSP:~/Desktop$
```

## 4. User Access Commands:

### 4.1. login

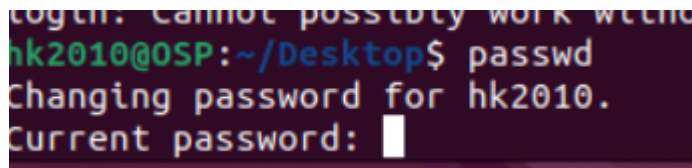
- **Command used for:** Logging into the system.
- **Command syntax:** `login`
- **Command output:** Prompts for username and password for login.

### 4.2. logout

- **Command used for:** Logging out of the system.
- **Command syntax:** `logout`
- **Command output:** Logs out the current user.

### 4.3. passwd

- **Command used for:** Changing the user password.
- **Command syntax:** `passwd`
- **Command output:** Allows the user to change their password.



```
login: cannot possibly work witho  
hk2010@OSP:~/Desktop$ passwd  
Changing password for hk2010.  
Current password: 
```

### 4.4. exit

- **Command used for:** Exiting from the current shell.
- **Command syntax:** `exit`
- **Command output:** Closes the current shell session.

## 5. Information Commands:

### 5.1. man

- **Command used for:** Displaying the manual for a command.
  - **Command syntax:** `man command\_name`
  - **Command output:** Shows the manual page for the specified command.
-

```
MKDIR(1)                                User Commands                                MKDIR(1)

NAME
    mkdir - make directories

SYNOPSIS
    mkdir [OPTION]... DIRECTORY...

DESCRIPTION
    Create the DIRECTORY(ies), if they do not already exist.

    Mandatory arguments to long options are mandatory for short options
    too.

    -m, --mode=MODE
        set file mode (as in chmod), not a=rwx - umask

    -p, --parents
        no error if existing, make parent directories as needed

    -v, --verbose
        print a message for each created directory
```

## 5.2. who

- **Command used for:** Showing a list of currently logged-in users.
- **Command syntax:** `who`
- **Command output:** Displays the list of logged-in users.

```
hk2010@OSP:~/Desktop$ man mkdir
hk2010@OSP:~/Desktop$ who
hk2010    tty2          2023-07-21 21:44 (tty2)
hk2010@OSP:~/Desktop$
```

## 5.3. date

- **Command used for:** Displaying the current date and time.
- **Command syntax:** `date`
- **Command output:** Shows the current date and time.

```
hk2010@OSP:~/Desktop$ date
Friday 21 July 2023 10:37:45 PM IST
hk2010@OSP:~/Desktop$
```

## 5.4. cal

- **Command used for:** Displaying the calendar of the current month.
  - **Command syntax:** `cal`
  - **Command output:** Shows the calendar of the current month.
-

```

hk2010@OSP:~/Desktop$ cal
      July 2023
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

```

### 5.5. tty

- **Command used for:** Displaying the file name of the terminal connected to the standard input.
- **Command syntax:** `tty`
- **Command output:** Displays the terminal file name.

```

hk2010@OSP:~/Desktop$ tty
/dev/pts/0
hk2010@OSP:~/Desktop$

```

### 5.6. calendar

- **Command used for:** Displaying a calendar for a specific month/year.
- **Command syntax:** `calendar month year`
- **Command output:** Shows the calendar for the specified month and year.

```

hk2010@OSP:~/Desktop$ cal 2024
      2024
   January  February  March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3  4  5  6      1  2  3      1  2
  7  8  9 10 11 12 13    4  5  6  7  8  9 10    3  4  5  6  7  8  9
14 15 16 17 18 19 20    11 12 13 14 15 16 17   10 11 12 13 14 15 16
21 22 23 24 25 26 27    18 19 20 21 22 23 24   17 18 19 20 21 22 23
28 29 30 31            25 26 27 28 29           24 25 26 27 28 29 30
                                     31

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

```

### 5.7. time

- **Command used for:** Measuring the time taken by a command.
- **Command syntax:** `time command`
- **Command output:** Shows the time taken by the specified command to execute.

```
hk2010@OSP:~/Desktop$ time  
  
real    0m0.000s  
user    0m0.000s  
sys     0m0.000s  
hk2010@OSP:~/Desktop$
```

### 5.8. bc

- **Command used for:** A calculator with arbitrary precision.
- **Command syntax:** `bc`
- **Command output:** Starts the interactive calculator.

```
hk2010@OSP:~/Desktop$ bc  
bc 1.07.1  
Copyright 1991-1994, 1997, 1998, 2  
Foundation, Inc.  
This is free software with ABSOLUT  
For details type `warranty'.  
56+78  
134
```

### 5.9. whoami

- **Command used for:** Displaying the current logged-in username.
- **Command syntax:** `whoami`
- **Command output:** Shows the current username.

```
hk2010@OSP:~/Desktop$ whoami  
hk2010  
hk2010@OSP:~/Desktop$
```

### 5.10. which

- **Command used for:** Displaying the location of a command.
- **Command syntax:** `which command`
- **Command output:** Shows the path of the specified command.

```
hk2010@OSP:~/Desktop$ which ls
/usr/bin/ls
hk2010@OSP:~/Desktop$
```

### 5.11. hostname

- **Command used for:** Displaying the system's hostname.
- **Command syntax:** `hostname`
- **Command output:** Shows the system's hostname.

```
hk2010@OSP:~/Desktop$ hostname
OSP
hk2010@OSP:~/Desktop$
```

### 5.12. history

- **Command used for:** Displaying the command history of the current session.
- **Command syntax:** `history`
- **Command output:** Shows a list of previously executed commands.

```
hk2010@OSP:~/Desktop$ history
1  pwd
2  cd /
3  mkdir test
4  sudo mkdir test
5  sudo user
6  sudo hk2010
7  LS -l
8  ls -l
9  cat >sorttext.txt
10 sort -t "|" -r -k 2 sorttext.txt
11 cat >sorttext.txt
12 ls
13 sort -t "|" -k 3,3 -k 2,2 sorttext.txt
14 ls
15 sort -t "|" -k 3,3 -k 2,2 sorttext.txt
16 cat sorttext.txt
17 sort -t "|" -k 3,3 -k 2,2 sorttext.txt
18 sort -t "|" -r -k 2 sorttext.txt
19 cat sorttext.txt
```

### 5.13. wc

- **Command used for:** Counting lines, words, and characters in a file.
- **Command syntax:** `wc filename`
- **Command output:** Displays the line, word, and character count of the file.

```
hk2010@OSP:~/Desktop$ wc demo.txt
1  3 20 demo.txt
hk2010@OSP:~/Desktop$
```

### 5.14. finger

- **Command used for:** Displaying user information.
- **Command syntax:** `finger username`
- **Command output:** Shows information about the specified user.

```
hk2010@OSP:~/Desktop$ finger hk2010
Login: hk2010                                Name: hk2010
Directory: /home/hk2010                      Shell: /bin/bash
On since Fri Jul 21 21:44 (IST) on tty2 from tty2
    1 hour 4 minutes idle
No mail.
No Plan.
hk2010@OSP:~/Desktop$
```

### 5.15. uname

- **Command used for:** Displaying system information.
- **Command syntax:** `uname options`
- **Command output:** Shows various system-related information.

```
hk2010@OSP:~/Desktop$ uname
Linux
hk2010@OSP:~/Desktop$
```

---



## 6. Help Commands:

### 6.1. man

- **Command used for:** Displaying the manual for a command.
- **Command syntax:** `man command\_name`
- **Command output:** Shows the manual page for the specified command.

```

MKDIR(1)                                User Commands                                MKDIR(1)

NAME
    mkdir - make directories

SYNOPSIS
    mkdir [OPTION]... DIRECTORY...

DESCRIPTION
    Create the DIRECTORY(ies), if they do not already exist.

    Mandatory arguments to long options are mandatory for short options
    too.

    -m, --mode=MODE      set file mode (as in chmod), not a=rwx - umask

    -p, --parents        no error if existing, make parent directories as needed

    -v, --verbose        print a message for each created directory
  
```

### 6.2. help

- **Command used for:** Displaying help information for shell built-ins.
- **Command syntax:** `help command`
- **Command output:** Shows help information for the specified built-in command.

```

hk2010@OSP:~/Desktop$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

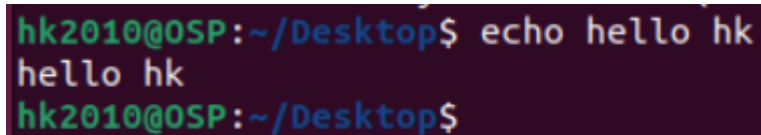
Mandatory arguments to long options are mandatory for short options too.
  -m, --mode=MODE      set file mode (as in chmod), not a=rwx - umask
  -p, --parents        no error if existing, make parent directories as needed
  -v, --verbose        print a message for each created directory
  -Z                  set SELinux security context of each created directory
                     to the default type
  --context[=CTX]     like -Z, or if CTX is specified then set the SELinux
                     or SMACK security context to CTX
  --help              display this help and exit
  --version            output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
hk2010@OSP:~/Desktop$
  
```

## 7. Terminal Commands:

### 7.1. echo

- **Command used for:** Printing text on the terminal.
- **Command syntax:** `echo message`
- **Command output:** Displays the specified message on the terminal.

A terminal window with a dark purple background. The prompt is 'hk2010@OSP:~/Desktop\$'. The command 'echo hello hk' has been entered and executed. The output 'hello hk' is displayed on the next line. The prompt 'hk2010@OSP:~/Desktop\$' is shown again on the third line.

```
hk2010@OSP:~/Desktop$ echo hello hk
hello hk
hk2010@OSP:~/Desktop$
```

### 7.2. clear

- **Command used for:** Clearing the terminal screen.
- **Command syntax:** `clear`
- **Command output:** Clears the terminal screen.

Student Signature

Grade/Marks

Examiner Signature

---