Experiment 1:

Identify the peripherals of a computer, components in a CPU and its functions. Draw the block diagram of the CPU along with the configuration of each peripheral

i) Disassemble and assemble the PC back to working condition.

ii) Install MS windows on the personal computer.

iii) Install Linux on the computer. This computer should have windows installed. The system should be configured as dual boot with both windows and Linux

PERIPHERALS OF A COMPUTER (INPUT/OUTPUT DEVICES):

A **computer peripheral** is any external device that provides input and output for the **computer**.

For example, a keyboard and mouse are input **peripherals**, while a monitor and printer are output **peripherals**. ... Some **peripherals**, such as external hard drives, provide both input and output for the **computer**.

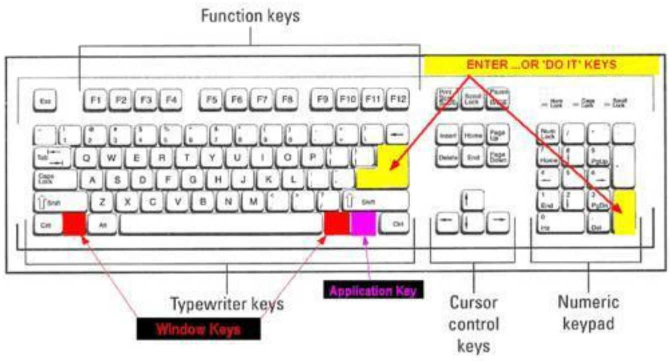
INPUT DEVICES:

- Converts data to binary form acceptable by the computer
- Sends data in binary form to the computer for further processing
- Accepts data from outside world

Some Standard Input Devices are : Keyboard Mouse Joystick and Trackball Scanner

Keyboard:

A keyboard is the most common input device. It is used to input data manually by typing. The computer keyboard is like a typewriter keyboard with some extra special keys called **function keys** and **control keys** which can be programmed by the user according to his needs.



Mouse :

Mouse is basically a pointing device. This is also an input device but instead of sending characters it send the coordinates of the point on the screen on which the associated cursor is placed. A mouse is useful for executing GUI based softwares.

Joystick And Tracker ball :

Both of these are again pointing devices and are used for the same purpose as a mouse**.**

Scanner :

The scanner captures the entire information of an image directly from the source (generally a page) stores it in graphic format for displaying back on the screen.

Touch Pads A device that lays on the desktop and responds to pressure.

Bar Code Readers

Bar coded data is generally used in labelling goods, numbering the books etc. Bar Code Reader scans a bar code image, converts it into an alphanumeric value which is then fed to the computer to which bar code reader is connected.

Optical Mark Reader(OMR)

OMR is a special type of optical scanner used to recognize the type of mark made by pen or pencil. It is used where one out of a few alternatives is to be selected and marked. It is specially used for checking the answer sheets of examinations having multiple choice questions.

OUTPUT DEVICES:

- Accepts data in binary form from the
- computer Converts the coded data to
- human acceptable form Outputs the
- converted result to the outside world

Some standard output devices are : Video display unit , Printers, Plotters

Video display unit

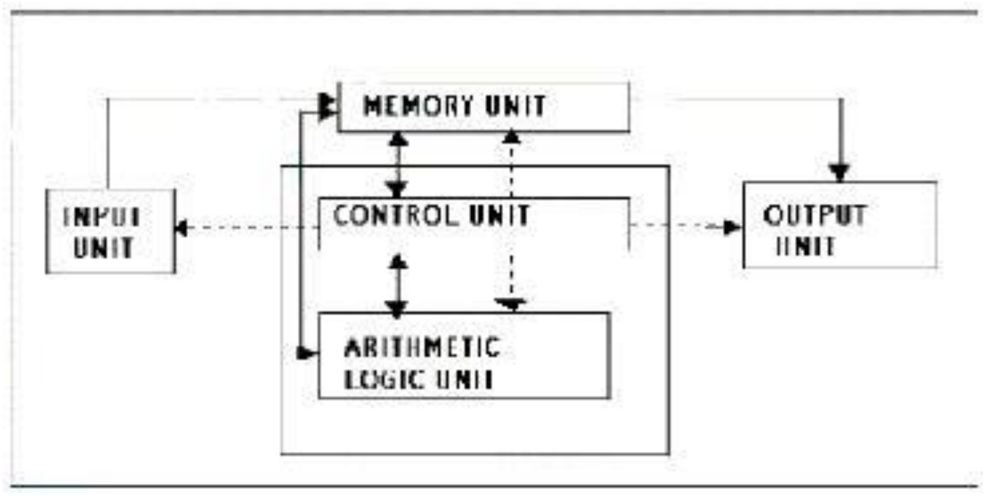
A TV like screen connected to the computer. It is also Called as raster scan device

Printers

Types of printers

- Character printer: Prints one character at a time
- Line printer : Prints one line of text at a time
- Page printer :Prints a complete page at a time

COMPONENTS OF CPU AND ITS FUNCTIONS



EVERY COMPUTER HAS THREE MAJOR COMPONENTS. THEY ARE:

- CPU
- INPUT UNIT
- OUTPUT UNIT.

(1) **CENTRAL PROCESSING UNIT(CPU):**

- The Central Processing Unit is also called as the brain of the computer. The CPU controls the execution of programs and performs the calculations. It stores data, results and program. It controls the operation of all parts of computer.

CPU Consist of following parts and Its Functions:

a) Memory Unit.

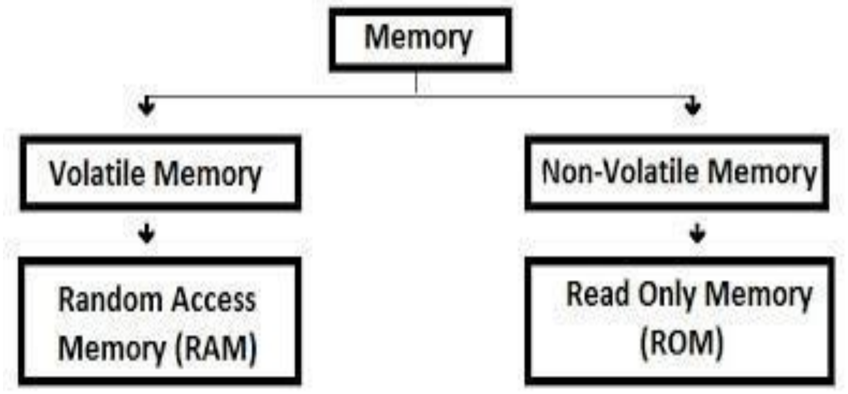
b) Arithmetic and Logic Unit

c) Control Unit

a)Memory Unit.

Memory is an essential element of a computer. Without its memory, a computer is of hardly any use. Memory plays an important role in saving and retrieving data. The performance of the computer system depends upon the size of the memory. Memory is of following types:

1. **Primary Memory / Volatile Memory.**
2. **Secondary Memory / Non Volatile Memory.**



1. **<u>Primary Memory / Volatile Memory</u>:** Primary Memory is internal memory of the computer. RAM AND ROM both form part of primary memory. The primary memory provides main working space to the computer. The following terms comes under primary memory of a computer are discussed below:

- **Random Access Memory (RAM):** RAM is the internal memory of the CPU for storing data, program and program result. It is read/write memory which stores data until the machine is working. As soon as the machine is switched off, data is erased.RAM is volatile, i.e. data stored in it is lost when we switch off the computer or if there is a power failure. Hence a backup uninterruptible power system(UPS) is often used with computers. RAM is small, both in terms of its physical size and in the amount of data it can hold.
- **Read Only Memory (ROM):** ROM stands for Read Only Memory. The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture. A ROM, stores such instructions that are required to start a computer. This operation is referred to as bootstrap. ROM chips are not only used in the computer but also in other electronic items like washing machine and microwave oven.

1. <u>Secondary Memory / Non-Volatile Memory</u>: Secondary memory is external and permanent in nature. The secondary memory is concerned with magnetic memory. Secondary memory can be stored on storage media like floppy disks, magnetic disks, magnetic tapes, This memory can also be stored optically on Optical disks - CD-ROM. The following terms comes under secondary memory of a computer are discussed below:

b) Arithmetic and Logic Unit

The ALU is responsible for performing all logical and arithmetic operations Some of the arithmetic operations are as follows: addition, subtraction, multiplication and division.-Some of the logical operations are as follows: comparison between numbers, letter and or special characters.- The ALU is also responsible for the following conditions: Equal-to conditions, Less-than condition and greater than condition.

c) Control Unit

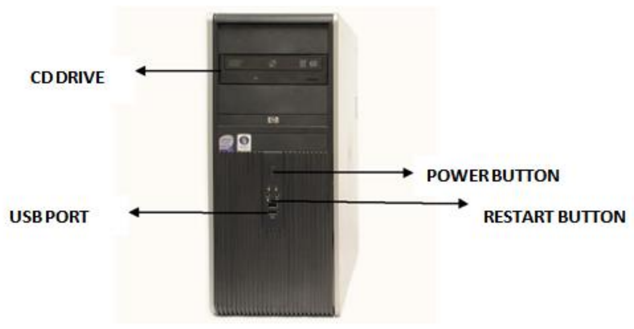
It controls all other units in the computer. The control unit instructs the input unit, where to store the data after receiving it from the user. It controls the flow of data and instructions from the storage unit to ALU. It also controls the flow of results from the ALU to the storage unit. The control unit is generally referred as the central nervous system of the computer that control and synchronizes its working.

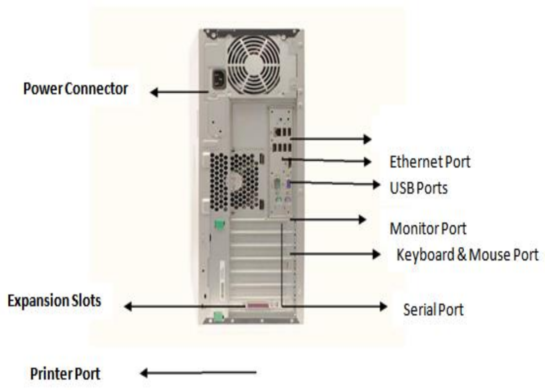
I)DEASSEMBLING AND ASSEMBLING THE PC

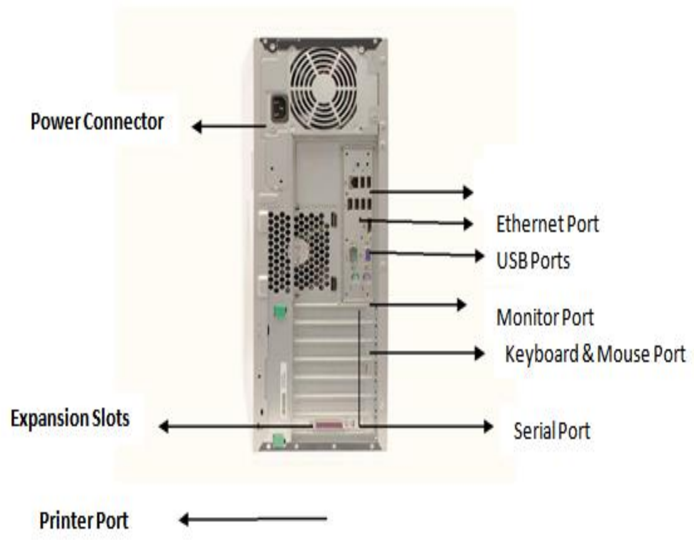
Before going to Disassembling and Assembling we have to discuss about few components .they are: Cabinet ,Mother Board, Processor, SMPS,RAM,HDD.

1. CABINET:

It is skin for every computer. It is used to protect precious components like mother board, processor and soon from dust, heat and moisture.



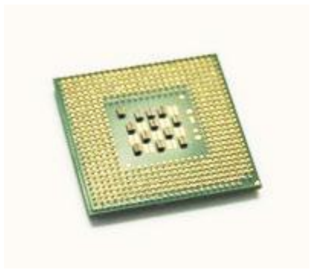



Power Connector
Ethernet Port
USB Ports
Monitor Port
Keyboard & Mouse Port
Expansion Slots
Serial Port
Printer Port

.MOTHERBOARD

The **motherboard** is the computer's **main circuit board**. It's a thin plate that holds the CPU, memory, connectors for the hard drive and optical drives, expansion cards to control the video and audio, and connections to your computer's ports (such as USB ports). The motherboard connects directly or indirectly to every part of the computer.

4.CPU/PROCESSOR



The central processing unit (CPU), also called a **processor**, is located inside the **computer case** on the motherboard. The CPU is generally a **two-inch ceramic square** with a **silicon**

chip located inside. The chip is usually about the size of a thumbnail. The CPU fits into the motherboard's **CPU socket**, which is covered by the **heat sink**, an object that absorbs heat from the CPU.

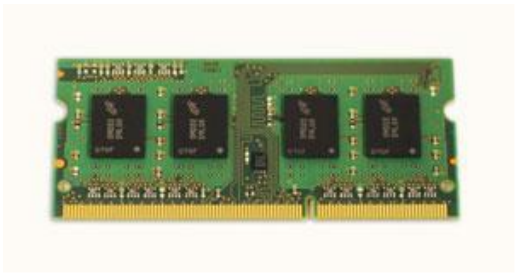
A processor's **speed** is measured in **megahertz (MHz)**, or millions of instructions per second; and **gigahertz (GHz)**, or billions of instructions per second. A faster processor can execute instructions more quickly. There are many processor

manufacturers for personal computers, but the most well-known ones are **Intel** and **AMD**.

5.POWER SUPPLY UNIT

The power supply unit in a computer **converts the power** from the wall outlet to the type of power needed by the computer. It sends power through cables to the motherboard and other components.

6.RAM (RANDOM ACCESS MEMORY)



RAM is your system's **short-term memory**. Whenever your computer performs calculations, it temporarily stores the data in the RAM until it is needed.

7.HARD DRIVE

The **hard drive** on your computer is where the software is installed, and it's also where your documents and other files are stored. The hard drive is **long-term**-**storage**, which means the data is still saved even if you turn the computer off or unplug it.When you run a program or open a file, the computer copies some of the data from the**hard drive** onto the **RAM**. When

you **save** a file, the data is copied back to the **hard drive**. The faster the hard drive is, the faster your computer can **start up** and **load programs**.

<u>STEPS FOR DISASSEMBLING:</u>

Step 1: Unplugging

The first thing you do, is unplug every cable that's plugged in to your computer. That includes the following cables:

- Power
- USB
- Mouse
- Keyboard
- Internet
- Ethernet
- Modem

Step 2: Outer Shell/Casing

First off, unscrew the four screws on the back of the computer. On most computer cases, there will be large knobs that you can unscrew by hand or by screw driver on the back-right side of the computer. The left side has small screws because on that side you can't access much on the inside.

Once the screws are removed, you can remove the **side panels**. On most computers, they just slide off. Start with the left side panel (the side that once had the knobs), slide it towards the back of the computer. Now you can remove the left panel. Just like the other one, slide it towards the back of the computer.

Step 3: Outer Shell/Casing (cont.)

Just like the side panels, the **top panel** slides off. Also like the side panels, the top one slides toward the back of the computer. The front panel clips on to the metal frame with four tabs, so you must push them in and slide the whole panel forward.

Step 4 : System Fan

Most computers have two fans: the system fan, the one blowing air into the computer, and the CPU fan, the one blowing air onto the CPU heat sink. I will start by removing the system fan first. It is located at the back side of the computer, the side with all the component plugins. First, unplug the fan from the motherboard. You can find the plug by following the wire from the fan. It should be labeled "SYS_FAN1". Next, you will have to unscrew the fan from the outside. You should now be able to lift the fan out of the PC.

Step 5: CPU Fan

The CPU fan is located right on top of the CPU heat sink, which is a large piece of metal with fins on the top. The CPU fan plugs into the motherboard in an awkward place, that is hard to access. But just follow the wires and you should easily find it. It is labelled "CPU FAN1". To remove the fan from the heat sink, remove the four screws securing it in place.

Step 6: Power Supply(SMPS)

The power supply is a large metal box located at the upper-back part of the computer. They sometimes come with an on/off switch that is accessible from the back of the computer. The main power cord also plugs into the back of the power supply.

The power supply supplies power to every component in a computer, therefore it has the most wires out of every other component in the computer. The first thing I will do is unplug every wire coming from the power supply. The list below is every thing that I had to disconnect:

- Motherboard (very large connector/plug)
- CD/DVD drive[s] power
- Internal hard drive power
- Portable hard drive slot power

Once everything is unplugged, unscrew the four screws holding the power supply in place, on the back of the computer. Next, push the power supply from the outside, then lift it out.

Step 7: CD/DVD Drive[s]

The CD/DVD drive is one of the easiest components to remove. First, unplug the ribbon from the back of the drive. Once that is completed, pull on the tab securing the drive in place, then push it out from the inside..

Step 8: Hard Disk Drive

First off, de-attach the connector at the back of the slot, and unplug the other end from the motherboard. Also unplug the SATA cable from the motherboard and the hard drive. The portable hard drive slot is secured the same way the CD/DVD drive is, with a tab. Pull on the tab, then slide the slot out.

To remove the hard drive from the side of the slot, unscrew the four screws securing it in place. You must be very careful to not drop the hard drive, as it is very delicate!

Step 9: Expansion Cards

Expansion cards give computer new capabilities, once installed. Different examples are:

- Modem.
- Network Card.
- Sound Card.
- Video Card.

There should be a single screw on top of each expansion card slot, whether it's occupied, or empty. Remove the screws on the occupied card slots. Once the screws are removed, you should be able to remove the cards by pulling them carefully upward.

Step 10: Connectivity Centre Cables (front and back panel)

The connectivity centre is the area on the front and back of the computer where there is many input sections, like usb , microphone, headphones, video, etc.. I remove the whole connectivity centre in this step, and I will unplug all the cables coming from it.

Do that (unplug all cables), then unplug the wires leading from the power button, hdd light, and power light.

Step 11: RAM (Random Access Memory)

So pretty much, the more RAM you have, the faster your computer runs. Most computers have 4 RAM slots, and two RAM chips. My computer came stock with two, but yours might have more or less. To remove the RAM, push down on both tabs holding the RAM in place, which are located at both ends of the RAM. Please see the pictures.

Step 12: Power Button ; Power LED + HDD LED

There is a zip tie holding the wires/cables for the front connectivity center and front power button/LEDs. Cut it.To remove the chasis, press in on the tabs that are located on the chasis' side. Refer to the pictures to see the tabs. Once the tabs are being pressed in, pull the whole chasis out of the computer.

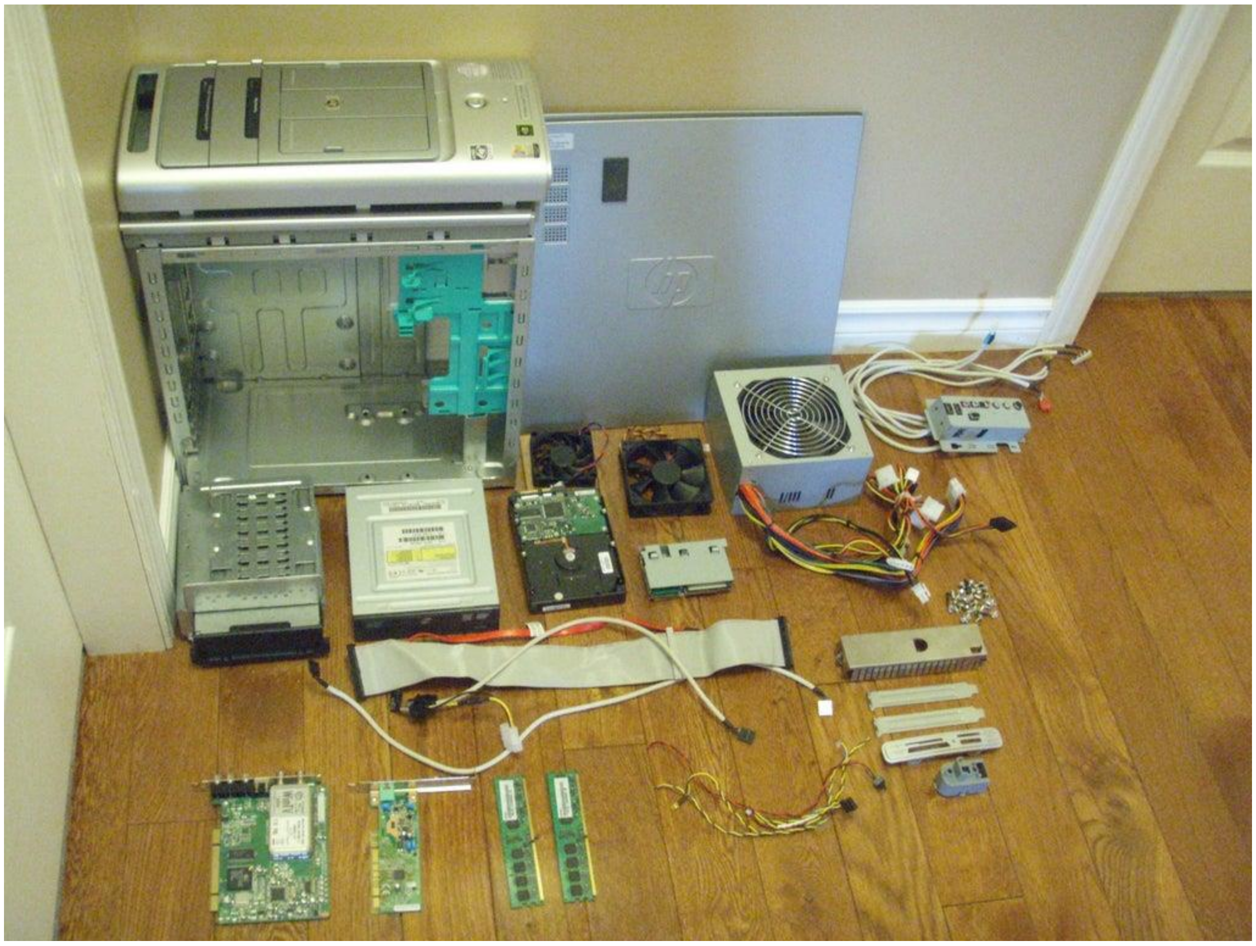
To remove the LEDs from the "chasis", push them from the front with a screw driver. To remove the button, you will need to push it from the back, the side with the wires. For clarification, see the pictures.

Step 13: Motherboard

The motherboard links every component in the computer together. The CPU, RAM, and expansion cards are attached directly to it, and every other part of the computer is in one way or another attached to it.

The motherboard has seven screws holding it to the frame, which are indicated by large white circles around them. Remove those seven, then lift the motherboard out of the frame.

Step 14: Done!



Case:

- Metal Right Panel
- Metal Left Panel
- Plastic Front Panel
- Plastic Top Panel
- Metal & Plastic Frame

Hardware:

- Hard Drive
- CD/DVD Drive
- Power Supply
- Expansion Cards
- RAM Chips
- Connectivity Center

Cables\Wires:

- SATA Cable (Hard Drive to Motherboard)
- Portable HDD Dock and Wires (Power and Data from Portable HDD to Motherboard)
- Drive Ribbon (CD/DVD Drive to Motherboard

Miscellaneous:

- 33 Screws
- Drive Slot Cover
- 2 Expansion Card Slot Covers
- Plastic Piece (Case referred to it as a "chassis" for a button and LEDs)
- Large System Fan
- Small CPU Fan
- Portable HDD Slot

STEPS FOR ASSEMBLING

- Fix the SMPS on the cabinet of PC using the screws provided.
- Fix the motherboard on the cabinet of PC using the screws provided.
- Connect the power cables from SMPS to motherboard.
- Insert the pre-processor into the slot provided such that the corner with no pin coincide with corner without pinhole on motherboard.
- Apply the appropriate adhesive on the processor for fixing the processor fan.
- Fix the processor fan on the processor and use clips on it to keep it firm.
- Connect the power cable to the processor fan
- Insert the RAM card into the slots provided on the motherboard.

- Set the jumpers setting on the hard disc drive.
- Fix the hard disc drive in the space provided in the PC cabinet using screws provided.
- Fix the HDD (2) in the space provided in the PC cabinet using screws provided.
- Fix the CD-ROM in the space provided in the PC cabinet using screws provided.

- Connect the HDD, CD-ROM drive to motherboard using flat ribbon.
- Connect power supply to the HDD, CD-ROM drive using the cables from the SMPS.
- Connect wires of speakers and lights of cabinet to the motherboard.
- Connect the network interface and other cards to motherboard by inserting in right slots and fix them in cabinet using the screws provided.
- Place the cabinet in right position.
- Fix the doors of the cabinet.
- Connect the data cable of monitor to the CPU.
- Connect the keyboard cable to the CPU.
- Connect the mouse cable to the CPU.
- Connect other devices to CPU.
- Connect the LAN cable to NIC in CPU.
- Connect the power supply to CPU.
- Connect the power supply to Monitor.
- Switch on the computer after giving the power supply

II)WINDOWS INSTALLATION

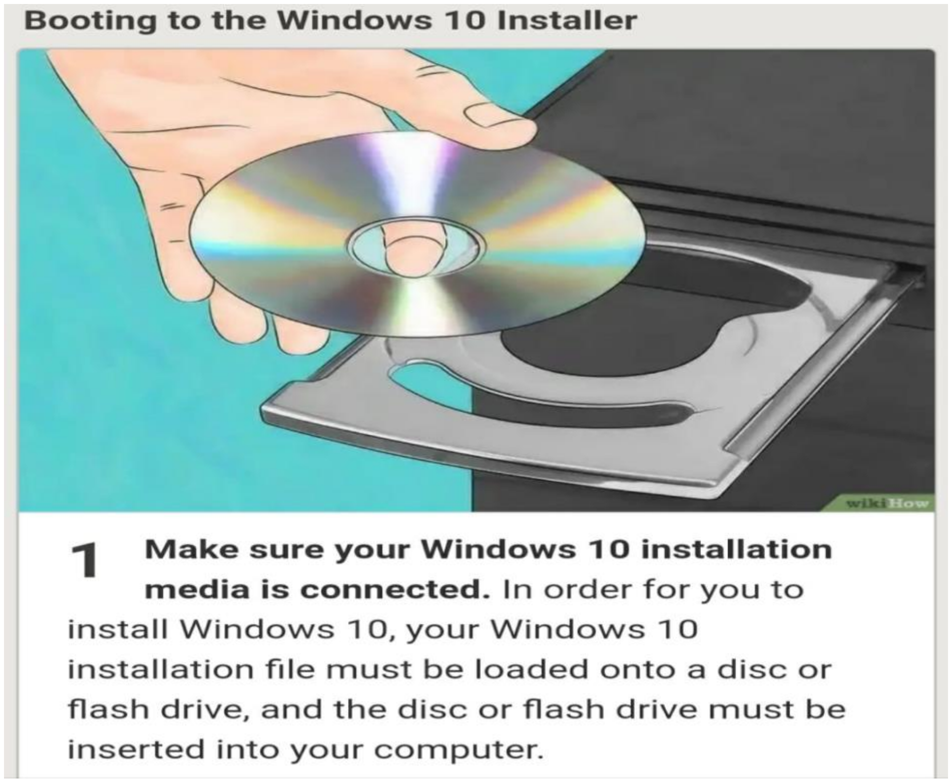
Windows 10 (also known as **Windows NT** family of operating system) is the latest desktop version of the Microsoft Windows operating system. It was made publicly available on july 15, 2015. Two editions of Windows 10 are most commonly available: **Windows 10 Home Edition** which is targeted at home users and **Windows 10 Professional** which has additional features such as dual-processor support and the ability to join a domain, a grouping of centrally managed Windows computers.

Installation process:

Installing Windows 10 can take up to 1 hour 30 minutes. To make the process more manageable, it has been broken up into several sections. When you are ready, install Windows 10.

Begin the installation

Step 1: Insert the Windows 10 CD or pen drive into your computer and restart your computer.



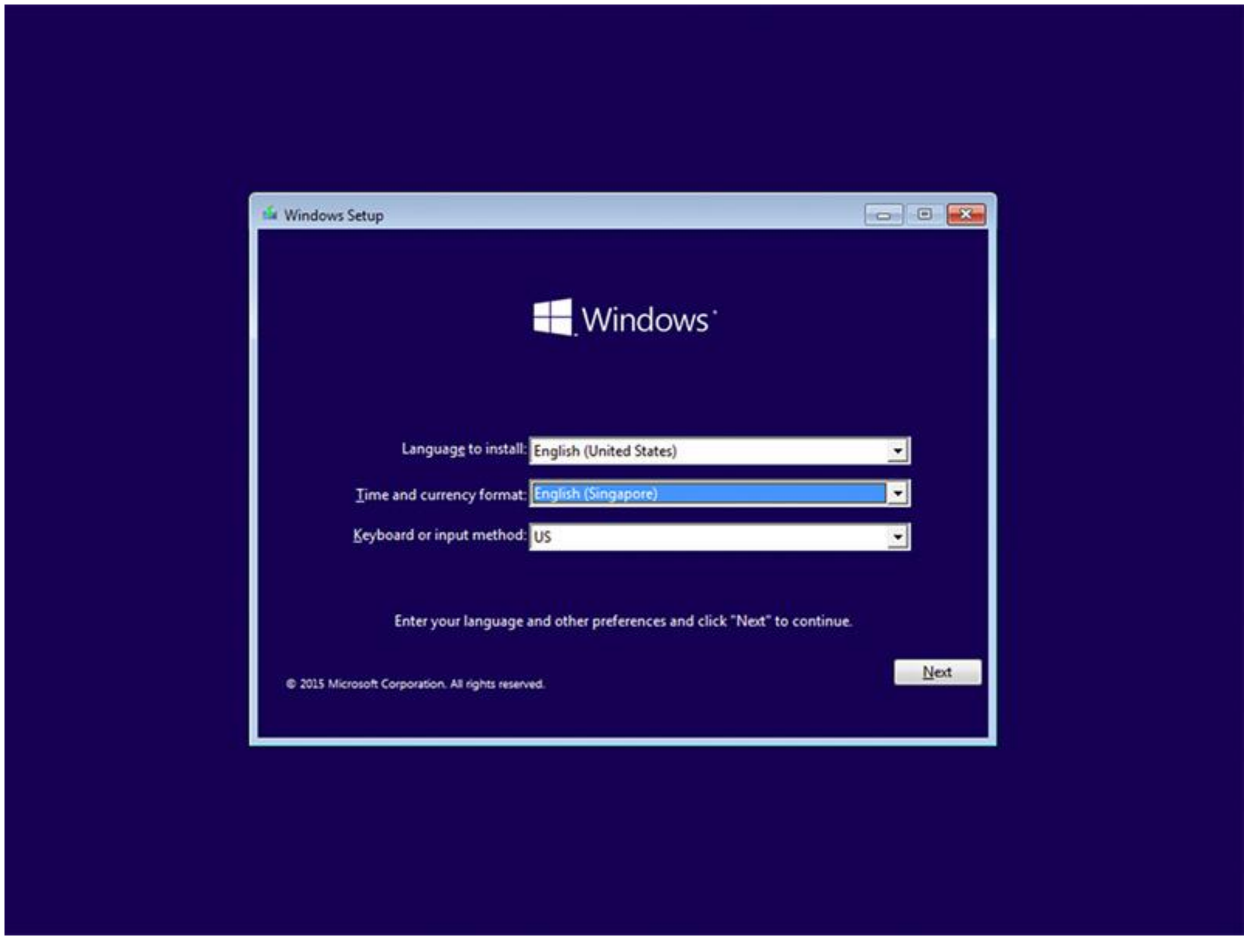
Booting to the Windows 10 Installer

1 **Make sure your Windows 10 installation media is connected.** In order for you to install Windows 10, your Windows 10 installation file must be loaded onto a disc or flash drive, and the disc or flash drive must be inserted into your computer.

Click Restart. It's in the pop-up menu above the power icon. Doing so will restart your computer.

Press and hold Del or F2 to enter setup.

Step 2. This is the first screen you will see if you install Windows 10 using a bootable USB flash drive or DVD. Here's where you choose the OS' language, time and currency format, and input method.

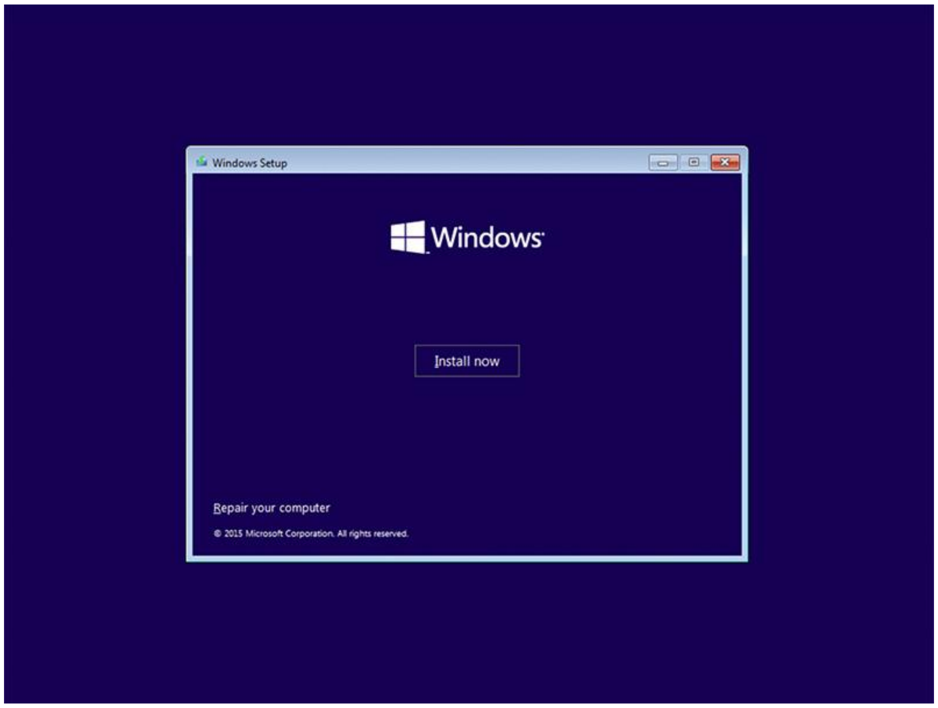


Language to install: English (United States)
Time and currency format: English (Singapore)
Keyboard or input method: US

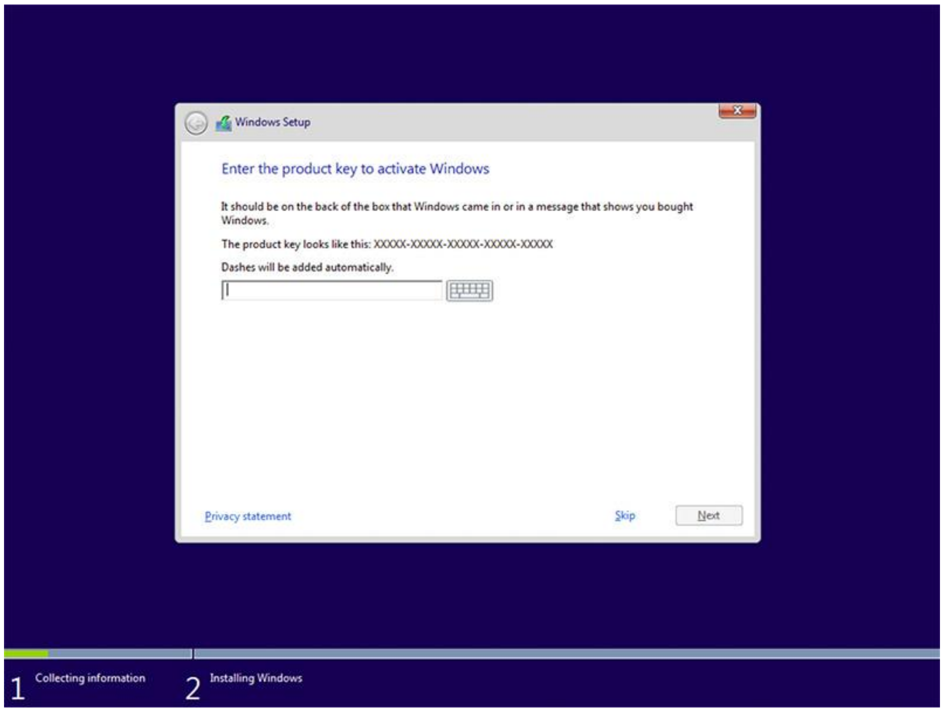
Enter your language and other preferences and click "Next" to continue.

© 2015 Microsoft Corporation. All rights reserved.

Step 3: You will get the below screen



Step 4: We have to Enter The Product Key



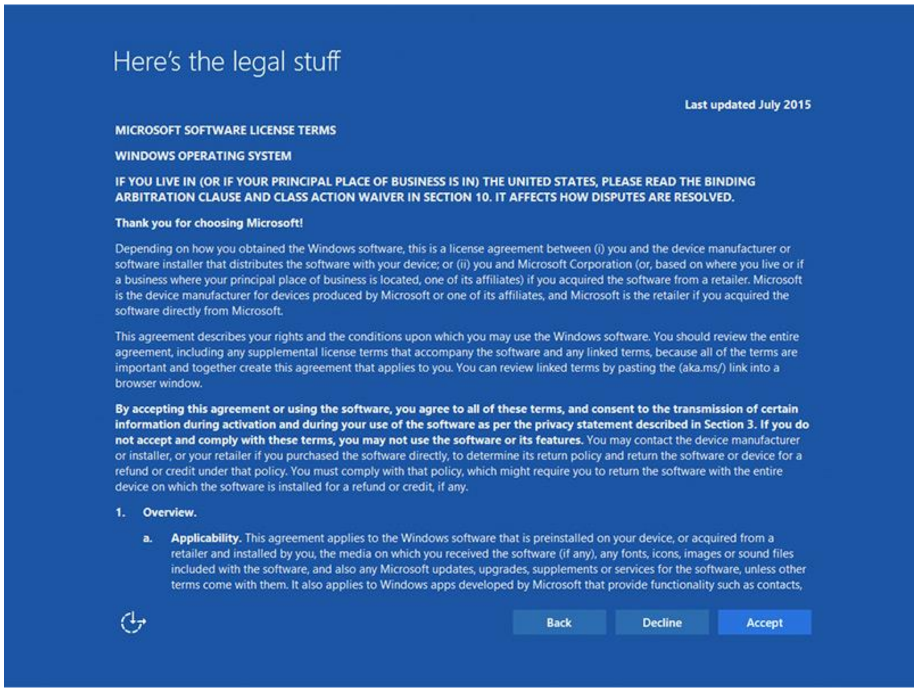
Enter the product key to activate Windows

It should be on the back of the box that Windows came in or in a message that shows you bought Windows.

The product key looks like this: XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

Dashes will be added automatically.

Step 5: Accept the license terms. After you read them.


Here's the legal stuff

Last updated July 2015

MICROSOFT SOFTWARE LICENSE TERMS

WINDOWS OPERATING SYSTEM

IF YOU LIVE IN (OR IF YOUR PRINCIPAL PLACE OF BUSINESS IS IN) THE UNITED STATES, PLEASE READ THE BINDING ARBITRATION CLAUSE AND CLASS ACTION WAIVER IN SECTION 10. IT AFFECTS HOW DISPUTES ARE RESOLVED.

Thank you for choosing Microsoft!

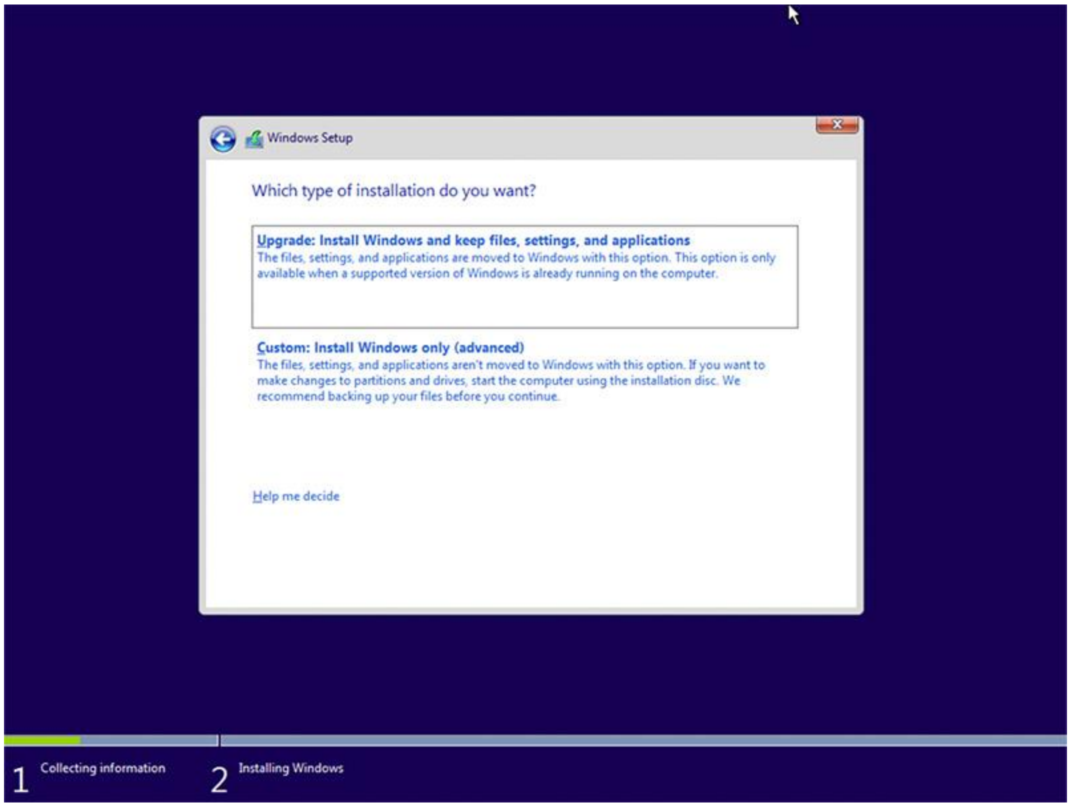
Depending on how you obtained the Windows software, this is a license agreement between (i) you and the device manufacturer or software installer that distributes the software with your device; or (ii) you and Microsoft Corporation (or, based on where you live or if a business where your principal place of business is located, one of its affiliates) if you acquired the software from a retailer. Microsoft is the device manufacturer for devices produced by Microsoft or one of its affiliates, and Microsoft is the retailer if you acquired the software directly from Microsoft.

This agreement describes your rights and the conditions upon which you may use the Windows software. You should review the entire agreement, including any supplemental license terms that accompany the software and any linked terms, because all of the terms are important and together create this agreement that applies to you. You can review linked terms by pasting the (aka.ms/) link into a browser window.

By accepting this agreement or using the software, you agree to all of these terms, and consent to the transmission of certain information during activation and during your use of the software as per the privacy statement described in Section 3. If you do not accept and comply with these terms, you may not use the software or its features. You may contact the device manufacturer or installer, or your retailer if you purchased the software directly, to determine its return policy and return the software or device for a refund or credit under that policy. You must comply with that policy, which might require you to return the software with the entire device on which the software is installed for a refund or credit, if any.

1. **Overview.**
   a. **Applicability.** This agreement applies to the Windows software that is preinstalled on your device, or acquired from a retailer and installed by you, the media on which you received the software (if any), any fonts, icons, images or sound files included with the software, and also any Microsoft updates, upgrades, supplements or services for the software, unless other terms come with them. It also applies to Windows apps developed by Microsoft that provide functionality such as contacts,

Step 6: Next, you'll be presented with this screen, where you can choose to do either an upgrade (files, settings, and apps are moved to Windows) or a custom install (files, settings, and apps aren't moved). The latter is the one to choose if you prefer a clean install, which was what we did.

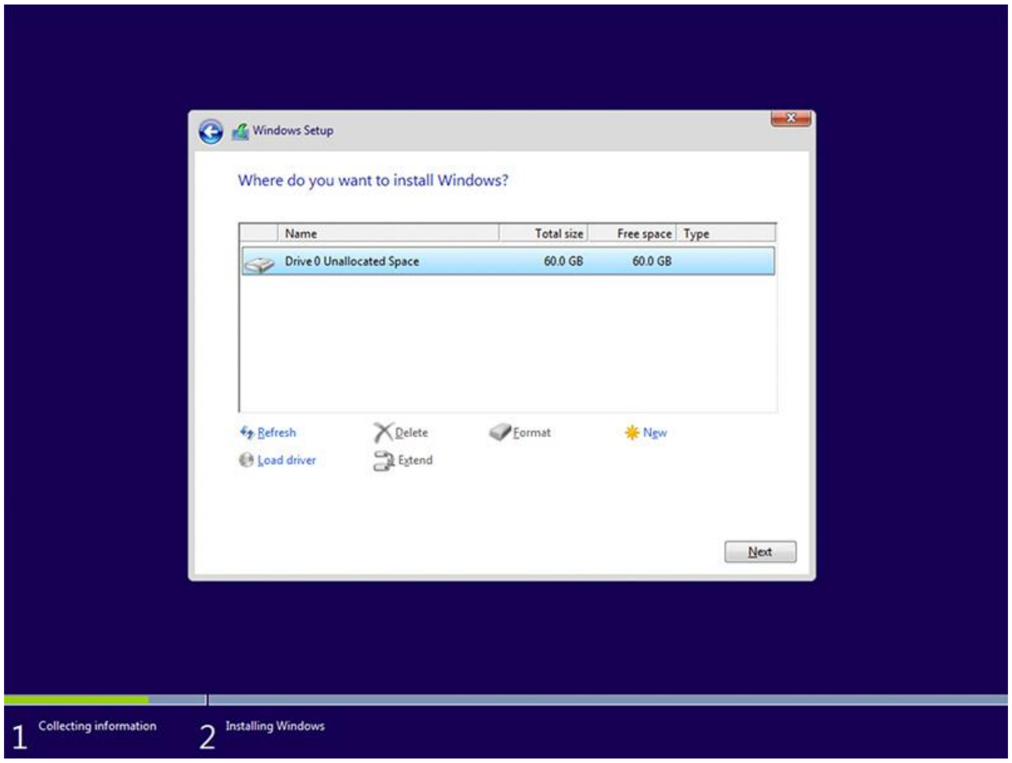


Which type of installation do you want?

Upgrade: Install Windows and keep files, settings, and applications
The files, settings, and applications are moved to Windows with this option. This option is only available when a supported version of Windows is already running on the computer.

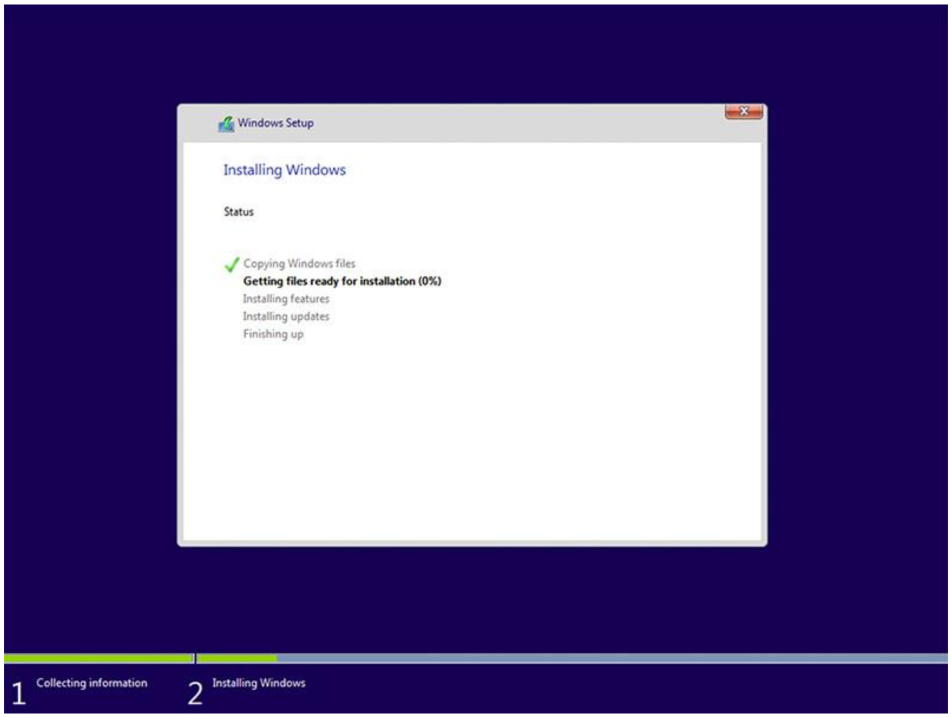
Custom: Install Windows only (advanced)
The files, settings, and applications aren't moved to Windows with this option. If you want to make changes to partitions and drives, start the computer using the installation disc. We recommend backing up your files before you continue.

Step 7: Here's where you select the drive to install Windows 10 on. You can format a drive here as well and also Partitions of Hard Drive



Name	Total size	Free space	Type
Drive 0 Unallocated Space	60.0 GB	60.0 GB	

Step 8: Now, you wait.



Installing Windows

Status

✓ Copying Windows files

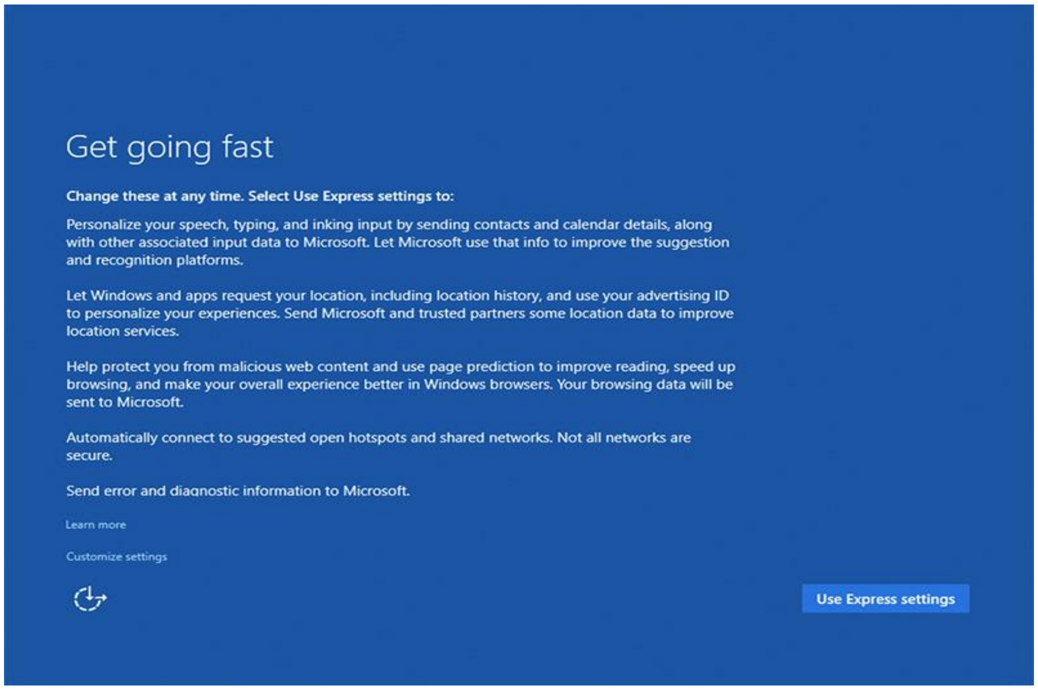
Getting files ready for installation (0%)

Installing features

Installing updates

Finishing up

Step 9: Choose ‘Customize settings’ to customize them. notice that little icon at the bottom left? Click on it to access an accessibility menu where you can turn on things like increase the screen contrast.



Get going fast

Change these at any time. Select Use Express settings to:

Personalize your speech, typing, and inking input by sending contacts and calendar details, along with other associated input data to Microsoft. Let Microsoft use that info to improve the suggestion and recognition platforms.

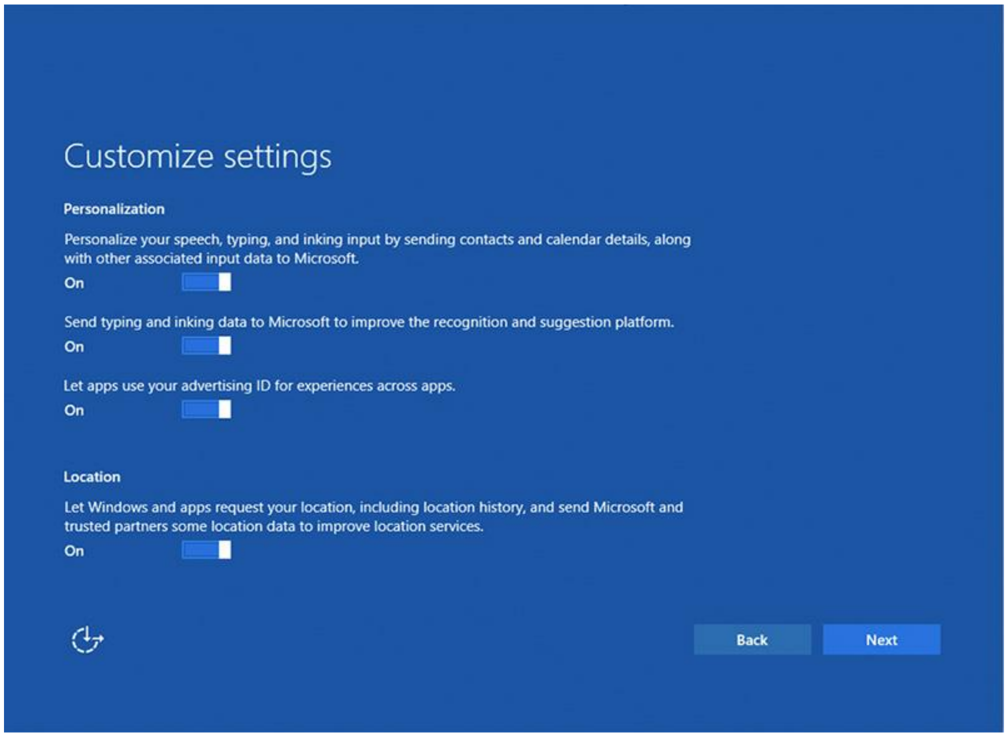
Let Windows and apps request your location, including location history, and use your advertising ID to personalize your experiences. Send Microsoft and trusted partners some location data to improve location services.

Help protect you from malicious web content and use page prediction to improve reading, speed up browsing, and make your overall experience better in Windows browsers. Your browsing data will be sent to Microsoft.

Automatically connect to suggested open hotspots and shared networks. Not all networks are secure.

Send error and diagnostic information to Microsoft.

Step 10: If you choose to customize the settings, the first page deals with your contact, calendar, input, and location data. Read these carefully to decide if you want to turn the settings on or off.



Customize settings

Personalization

Personalize your speech, typing, and inking input by sending contacts and calendar details, along with other associated input data to Microsoft.

On

Send typing and inking data to Microsoft to improve the recognition and suggestion platform.

On

Let apps use your advertising ID for experiences across apps.

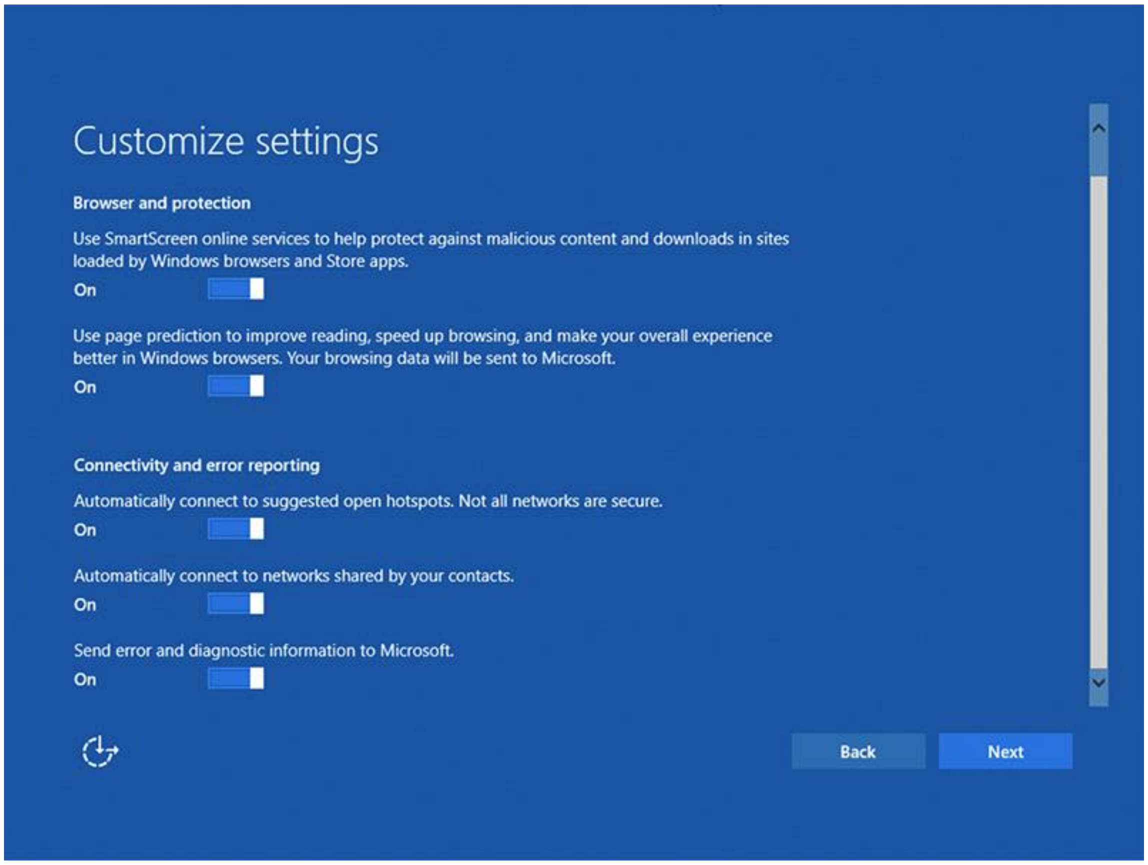
On

Location

Let Windows and apps request your location, including location history, and send Microsoft and trusted partners some location data to improve location services.

On

Step 11: The next page deals with browser data, connectivity, and error reporting.



Customize settings

Browser and protection

Use SmartScreen online services to help protect against malicious content and downloads in sites loaded by Windows browsers and Store apps.

On

Use page prediction to improve reading, speed up browsing, and make your overall experience better in Windows browsers. Your browsing data will be sent to Microsoft.

On

Connectivity and error reporting

Automatically connect to suggested open hotspots. Not all networks are secure.

On

Automatically connect to networks shared by your contacts.

On

Send error and diagnostic information to Microsoft.

On

USER ACCOUNT CREATION & PERMISSIONS

- Admin
- Multiple User

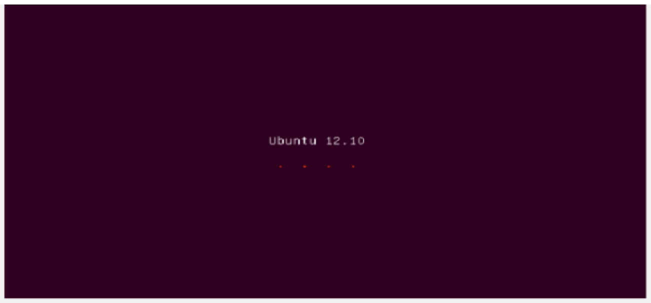
THANK YOU

III)Ubuntu (LINUX) Installation

Ubuntu is a Debian-based Linux operating system and distribution for personal computers, smart phones and network servers. It uses Unity as its default desktop environment. It is based on free software and named after the Southern African philosophy of *ubuntu* (literally, "human-ness"), which often is translated as "humanity towards others" or "the belief in a universal bond of sharing that connects all humanity".

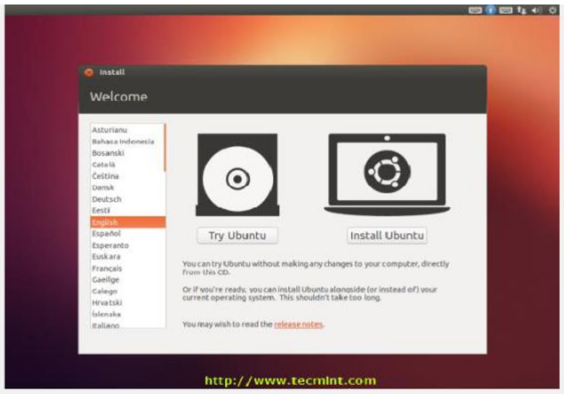
Installation process:

1. Boot Computer with Ubuntu 12.10 Installation CD/DVD or ISO.



Boot Ubuntu 12.10

2. Welcome screen of Ubuntu 12.10



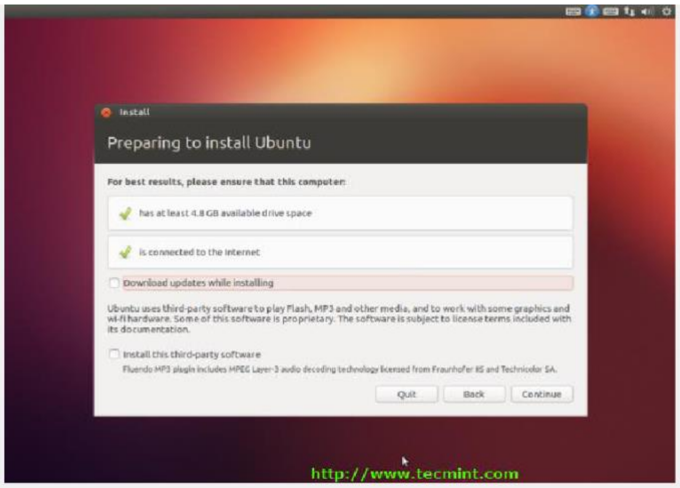
You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

Ubuntu 12.10 Welcome Scree

Preparing to install Ubuntu, select options and click continue.


Install

Preparing to install Ubuntu

For best results, please ensure that this computer:

✓ has at least 4.8 GB available drive space

✓ is connected to the internet

☐ Download updates while installing

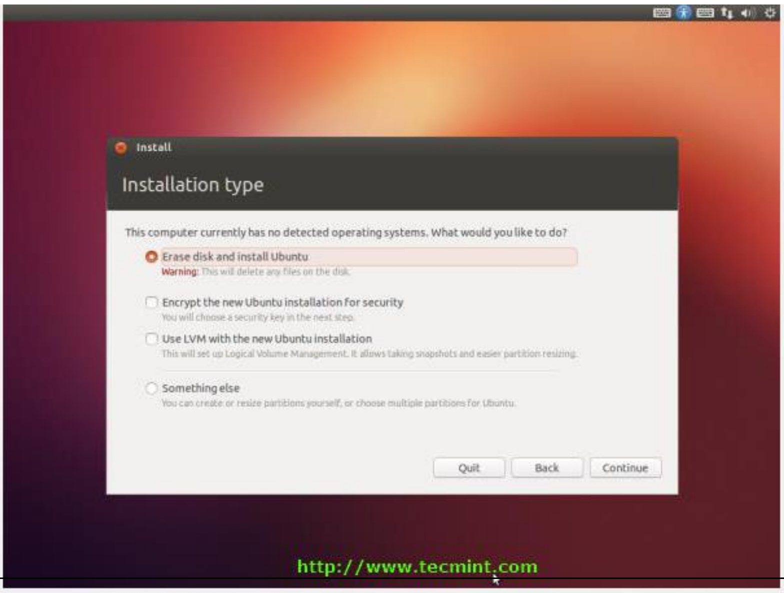
Ubuntu uses third-party software to play Flash, MP3 and other media, and to work with some graphics and wi-fi hardware. Some of this software is proprietary. The software is subject to license terms included with its documentation.

☐ Install this third-party software

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Preparing Ubuntu 12.10 Installation

3. Installation type selection.


Install

Installation type

This computer currently has no detected operating systems. What would you like to do?

[✓ Erase disk and install Ubuntu]
Warning: This will delete any files on the disk.

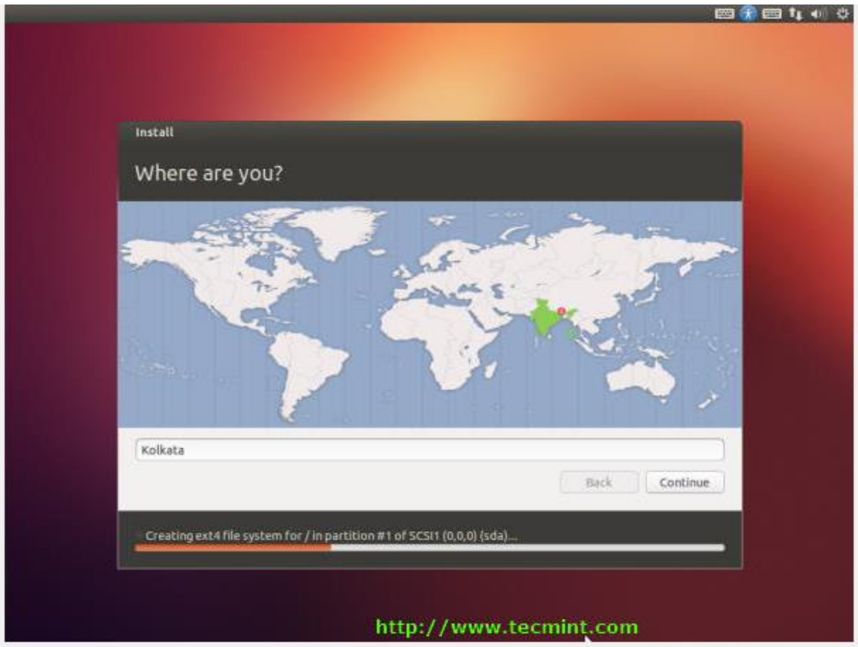
☐ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

☐ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Select Ubuntu 12.10 Installation Type

4. Select nearest city in your time zone.

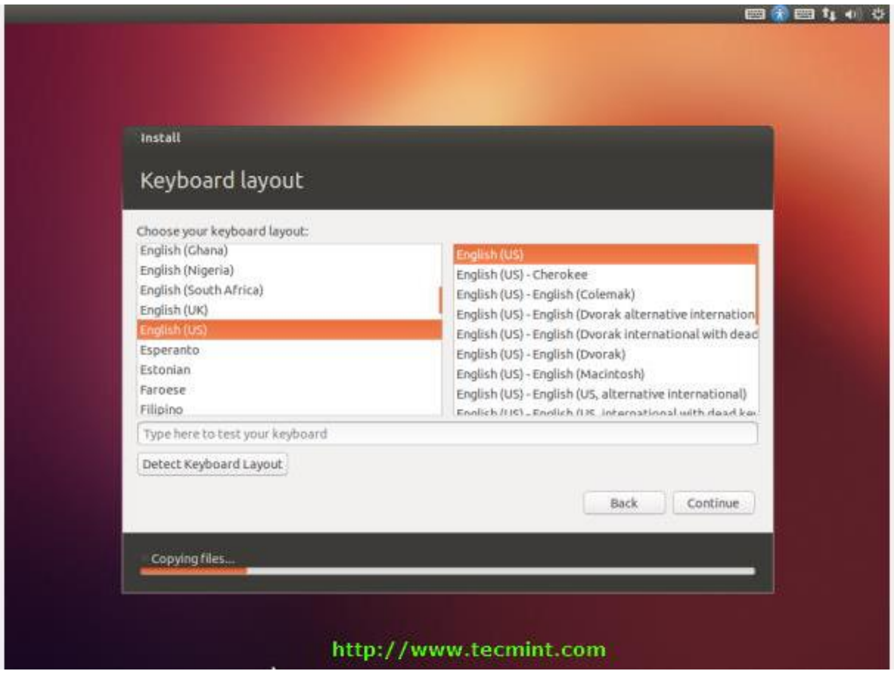


Kolkata

Creating ext4 file system for / in partition #1 of SCSI1 (0,0,0) (sda)...

Select Ubuntu 12.10 Time Zone

5. Keyboard selection.



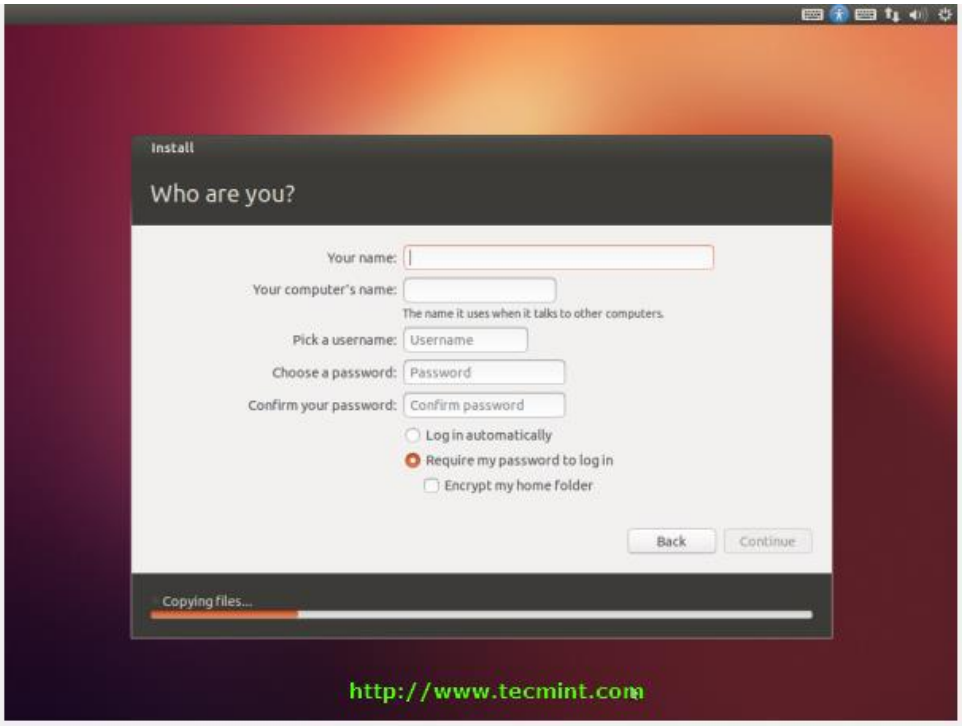
Choose your keyboard layout:

English (Ghana)	English (US)
English (Nigeria)	English (US) - Cherokee
English (South Africa)	English (US) - English (Colemak)
English (UK)	English (US) - English (Dvorak alternative internation
English (US)	English (US) - English (Dvorak international with dead
Esperanto	English (US) - English (Dvorak)
Estonian	English (US) - English (Macintosh)
Faroese	English (US) - English (US, alternative international)
Filipino	

Type here to test your keyboard

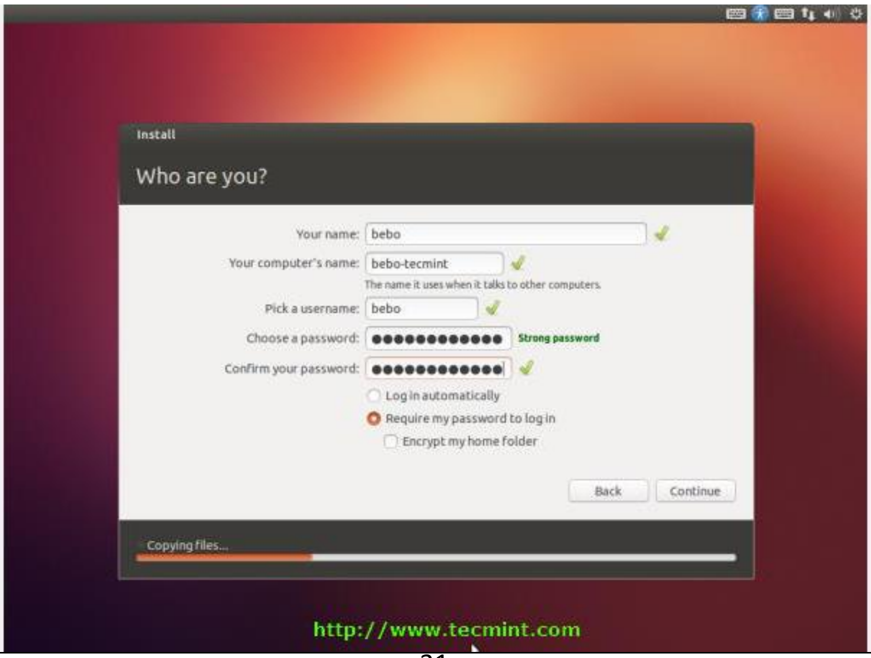
Copying files...

6. Fill information and continue.

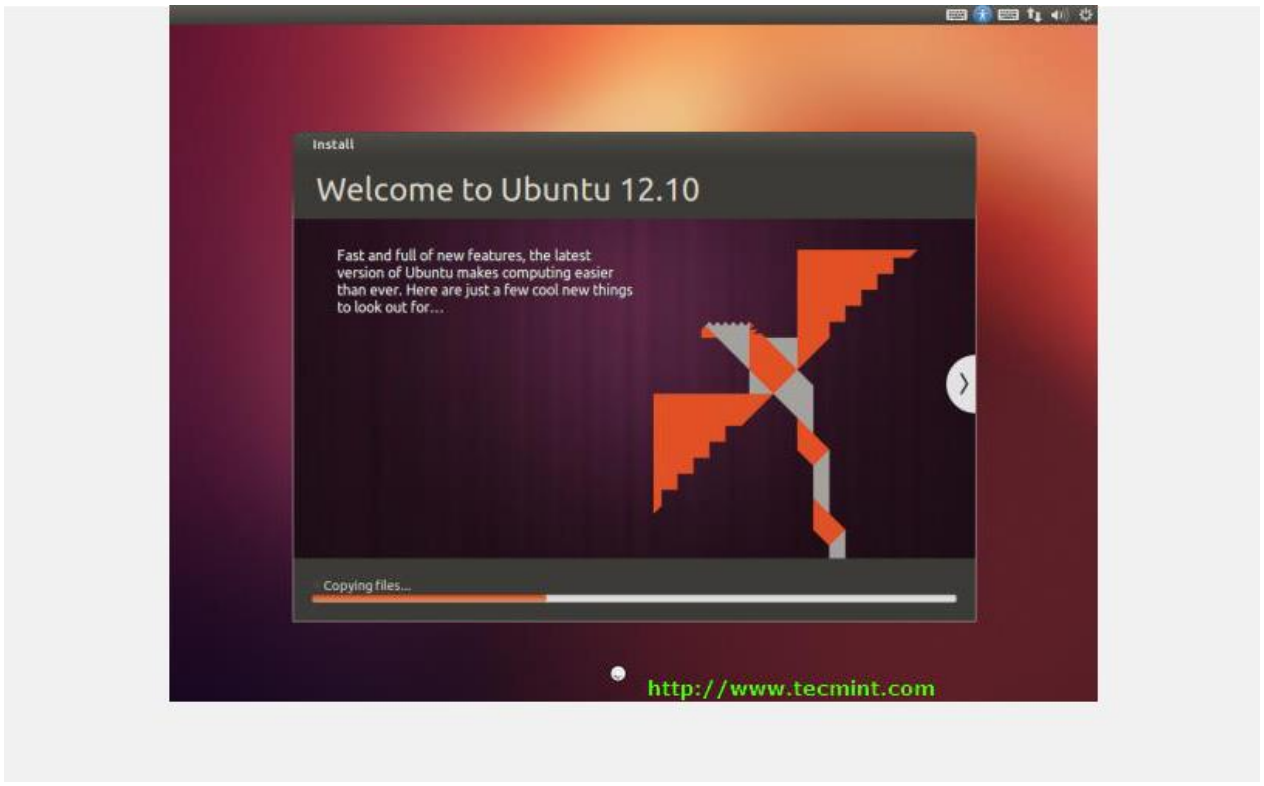

Install
Who are you?
Your name:
Your computer's name:
The name it uses when it talks to other computers.
Pick a username: Username
Choose a password: Password
Confirm your password: Confirm password
Log in automatically
Require my password to log in
Encrypt my home folder
Back
Continue
Copying files...
http://www.tecmint.com

Enter Ubuntu 12.10 Information

7. Installation is in progress. This may take several minutes.


Install
Who are you?
Your name: bebo
Your computer's name: bebo-tecmint
The name it uses when it talks to other computers.
Pick a username: bebo
Choose a password: ●●●●●●●●●●●● Strong password
Confirm your password: ●●●●●●●●●●●●
Log in automatically
Require my password to log in
Encrypt my home folder
Back
Continue
Copying files...
http://www.tecmint.com

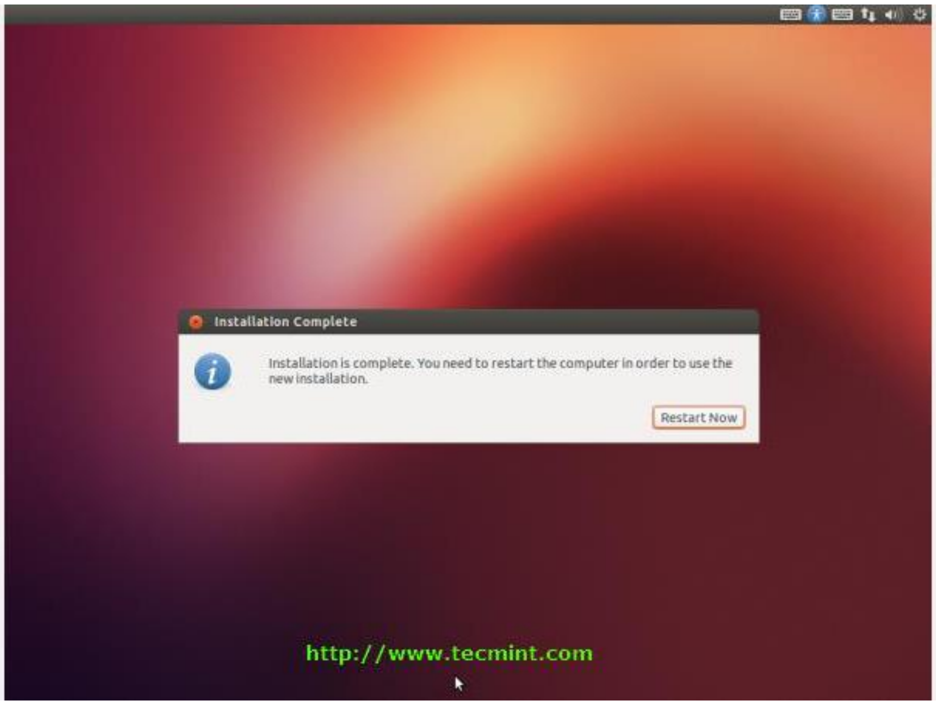
9. Installation is completed, eject CD/DVD and restart.



Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for...

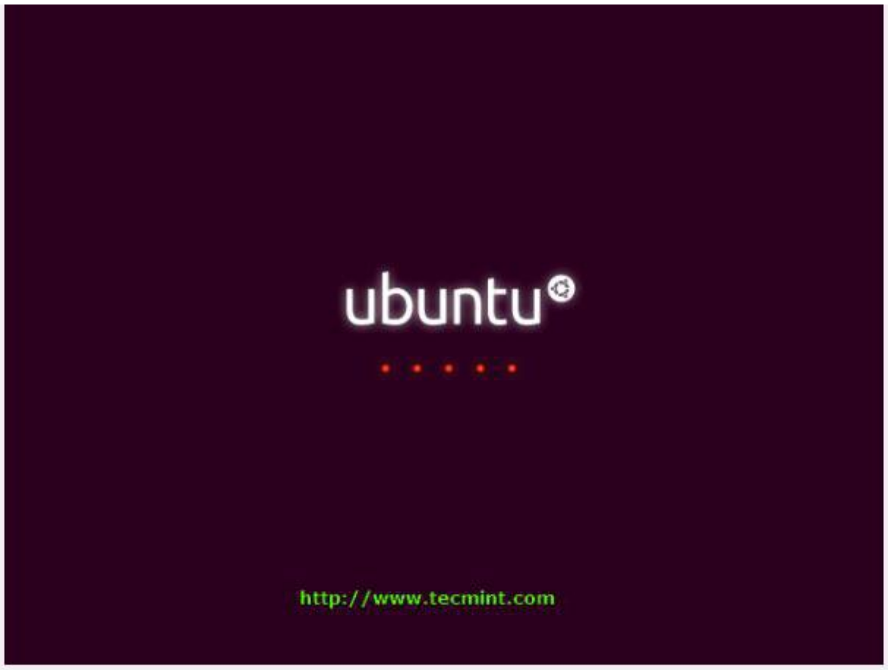
Ubuntu 12.10 Installation Completed

*10.*Booting your Ubuntu Box.


Installation Complete

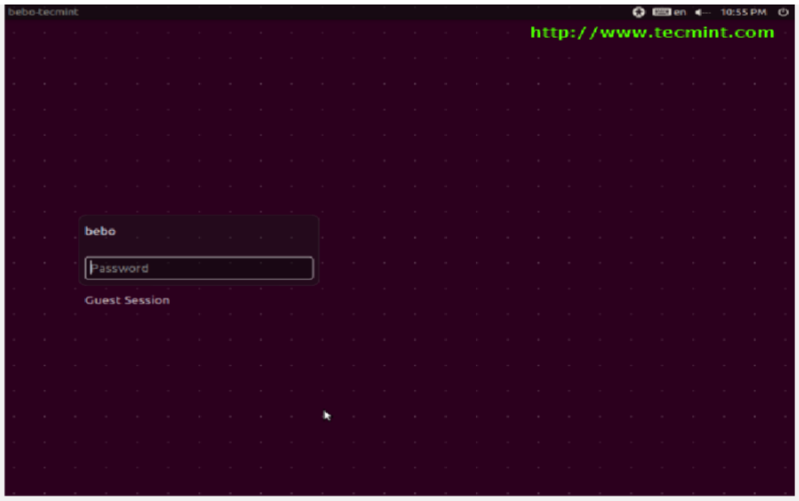
Installation is complete. You need to restart the computer in order to use the new installation.

11. Login screen.

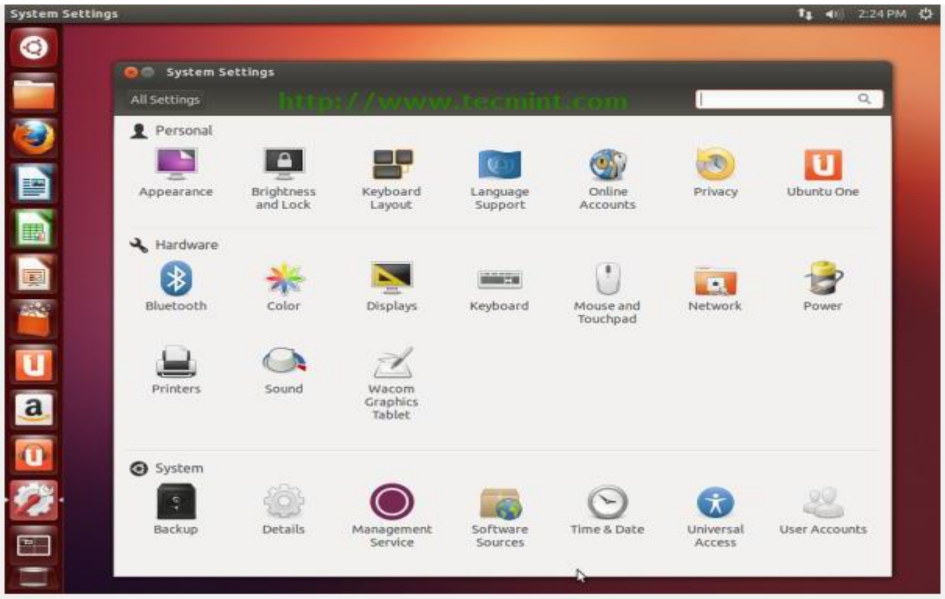


Ubuntu 12.10 Login Screen

12. Ubuntu Desktop.



13.Ubuntu Desktop with system settings.



Ubuntu 12.10 System Setting

That’s it. Ubuntu is now installed.

EXPERIMENT-2

i).Assume you have the following jobs to execute with one processor,with the jobs arriving in the order listed here:

i	T(pi)
0	80
1	20
2	10
3	20
4	50

with the following values write a program to get the required output which is listed below

a. Suppose a system uses FCFS scheduling .Create a Gantt chart illustrating the execution of these processes?

b. What is the turnaround time for process p3?

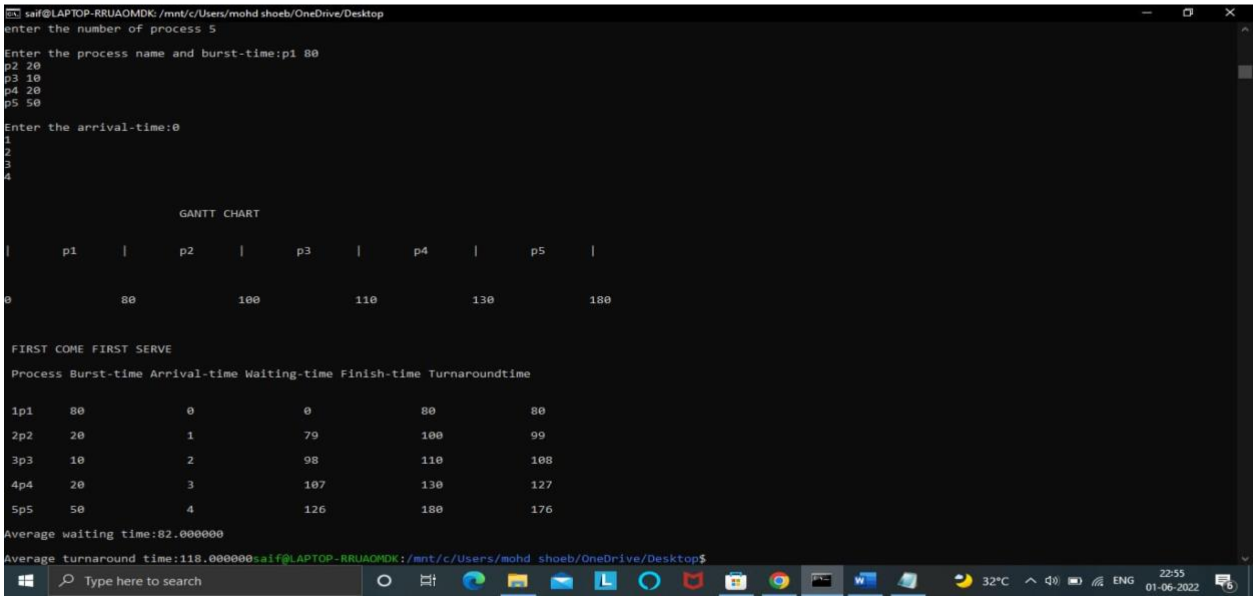
c. What is the average wait time for the processes?

Program:

```
#include<stdio.h>
void main()
{
int bt[50],wt[80],at[80],wat[30],ft[80],tat[80];
int i,n;
float awt,att,sum=0,sum1=0;
char p[10][5];
printf("\nenter the number of process ");
scanf("%d",&n);
printf("\nEnter the process name and burst-time:");
for(i=0;i<n;i++)
scanf("%s%d",p[i],&bt[i]);
```

```
printf("\nEnter the arrival-time:");

for(i=0;i<n;i++)

scanf("%d",&at[i]);

wt[0]=0;

for(i=1;i<=n;i++)

wt[i]=wt[i-1]+bt[i-1];

ft[0]=bt[0];

for(i=1;i<=n;i++)

ft[i]=ft[i-1]+bt[i];

printf("\n\n\t\t\tGANTT CHART\n");

printf("\n \n");

for(i=0;i<n;i++)

printf("|\t%s\t",p[i]);

printf("|\t\n");

printf("\n \n");

printf("\n");

for(i=0;i<n;i++)

printf("%d\t\t",wt[i]);

printf("%d",wt[n]+bt[n]);

printf("\n \n");

printf("\n");

for(i=0;i<n;i++)

wat[i]=wt[i]-at[i];

for(i=0;i<n;i++)

tat[i]=ft[i]-at[i];

printf("\n FIRST COME FIRST SERVE\n");
printf("\n Process Burst-time Arrival-time Waiting-time Finish-time Turnaroundtime\n");

for(i=0;i<n;i++)

printf("\n\n %d%s \t %d\t\t %d \t\t %d\t\t %d \t\t%d",i+1,p[i],bt[i],at[i],wat[i],ft[i],tat[i]);

for(i=0;i<n;i++)

sum=sum+wat[i];

awt=sum/n;

for(i=0;i<n;i++)

sum1=sum1+tat[i];

att=sum1/n;

printf("\n\nAverage waiting time:%f",awt);

printf("\n\nAverage turnaround time:%f",att);

}
```

OUTPUT:



```
saif@LAPTOP-RRUAOMDK: /mnt/c/Users/mohd shoeb/OneDrive/Desktop
enter the number of process 5

Enter the process name and burst-time:p1 80
p2 20
p3 10
p4 20
p5 50

Enter the arrival-time:0
1
2
3
4


                        GANTT CHART


|       p1      |       p2      |       p3      |       p4      |       p5      |


0               80              100             110             130             180


 FIRST COME FIRST SERVE

 Process Burst-time Arrival-time Waiting-time Finish-time Turnaroundtime


 1p1     80              0               0               80              80

 2p2     20              1               79              100             99

 3p3     10              2               98              110             108

 4p4     20              3               107             130             127

 5p5     50              4               126             180             176

Average waiting time:82.000000

Average turnaround time:118.000000saif@LAPTOP-RRUAOMDK:/mnt/c/Users/mohd shoeb/OneDrive/Desktop$
```

ii) Write a C program to illustrate the following IPC mechanisms
a) Pipes b) FIFOs c) Message Queue d)Shared Memory

a)Pipes

```
#include<stdio.h>
#include<unistd.h>
int main() {
  int pipefds[2];
  int returnstatus;
  int pid;
  char writemessages[2][20]={"Hi", "Hello"};
  char readmessage[20];
  returnstatus = pipe(pipefds);
  if (returnstatus == -1) {
    printf("Unable to create pipe\n");
    return 1;
  }
  pid = fork();
    // Child process
 if (pid == 0) {
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Child Process - Reading from pipe – Message 1 is %s\n", readmessage);
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Child Process - Reading from pipe – Message 2 is %s\n", readmessage);
  } else { //Parent process
    printf("Parent Process - Writing to pipe - Message 1 is %s\n", writemessages[0]);
    write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
    printf("Parent Process - Writing to pipe - Message 2 is %s\n", writemessages[1]);
    write(pipefds[1], writemessages[1], sizeof(writemessages[1]));
  }
  return 0;
}
```

OUTPUT:

b)FIFO

fifo1.c:

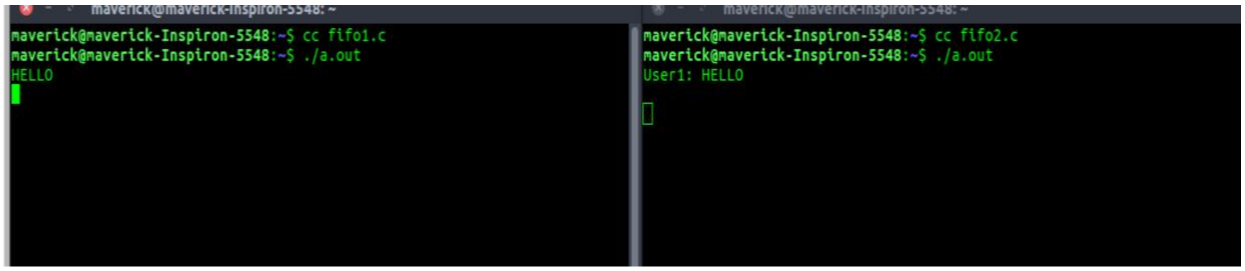
```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd;

    // FIFO file path
    char * myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)
    // mkfifo(<pathname>, <permission>)
    mkfifo(myfifo, 0666);

    char arr1[80], arr2[80];
    while (1)
    {
        // Open FIFO for write only
        fd = open(myfifo, O_WRONLY);
// Take an input arr2ing from user.
        // 80 is maximum length
        fgets(arr2, 80, stdin);

        // Write the input arr2ing on FIFO
        // and close it
        write(fd, arr2, strlen(arr2)+1);
        close(fd);

        // Open FIFO for Read only
        fd = open(myfifo, O_RDONLY);

        // Read from FIFO
        read(fd, arr1, sizeof(arr1));

        // Print the read message
        printf("User2: %s\n", arr1);
        close(fd);
    }
    return 0;
}
```

Fifo2.c:

```
 #include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
```

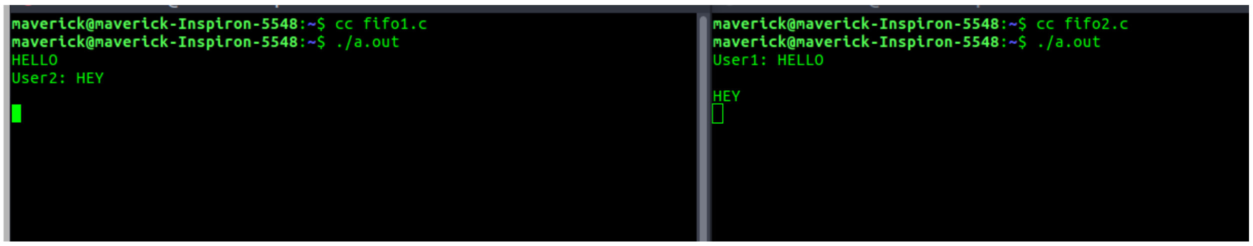
```
{
   int fd1;

   // FIFO file path
   char * myfifo = "/tmp/myfifo";

   // Creating the named file(FIFO)
   // mkfifo(<pathname>,<permission>)
   mkfifo(myfifo, 0666);

   char str1[80], str2[80];
   while (1)
   {
      // First open in read only and read
      fd1 = open(myfifo,O_RDONLY);
      read(fd1, str1, 80);

      // Print the read string and close
      printf("User1: %s\n", str1);
      close(fd1);

      // Now open in write mode and write
      // string taken from user.
      fd1 = open(myfifo,O_WRONLY);
      fgets(str2, 80, stdin);
      write(fd1, str2, strlen(str2)+1);
      close(fd1);
   }
   return 0;
}
```

OUTPUT:



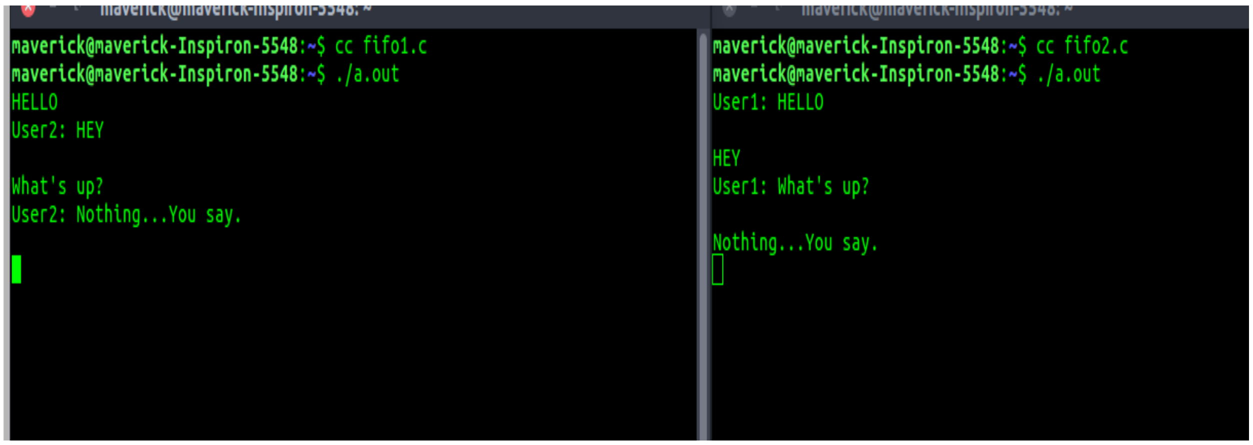
```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO
```



```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
```



```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
HELLO
User2: HEY

What's up?
User2: Nothing...You say.
```

```
maverick@maverick-Inspiron-5548:~$ cc fifo2.c
maverick@maverick-Inspiron-5548:~$ ./a.out
User1: HELLO

HEY
User1: What's up?

Nothing...You say.
```

C. Message Queues

File: msgq_send.c

```
/* Filename: msgq_send.c */
client.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

// message queue structure
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    // generate unique key
    key = ftok("somefile", 65);

// create a message queue and return identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;

    printf("Insert message  : ");
    gets(message.mesg_text);

    // send message
    msgsnd(msgid, &message, sizeof(message), 0);


    // display the message
    printf("Message sent to server : %s\n", message.mesg_text);


    return 0;
}
```

OUTPUT:

File: msgq_recv.c

```
/* Filename: msgq_recv.c */
server.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

// structure for message queue
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    // generate unique key
    key = ftok("somefile", 65);

// create a message queue and return identifier
    msgid = msgget(key, 0666 | IPC_CREAT);

    printf("Waiting for a message from client...\n");
    // receive message
    msgrcv(msgid, &message, sizeof(message), 1, 0);

    // display the message
    printf("Message received from client : %s\n",message.mesg_text);

    // to destroy the message queue
    msgctl(msgid, IPC_RMID, NULL);

    return 0;
}
```

OUTPUT:

d) SHARED MEMORY FOR WRITER PROCESS

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/shm.h>
#include <string.h>

int main()
{
        void *shared_memory;
        char buff[100];
        int shmid;
        shmid=shmget((key_t)1122,1024,0666|IPC_CREAT);
        printf("Key of Shared Memory is %d\n",shmid);
        shared_memory=shmat(shmid,NULL,0);
        printf("Process attached at %p\n",shared_memory);
        read(0,buff,100);
        strcpy(shared_memory,buff);
        printf("You wrote : %s\n",(char*)shared_memory);
}
```

OUTPUT:

SHARED MEMORY FOR READER PROCESS

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/shm.h>
#include <string.h>

int main()
{
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)1122,1024,0666);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Data read from the shared memory is : %s\n",(char*)shared_memory);
}
```

OUTPUT:

EXPERIMENT-3

i) Write a c program to solve producer-consumer problem using mutex and semaphore.

```
#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

/*

This program provides a possible solution for producer-consumer problem using mutex and
semaphore.

I have used 5 producers and 5 consumers to demonstrate the solution. You can always play with
these values.

*/

#define MaxItems 5 // Maximum items a producer can produce or a consumer can consume

#define BufferSize 5 // Size of the buffer

sem_t empty;

sem_t full;

int in = 0;

int out = 0;

int buffer[BufferSize];

pthread_mutex_t mutex; //declaration of mutex


void *producer(void *pno)

{
    int item;
    for(int i = 0; i < MaxItems; i++) {
        item = rand(); // Produce an random item
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Producer %d: Insert Item %d at %d\n", *((int *)pno),buffer[in],in);
        in = (in+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
void *consumer(void *cno)
{
    for(int i = 0; i < MaxItems; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item, out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}
```

```
int main()
{

    pthread_t pro[5],con[5];//declaring 5 producer threads and 5 consumer threads
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,BufferSize);
    sem_init(&full,0,0);

    int a[5] = {1,2,3,4,5}; //Just used for numbering the producer and consumer

    for(int i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }
//threads are created in heap, so destroy them before exit
```

```
    pthread_mutex_destroy(&mutex);

    sem_destroy(&empty);

    sem_destroy(&full);


    return 0;


}
```

OUPUT:

ii) Implement following memory allocation policies with given data

First Fit

Best Fit

Available memory: 35 units

OS : 10 units

User process : 25 units

Time of arrival (units)	0	3	7	12	18	25	29
Processing Time (units)	5	3	9	10	16	2	8
Memory Required (units)	3	5	8	12	2	6	9

PROGRAM

FIRST FIT

```
#include<stdio.h>

void main()
{
int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;

for(i = 0; i < 10; i++)
{
flags[i] = 0;
allocation[i] = -1;
}
printf("Enter no. of blocks: ");
scanf("%d", &bno);
printf("\nEnter size of each block: ");
for(i = 0; i < bno; i++)
scanf("%d", &bsize[i]);

printf("\nEnter no. of processes: ");
scanf("%d", &pno);
printf("\nEnter size of each process: ");
for(i = 0; i < pno; i++)
scanf("%d", &psize[i]);
for(i = 0; i < pno; i++)         //allocation as per first fit
for(j = 0; j < bno; j++)
if(flags[j] == 0 && bsize[j] >= psize[i])
{
allocation[j] = i;
flags[j] = 1;
break;
}
//display allocation details
printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
for(i = 0; i < bno; i++)
{
printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
if(flags[i] == 1)
printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);
else
printf("Not allocated");
}
}
```

Output:

```
Enter the number of blocks: 3

Enter the size of each block: 12

7

4

Enter the number of processes: 3

Enter the size of each process: 7

4

9

Block no. Size Process No. Size

1 12 1 7
```

```
2 7 2 4
3 4 Not Allocated
```

BEST FIT

```
#include<stdio.h>
// main function
void main()
{
    // declaration and initialization of variables
    int fragment[20],b[20],p[20],i,j,nb,np,tem,low=9999;
    static int barray[20],parray[20];
    printf("Memory Management Scheme - Best Fit");
    //input number of processes
    printf("Enter the number of processes:");
    scanf("%d",&n_p);
    //input number of blocks
    printf("\nEnter the number of blocks:");
    scanf("%d",&n_b);
    //input number of blocks
    printf("\nEnter the size of the blocks:-\n");
    for(i=1;i<=n_b;i++)
    {
        printf("Block no.%d:",i);
        scanf("%d",&b[i]);
    }
```

```
        //input the size of process
        printf("\nEnter the size of the processes :-\n");
        for(i=1;i<=n_p;i++)
{
    printf("Process no.%d:",i);

    scanf("%d",&p[i]);
}
        for(i=1;i<=n_p;i++)
        {
                for(j=1;j<=n_b;j++)
                {
                        if(barray[j]!=1)
                        {
                                tem=b[j]-p[i];
                                if(tem>=0)
                                        if(low>temp)
                                        {
                                                parray[i]=j;
                                                low=tem;
                                        }
                        }
                }
                fragment[i]=low;
                barray[parray[i]]=1;
                low=10000;
        }
        //Print the result
         printf("\nProcess_number \tProcess_size\tBlock_number \tBlock_size\tFragment");
        for(i=1;i<=np && parray[i]!=0;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]);
}
```

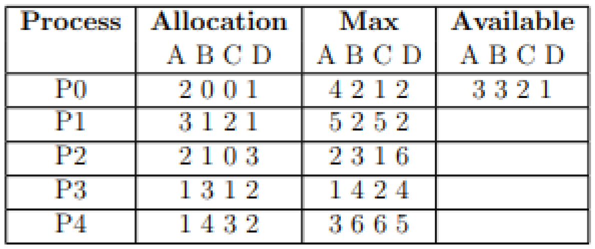
Output

```
Enter the number of blocks: 5
Enter the number of processes: 4
Enter the size of the blocks:
Block number 1: 10
Block number 2: 15
Block number 3: 5
Block number 4: 9?
Block number 5: 3
Enter the size if the process
Process number 1: 1
Process number 2: 4
Process number 3: 7
Process number 4: 12
```

Process number	Process size	Block number	Block size	Fragment
1	1	5	3	2
2	4	3	5	1
3	7	4	9	2
4	12	2	15	3

EXPERIMENT 4

Consider the following snapshot of a system. P0, P1, P2, P3, P4 are the processes and A, B, C, D are the resource types. The values in the table indicates the number of instances of a specific resource (for example: 3 3 2 1 under the last column indicates that there are 3 A-type, 3 B-type, 2 C-type and 1 D-type resources are available after allocating the resources to all five processes). The numbers under allocation-column indicate that those number of resources are allocated to various processes mentioned in the first column. The numbers under Max-column indicate the maximum number of resources required by the processes. For example: in 1st row under allocation-column 2 0 0 1 indicate there are 2 A-type, 0 B-type, 0 C-type and 1 D-type resources are allocated to process P0. Whereas 4 2 1 2 under Max-column indicate that process P0's maximum requirement is 4 A-type, 2 B-type, 1 C-type and 2 D-type resources.



Process	Allocation A B C D	Max A B C D	Available A B C D
P0	2 0 0 1	4 2 1 2	3 3 2 1
P1	3 1 2 1	5 2 5 2	
P2	2 1 0 3	2 3 1 6	
P3	1 3 1 2	1 4 2 4	
P4	1 4 3 2	3 6 6 5	

Answer the following questions using banker's algorithm by providing all intermediate steps

i. How many instances of resources are present in the system under each type of a resource?

ii. Compute the Need matrix for the given snapshot of a system.

iii. Verify whether the snapshot of the present system is in a safe state by demonstrating an order in

which the processes may complete. iv. If a request from process P1 arrives for (1,1,0,0), can the

request be granted immediately? v. If a request from process P4 arrives for (0,0,2,0), can the request be granted immediately?

PROGRAM

```
#include<stdio.h>

#include <stdbool.h>

bool check(int resources,int need[resources],int available[resources]);

void getSafeSequence(int processors,int resources,int allocated[processors][resources],int
max[processors][resources],int need[processors][resources],int *available);
```

```
void check_request(int process_number,int processors,int resources,int request[resources],int
allocated[processors][resources],int max[processors][resources],int need[processors][resources],int
available[resources]);

 int main(){

 int processors,resources;

 printf("Enter number of processors : "); scanf("%d",&processors);

 printf("Enter number of resources : "); scanf("%d",&resources);

 int allocated[processors][resources],max[processors][resources];

 printf("Enter Allocation Matrix\n");

 for(int i=0;i<processors;i++){

   for(int j=0;j<resources;j++){

     scanf("%d",&allocated[i][j]);

   }

 }

 printf("Enter Max Matrix\n");

 for(int i=0;i<processors;i++){

   for(int j=0;j<resources;j++){

     scanf("%d",&max[i][j]);

   }

 }

 int available[resources];

 for(int i=0;i<resources;i++){

   printf("\nEnter available of resource %c :",(i+65)); //(i+65) prints the characters starting from A (Ascii
value of A is 65)

   scanf("%d",&available[i]);

 }
 printf("\nThe Number Of Instances Of Resource Present In The System Under Each Type Of Resource
are :\n");
 //total instances are sum of resources available and allocated for each processor
 int instances[resources];
 for(int i=0;i<resources;i++){instances[i]=0;}
 for(int i=0;i<processors;i++){
  for(int j=0;j<resources;j++){
   instances[j]+=allocated[i][j];
  }
 }
 for(int i=0;i<resources;i++){instances[i]+=available[i];}
 for(int i=0;i<resources;i++){printf("%c = %d\n",(i+65),instances[i]);}
 printf("\nThe Need Matrix is \n"); //need matrix is the difference between maximum and allocated
 int need[processors][resources];
 for(int i=0;i<processors;i++){
  for(int j=0;j<resources;j++){
   need[i][j]=max[i][j]-allocated[i][j];
   printf("%d ",need[i][j]);
  }
  printf("\n"); //just to allign the matrix to next row
 }
 int current_available[processors];
 for(int i=0;i<processors;i++){current_available[i]=available[i];}
 getSafeSequence(processors,resources,allocated,max,need,current_available);
```

```
  printf("\n\nIf a request from process p1 arrives for (1,1,0,0), can the request be granted?"); //according
to question only 4 resources

  int request1[4];

  request1[0]=1; request1[1]=1;

  int current_available1[processors];

  for(int i=0;i<processors;i++){current_available1[i]=available[i];}

  check_request(1,processors,resources,request1,allocated,max,need,current_available1);


  printf("\n\nIf a request from process p4 arrives for (0,0,2,0), can the request be granted?\n");

  int request2[4];

  request2[2]=2;

  int current_available2[processors];

  for(int i=0;i<processors;i++){current_available2[i]=available[i];}

  check_request(4,processors,resources,request2,allocated,max,need,current_available2);

}

void getSafeSequence(int processors,int resources,int allocated[processors][resources],int
max[processors][resources],int need[processors][resources],int available[resources]){

  int computed=0;

  int computed_order[processors]; //order of computed processors

  int pointer_to_computed=0; //a pointer to add order of processors to computed_order

  bool processed[processors];

  for(int i=0;i<processors;i++){processed[i]=false;} //boolean array that stores whether process is
completed or not

  while(computed<processors){

    bool any_process_computed=false;

    for(int i=0;i<processors;i++){

      if(processed[i]){continue;}
      if(check(resources,need[i],available)){ //check if need is less than available
        for(int j=0;j<resources;j++){
          available[j]+=allocated[i][j]; //adding the allocated resources because the process is completed
        }
        processed[i]=true;
        any_process_computed=true;
        computed_order[pointer_to_computed++]=i;
        computed+=1;
      }
    }
    if(!any_process_computed){break;} //the system is not in safe state because no resource can be
allocated
  }
  if(computed==processors){ //if all processors are computed the system is in safe state
    printf("\nThe System is in safe state and the safe sequence is :\n");
    for(int i=0;i<processors;i++){printf("P%d ",computed_order[i]);}
  }
  else{
    printf("The System is not in safe state and the processes will be in deadlock\n");
  }
}
bool check(int resources,int need[resources],int available[resources]){
  for(int i=0;i<resources;i++){
    if(need[i]>available[i]){return false;}
  }
  return true;
```

```
}

void check_request(int process_number,int processors,int resources,int request[resources],int
allocated[processors][resources],int max[processors][resources],int need[processors][resources],int
available[resources]){

 if(check(resources,request,available)){

  if(check(resources,request,need[process_number])){

   for(int i=0;i<resources;i++){

    available[i]-=request[i];

    allocated[process_number][i]+=request[i];

    need[process_number][i]=max[process_number][i]-allocated[process_number][i];

   }

   getSafeSequence(processors,resources,allocated,max,need,available);

   return;

  }

 }

 printf("Request cannot be granted\n");

}
```

OUTPUT

```
Enter number of processors :   5
Enter number of resources  :   4
Enter Allocation Matrix
2 0 0 1
3 1 2 1
2 1 0 3
1 3 1 2
1 4 3 2
Enter Max Matrix
4 2 1 2
5 2 5 2
2 3 1 6
1 4 2 4
3 6 6 5
Enter available of resource A  :3
Enter available of resource B  :3
Enter available of resource C :2

Enter available of resource ❑ :1

The Number of Instances of Resource Present In The System Under Each Type of
Resource are

A = 12
B = 12
C = 8
D = 10

The Need Matrix is

2 2 1 1
2 1 3 1
0 2 1 3
0 1 1 2
2 2 3 3

The System is in safe state and the safe sequence is

P0 P3 P4 P1 P2

If a request from process p1 arrives for (1,1,0,0) can the request be granted?

The System is in safe state and the safe sequence is :

P0 P3 P4 P1 P2

If a request from process p4 arrives for (0,0,2,0) can the request be

granted? The System is not in safe state and the processes will be in deadlock
```

EXPERIMENT 5

Write a C program to simulate following memory management techniques.

a)Paging b)segmentation

a)Paging

```
#include<stdio.h>

void main()

{

int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;

int s[10], fno[10][20];

printf("\nEnter the memory size -- ");

scanf("%d",&ms);

printf("\nEnter the page size -- ");

scanf("%d",&ps);

nop = ms/ps;

printf("\nThe no. of pages available in memory are -- %d ",nop);

printf("\nEnter number of processes -- ");

scanf("%d",&np);

rempages = nop;

for(i=1;i<=np;i++)

{

printf("\nEnter no. of pages required for p[%d]-- ",i);

scanf("%d",&s[i]);

if(s[i] >rempages)
{
printf("\nMemory is Full");
break;
}
rempages = rempages - s[i];
printf("\nEnter pagetable for p[%d] --- ",i);

for(j=0;j<s[i];j++)
scanf("%d",&fno[i][j]);
}
printf("\nEnter Logical Address to find Physical Address ");
printf("\nEnter process no. and pagenumber and offset -- ");
scanf("%d %d %d",&x,&y, &offset);
if(x>np || y>=s[i] || offset>=ps)
printf("\nInvalid Process or Page Number or offset");
else
{
pa=fno[x][y]*ps+offset;
printf("\nThe Physical Address is -- %d",pa);
}
}
```

INPUT

```
Enter the memory size -- 1000
Enter the page size -- 100
```

```
The no. of pages available in memory are -- 10

Enter number of processes -- 3

Enter no. of pages required for p[1]-- 4

Enter pagetable for p[1] --- 8 6 9 5

Enter no. of pages required for p[2]-- 5

Enter pagetable for p[2] --- 1 4 5 7 3

Enter no. of pages required for p[3]-- 5
```

OUTPUT

```
Memory is Full

Enter Logical Address to find Physical Address

Enter process no. and page number and offset -- 2 3 60


The Physical Address is – 760
```

b)Segmentation:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int sost;
void gstinfo();
void ptladdr();
struct segtab
{
int sno;
int baddr;
int limit;
int val[10];
}st[10];
void gstinfo()
{
int i,j;
printf("\n\tEnter the size of the segment table: ");
scanf("%d",&sost);
for(i=1;i<=sost;i++)
{
printf("\n\tEnter the information about segment: %d",i);
st[i].sno = i;
printf("\n\tEnter the base Address: ");
scanf("%d",&st[i].baddr);
printf("\n\tEnter the Limit: ");
scanf("%d",&st[i].limit);
for(j=0;j<st[i].limit;j++)
{
printf("Enter the %d address Value: ",(st[i].baddr + j));
scanf("%d",&st[i].val[j]);
}
}
}
void ptladdr()
{
int i,swd,d=0,n,s,disp,paddr;
clrscr();
printf("\n\n\t\t\t SEGMENT TABLE \n\n");
printf("\n\t SEG.NO\tBASE ADDRESS\t LIMIT \n\n");
for(i=1;i<=sost;i++)
printf("\t\t%d \t\t%d\t\t%d\n\n",st[i].sno,st[i].baddr,st[i].limit);
printf("\n\nEnter the logical Address: ");
scanf("%d",&swd);
n=swd;
while (n != 0)
{
n=n/10;
d++;
}
s = swd/pow(10,d-1);
disp = swd%(int)pow(10,d-1);
if(s<=sost)


{
if(disp < st[s].limit)
{
paddr = st[s].baddr + disp;
printf("\n\t\tLogical Address is: %d",swd);
printf("\n\t\tMapped Physical address is: %d",paddr);
printf("\n\tThe value is: %d",( st[s].val[disp] ) );
}
else
printf("\n\t\tLimit of segment %d is high\n\n",s);
}

else
printf("\n\t\tInvalid Segment Address \n");
}
void main()
{
char ch;
clrscr();
gstinfo();
do
{
ptladdr();
printf("\n\t Do U want to Continue(Y/N)");
flushall();
scanf("%c",&ch);
}while (ch == 'Y' || ch == 'y' );
getch();
}
```

OUTPUT

INPUT AND OUTPUT:

```
Enter the size of the segment table: 3

Enter the information about segment: 1

Enter the base Address: 4

Enter the Limit: 5

Enter the 4 address Value: 11

Enter the 5 address Value: 12

Enter the 6 address Value: 13

Enter the 7 address Value: 14

Enter the 8 address Value: 15

Enter the information about segment: 2

Enter the base Address: 5

Enter the Limit: 4

Enter the 5 address Value: 21

Enter the 6 address Value: 31

Enter the 7 address Value: 41

Enter the 8 address Value: 51

Enter the information about segment: 3

Enter the base Address: 3

Enter the Limit: 4

Enter the 3 address Value: 31

Enter the 4 address Value: 41


Enter the 5 address Value: 41
Enter the 6 address Value: 51

SEGMENT TABLE

SEG.NO BASE ADDRESS LIMIT

1 4 5

2 5 4

3 3 4

Enter the logical Address: 3

Logical Address is: 3

Mapped Physical address is: 3

The value is: 31

Do U want to Continue(Y/N)

SEGMENT TABLE

SEG.NO BASE ADDRESS LIMIT

1 4 5

2 5 4

3 3 4

Enter the logical Address: 1

Logical Address is: 1


Mapped Physical address is: 4

The value is: 11

Do U want to Continue(Y/N)
```