**EXPERIMENT - 01**

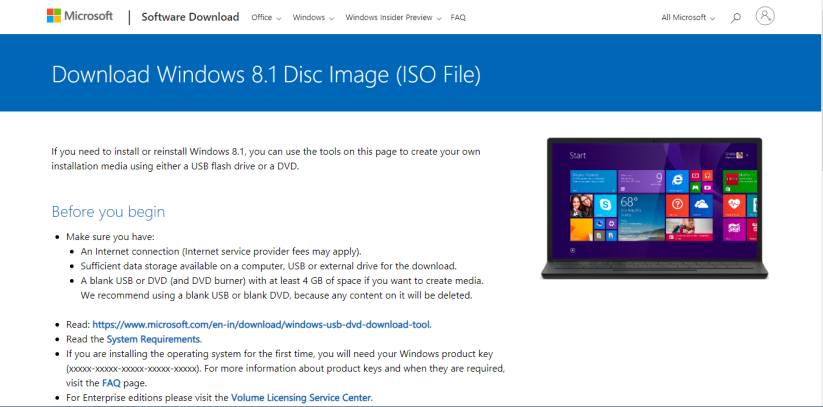
**Aim: Installation of various operating systems.**

**A). Objective:** Steps to install windows operating system.

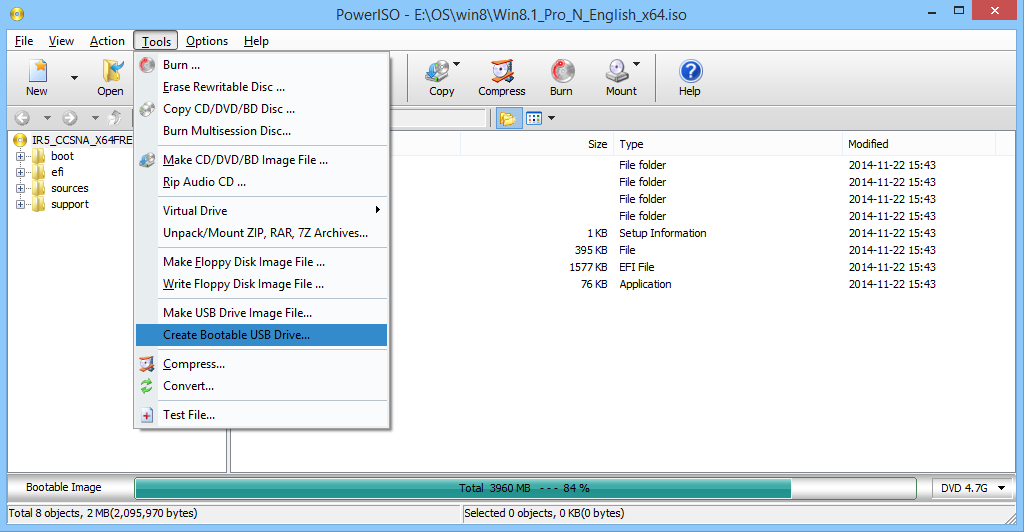
**Introductions:** Windows may be a commissioned OS within which ASCII text file is inaccessible. It is designed for the people with the angle of getting no programming information and for business and alternative industrial users. It’s terribly straightforward and simple to use.

**Steps to install windows Operating System:**

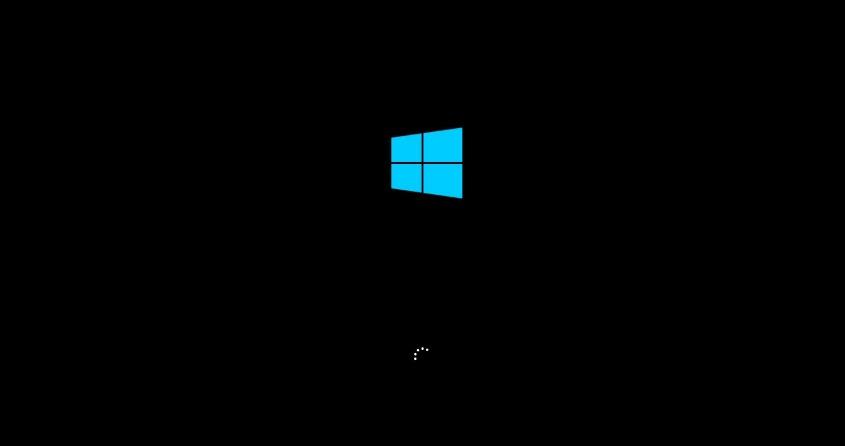
1. First get your ISO disk image file from Microsoft.com and buy a CD key.



1. Now get an USB flash drive or optical CD Drive and create a bootable device drive from PowerISO /Rufus. Here I’m using a USB flash Drive.



1. Once Completed reboot your PC and press ESC/DEL/F10 to enter into the BIOS. Check the key online according to your PC brand. In BIOS edit boot sequence program and set your bootable device on top. Now Save and Exit BIOS. If done correctly your PC will start booting from your bootable device.



1. Once booted successfully you’ll see the first window of installation. Now here Choose your Language, Time and currency format, Keyboard layout or let it be the Default i.e. English (US) and hit Next.



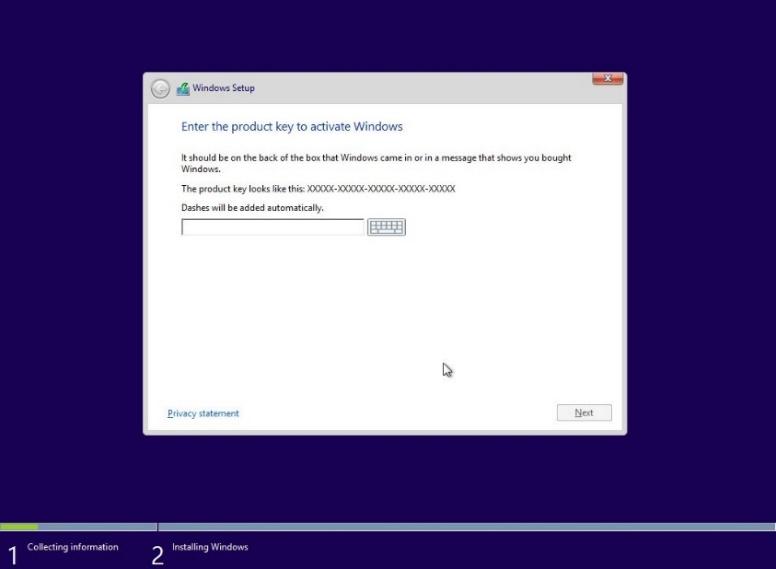
1. Now you will see a Install Now button hit it to start the setup.



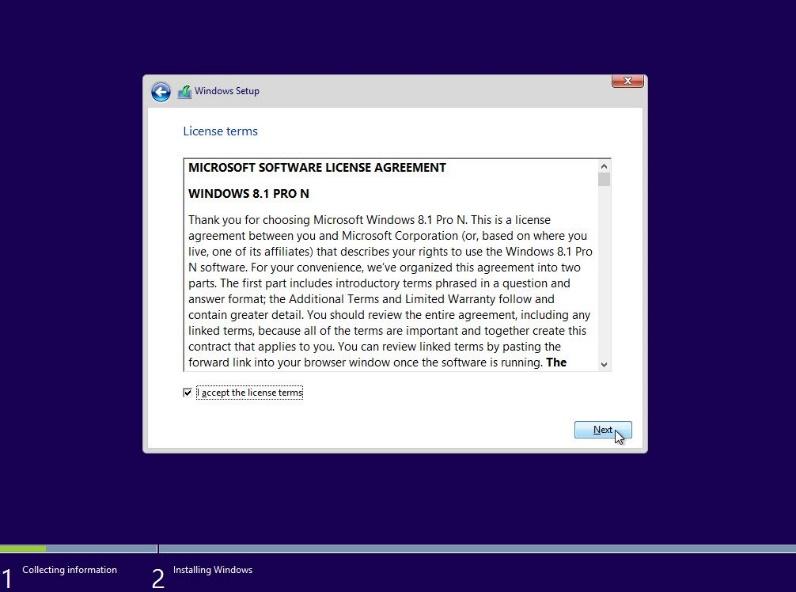
1. Once you hit Install Now the setup will start gathering hardware information and it will automatically take you to the next level.



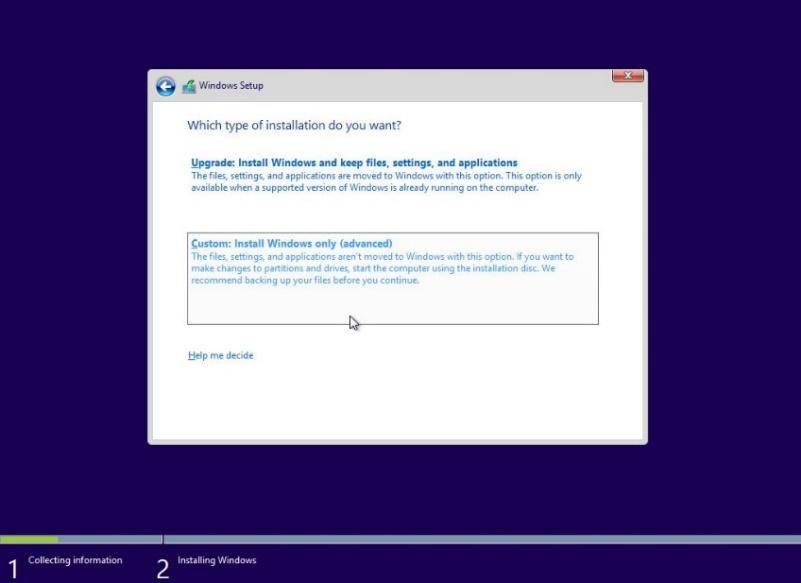
1. Now on next window it will ask you for your CD key. Enter the CD key you have bought from the Microsoft. You can use the on-screen keyboard or normal keyboard.



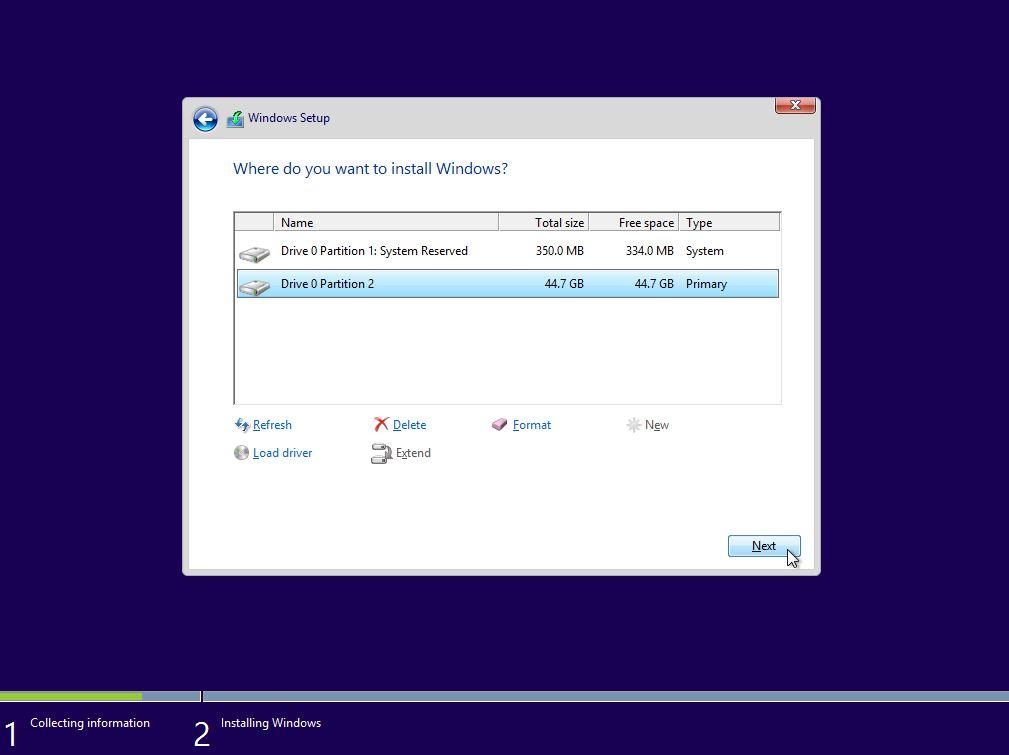
1. Now accept the Terms and Conditions from Microsoft and hit Next.



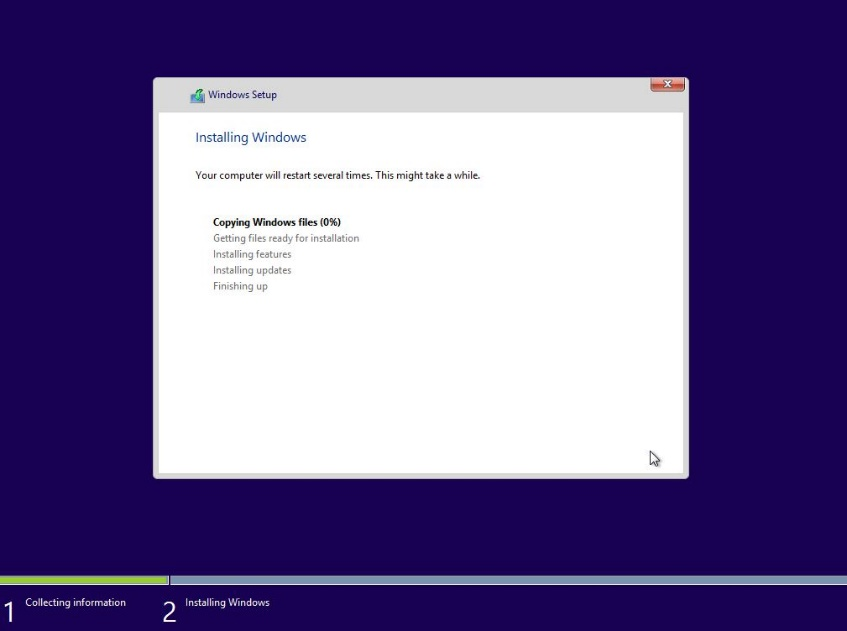
1. Now choose an option suitable for you. You can upgrade a previously installed window or custom install from the scratch. Here I’m selecting Custom installation.



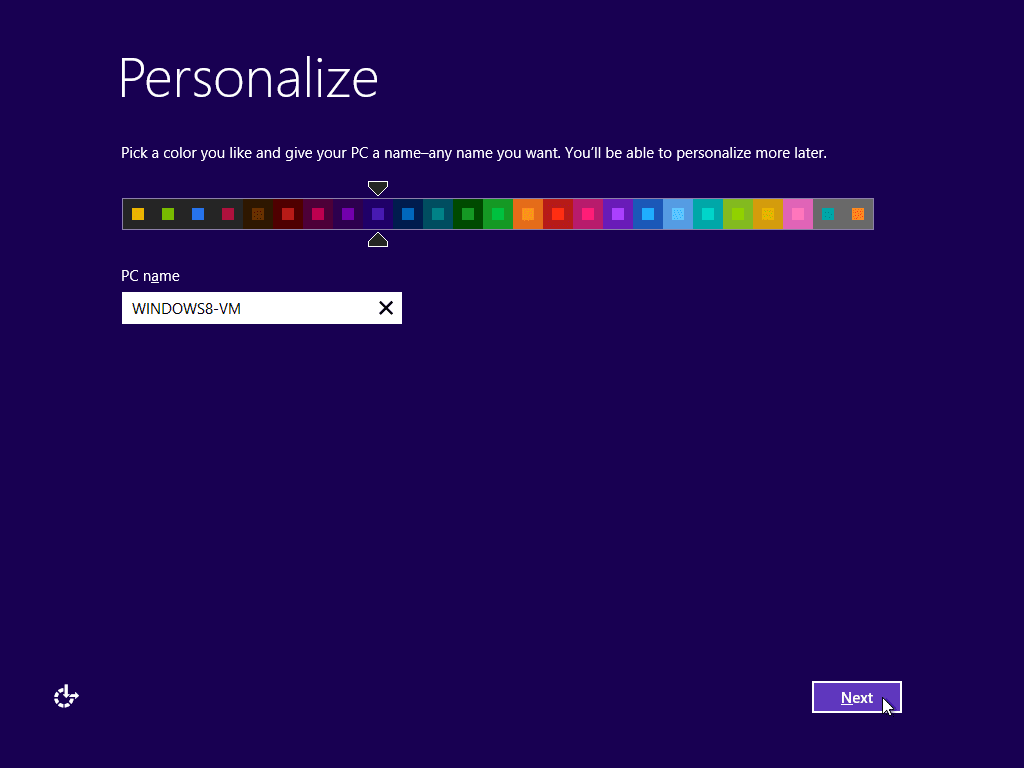
1. Now after you select Custom installation you will land on a window called as Partitioning window. Here you can create different Local Disk partitions of your HDD drive or you can select the partition containing the previous window.



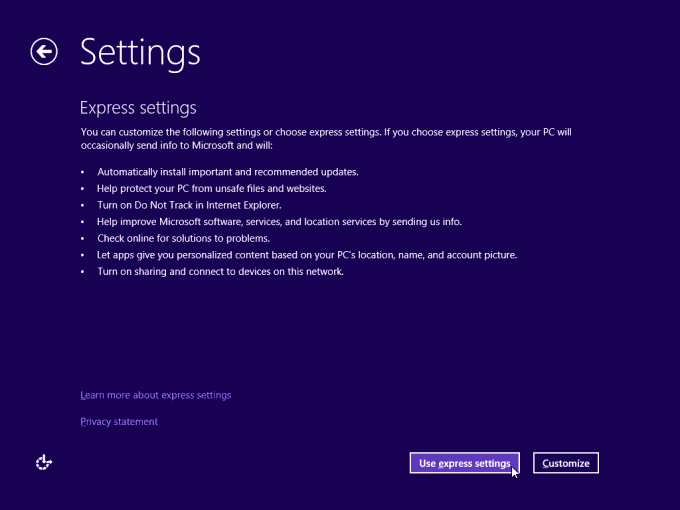
1. Once you are done with your partitioning hit Next and your window will start installing. After it finishes up the window will ask for reboot. Reboot the window and remove your bootable device before window boot again.



1. Once the window boot again you will see the this personalize section where you can change the background colour of your window choose your favourite colour and hit Next.



1. Now after Personalizing, select the type of setting you want to set for your window. Here I’m using Express settings. You can also use custom to change some settings. These setting contains Automatic Update installation, Data sharing with Microsoft etc.



1. After your settings you will be asked to Name your PC and User and set a password. You can set password as None if you want and hit Done. This will be your last step after that your window will be shown as bellow.



**Result:** Windows 8.1 is successfully installed in computer system.

**Outcome:** Students will learn about the steps to install windows operating system.

**B). Objective:** Installation of Linux Operating System.

**Introduction: Ubuntu** is an operating system with Linux kernel based on Debian and distributed as free and open source software. It’s one of the most popular operating systems for Desktop and Server.

Ubuntu can be installed in two ways as follows:

1. **Using a DVD?**

It’s easy to install Ubuntu from a DVD. Here’s what you need to do:

Put the Ubuntu DVD into the DVD-drive

Restart your computer. You should see a welcome screen prompting you to choose your language and giving you the option to install Ubuntu or try it from the DVD.

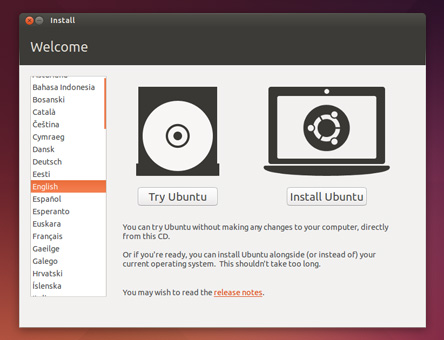
If you don’t get this menu, [read the booting from the DVD](https://help.ubuntu.com/community/BootFromCD) guide for more information.

1. **Using a USB drive**

**The steps to install Ubuntu using USB are as follows:**

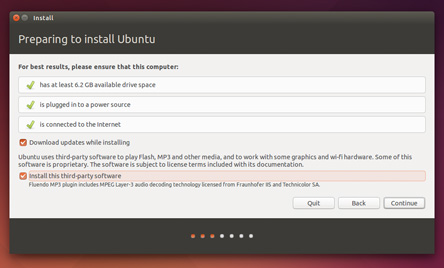
**Step 1:** Most new computers can boot from USB. You should see a welcome screen prompting you to choose your language and giving you the option to install Ubuntu or try it from the CD.

If your computer doesn’t automatically do so, you might need to press the **F12 key** to bring up the boot menu, but be careful not to hold it down – that can cause an error message.



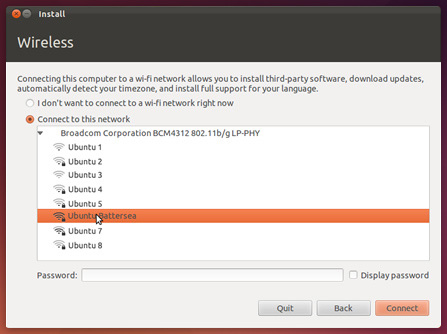
**Step 2: Prepare to install Ubuntu**

* We recommend you plug your computer into a power source.
* You should also make sure you have enough space on your computer to install Ubuntu.
* We advise you to select Download updates while installing and Install this third-party software now.
* You should also stay connected to the internet so you can get the latest updates while you install Ubuntu.
* If you’re not connected to the internet, we’ll help you set up wireless at the next step.



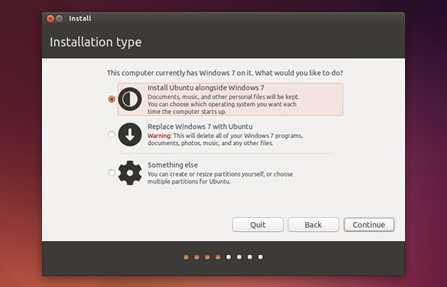
Step 3: **Set up wireless**

If you are not connected to the internet, you will be asked to select a wireless network, if available. We advise you to connect during the installation so we can ensure your machine is up to date. So, if you set up your wireless network at this point, it’s worth then clicking the Back button to go back to the last screen (Preparing to install Ubuntu) and ticking the box marked **’Download updates while installing’**.



Step 4: **Allocate drive space**

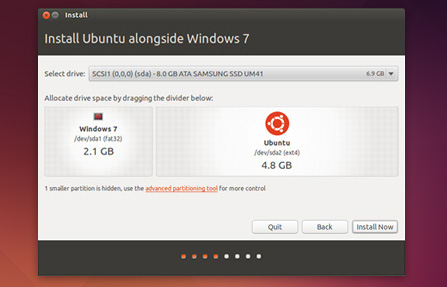
Use the checkboxes to choose whether you’d like to Install Ubuntu alongside another operating system, delete your existing operating system and replace it with Ubuntu, or — if you’re an advanced user — choose the **’Something else’** option.



Step 5: **Begin the installation**

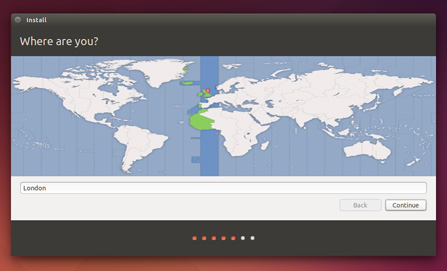
Depending on your previous selections, you can now verify that you have chosen the way in which you would like to install Ubuntu. The installation process will begin when you click the Install Now button.

Ubuntu needs about 4.5 GB to install, so add a few extra GB to allow for your files.



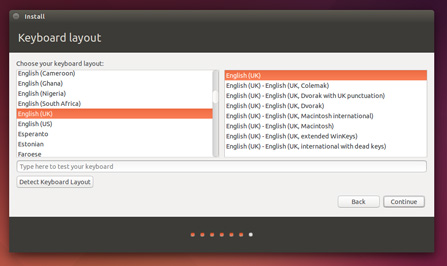
Step 6 : **Select your location**

If you are connected to the internet, this should be done automatically. Check your location is correct and click **’Forward’** to proceed. If you’re unsure of your time zone, type the name of the town you’re in or click on the map and we’ll help you find it.

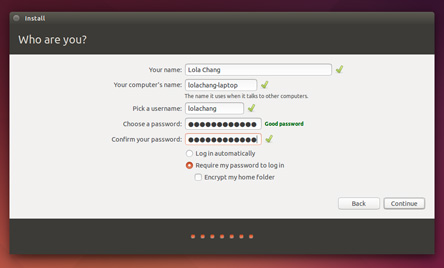


Step 7: **Select your preferred keyboard layout**

Click on the language option you need. If you’re not sure, click the **’Detect Keyboard Layout’** button for help.



Step 8: **Enter your login and password details.**

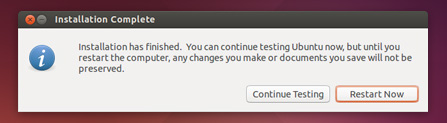


Step 9: **Learn more about Ubuntu while the system installs…**



Step 10: **That’s it.**

All that’s left is to restart your computer and start enjoying Ubuntu!



**Result:** Linux Ubuntu is successfully installed in computer system.

**Outcome:** Students will learn about the steps to install Linux operating system.

**EXPERIMENT - 02**

**Aim:** Implementation of CPU First Come First Serve Scheduling algorithm to find turnaround time and waiting time.

**Objective:** To write the program to implement CPU & scheduling algorithm for first come first serve scheduling.

**First Come First Serve Scheduling:**

### ALGORITHM:

1. Start the program.
2. Get the number of processes and their burst time.
3. Initialize the waiting time for process 1 and 0.

4. Process for(i=2;i<=n;i++),wt.p[i]=p[i-1]+bt.p[i-1].

1. The waiting time of all the processes is summed then average value time is calculated.
2. The waiting time of each process and average times are displayed.
3. Stop the program.

**PROGRAM**

#include <iostream>

using namespace std;

void calculation(int arv[], int bt[], int process)

{

int ct[20] = {0};

for (int i = 1; i <= process; i++)

{

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

}

cout << "the compile time of given process is :->[";

for (int i = 1; i <= process; i++)

{

cout << ct[i] << " ";

}

cout << "]" << endl;

cout << "the turn around time of given process is :->[";

int tat[20] = {0};

for (int i = 1; i <= process; i++)

{

tat[i] = ct[i] - arv[i];

cout << ct[i] - arv[i] << " ";

}

cout << "]" << endl;

cout << "the waiting time of given process is :->[";

for (int i = 1; i <= process; i++)

{

cout << tat[i] - bt[i] << " ";

}

cout << "]" << endl;

}

void compileTime(int bt[], int \*ct, int process)

{

for (int i = 1; i <= process; i++)

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

cout << "the compile time of given process is :->[";

for (int i = 1; i <= process; i++)

{

cout << ct[i] << " ";

}

cout << "]" << endl;

}

void turnAroundTime(int arv[], int \*ct, int \*tat, int process)

{

cout << "the turn around time of given process is :->[";

for (int i = 1; i <= process; i++)

{

tat[i] = ct[i] - arv[i];

cout << ct[i] - arv[i] << " ";

}

cout << "]" << endl;

}

void waitingTime(int bt[], int \*tat, int process)

{

cout << "the waiting time of given process is :->[";

for (int i = 1; i <= process; i++)

{

cout << tat[i] - bt[i] << " ";

}

cout << "]" << endl;

}

int main()

{

int process,j=1;

cout << "enter the number of process : ";

cin >> process;

int arv[20],bt[20];;

int ct[20] = {0}, tat[20] = {0};

cout << "enter the Arrival time of the process : ";

for (int i = 1; i <= process\*2; i++)

{

if(process-i>=0)

cin >> arv[i];

if(process-i==0)

cout << "enter the Burst time of process : ";

if(process-i<0)

cin>>bt[i-process];

}

compileTime(bt, ct, process);

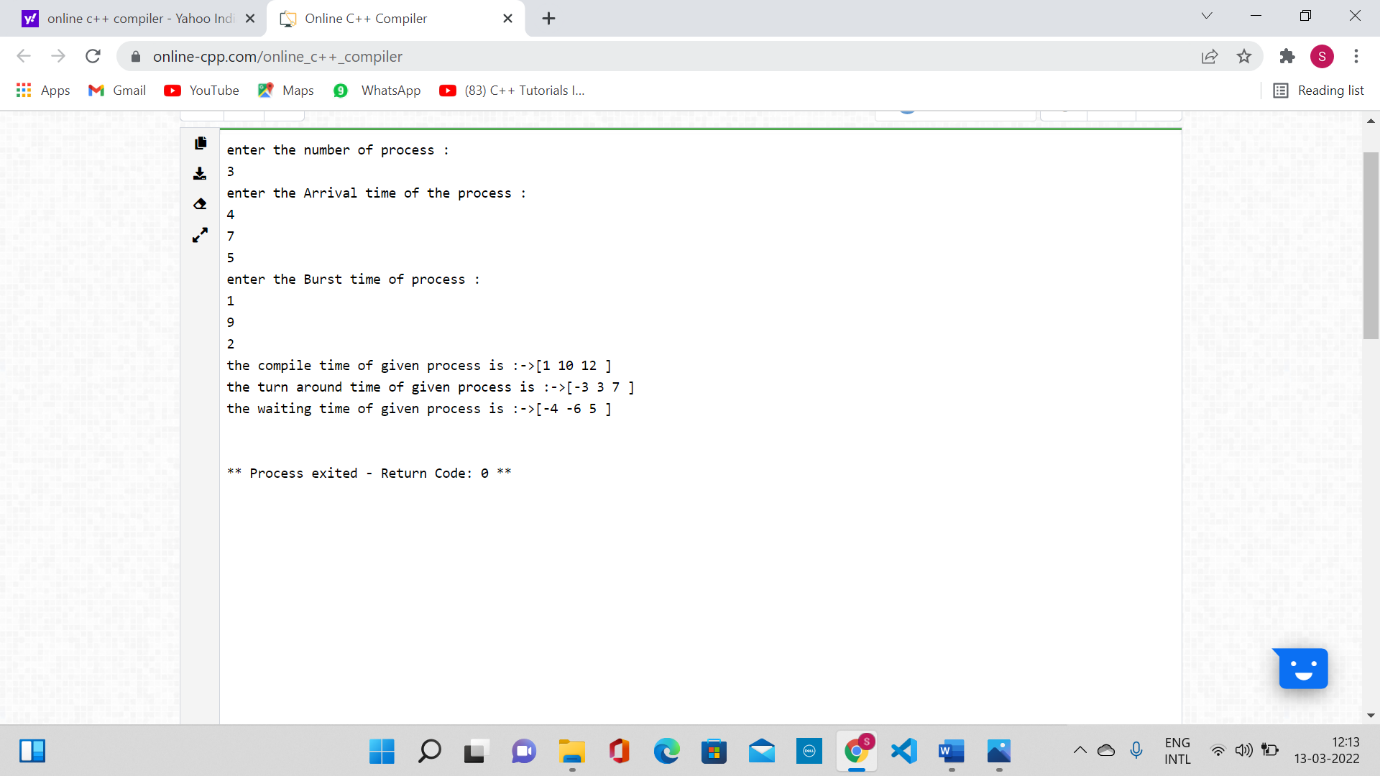
turnAroundTime(arv,ct,tat,process);

waitingTime(bt,tat,process);

return 0;

}

**OUTPUT:**



**Result**: Thus the FCFS process scheduling program was executed and verified successfully.

**Outcome:** Students will learnt about the execution of FCFS Scheduling program.

**EXPERIMENT - 03**

**Aim**: Implementation of CPU Shortest Job First Scheduling algorithm to find turnaround time and waiting time.

**Objective:** To write the program to implement CPU & scheduling algorithm for shortest job first scheduling.

**Shortest Job First Scheduling :**

**ALGORITHM:**

1. Start the program. Get the number of processes and their burst time.

2. Initialize the waiting time for process 1 as 0.

3. The processes are stored according to their burst time.

4. The waiting time for the processes are calculated a follows: for(i=2;i<=n;i++).wt.p[i]=p[i=1]+bt.p[i-1].

5. The waiting time of all the processes summed and then the average time is calculate

6. The waiting time of each processes and average time are displayed.

7. Stop the program.

**PROGRAM:**

#include <iostream>

using namespace std;

int mat[10][6];

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void arrangeArrival(int num, int mat[][6])

{

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

for (int j = 0; j < num - i - 1; j++) {

if (mat[j][1] > mat[j + 1][1]) {

for (int k = 0; k < 5; k++) {

swap(mat[j][k], mat[j + 1][k]);

}

}

}

}

}

void completionTime(int num, int mat[][6])

{

int temp, val;

mat[0][3] = mat[0][1] + mat[0][2];

mat[0][5] = mat[0][3] - mat[0][1];

mat[0][4] = mat[0][5] - mat[0][2];

for (int i = 1; i < num; i++) {

temp = mat[i - 1][3];

int low = mat[i][2];

for (int j = i; j < num; j++) {

if (temp >= mat[j][1] && low >= mat[j][2]) {

low = mat[j][2];

val = j;

}

}

mat[val][3] = temp + mat[val][2];

mat[val][5] = mat[val][3] - mat[val][1];

mat[val][4] = mat[val][5] - mat[val][2];

for (int k = 0; k < 6; k++) {

swap(mat[val][k], mat[i][k]);

}

}

}

int main()

{

int num, temp;

cout << "Enter number of Process: ";

cin >> num;

cout << "...Enter the process ID...\n";

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

cout << "...Process " << i + 1 << "...\n";

cout << "Enter Process Id: ";

cin >> mat[i][0];

cout << "Enter Arrival Time: ";

cin >> mat[i][1];

cout << "Enter Burst Time: ";

cin >> mat[i][2];

}

cout << "Before Arrange...\n";

cout << "Process ID\tArrival Time\tBurst Time\n";

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

cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"

<< mat[i][2] << "\n";

}

arrangeArrival(num, mat);

completionTime(num, mat);

cout << "Final Result...\n";

cout << "Process ID\tArrival Time\tBurst Time\tWaiting "

"Time\tTurnaround Time\n";

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

cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"

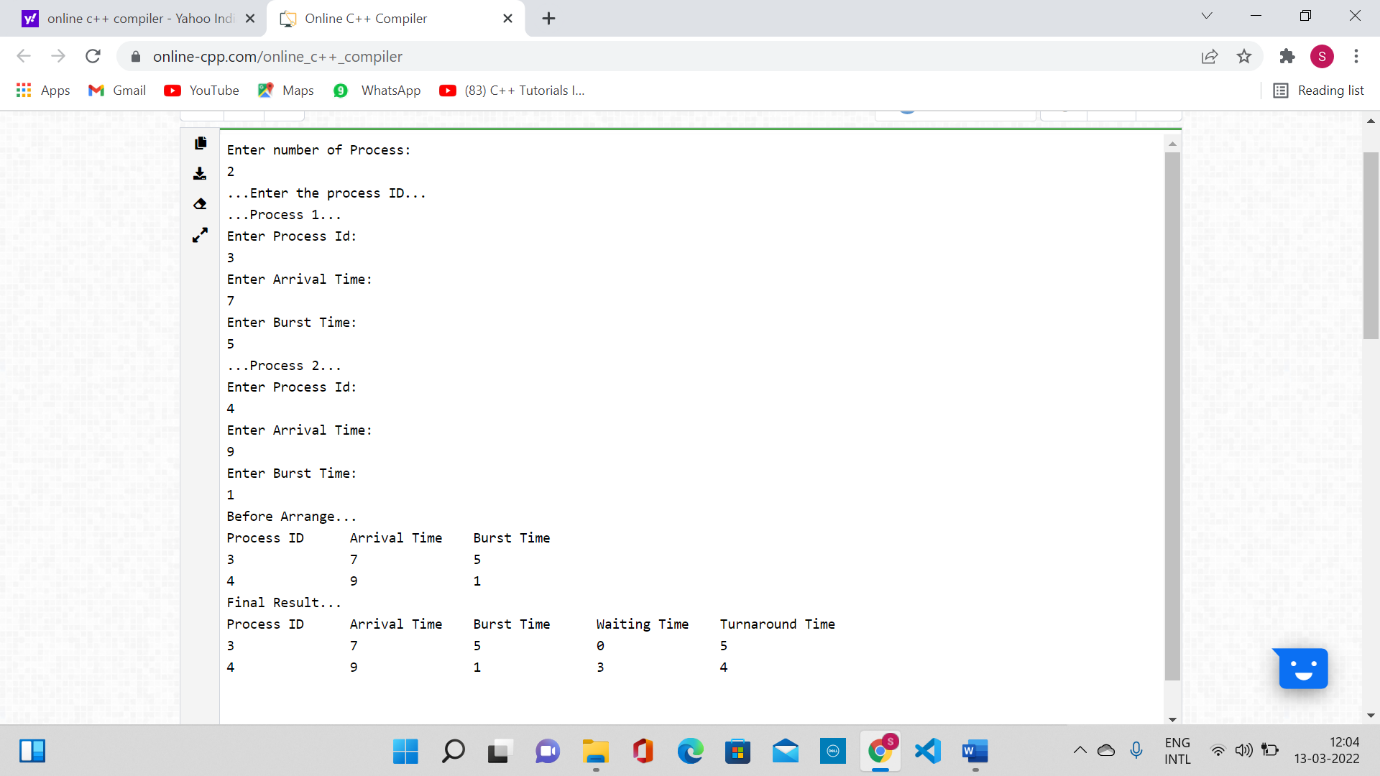
<< mat[i][2] << "\t\t" << mat[i][4] << "\t\t"

<< mat[i][5] << "\n";

}

}

**OUTPUT:**



**Result:** Thus the SJF program was executed and verified successfully.

**Outcome:** Students will learn about the execution of SJF Program.

**EXPERIMENT - 04**

**Aim**: Implementation of CPU Round Robin Scheduling algorithm to find turnaround time and waiting time.

**Objective:** To write the program to implement CPU & scheduling algorithm for Round Robin scheduling.

**Round Robin Scheduling :**

**ALGORITHM:**

1. Get the number of process and their burst time.

2. Initialize the array for Round Robin circular queue as ‘0’.

3. The burst time of each process is divided and the quotients are stored on the round Robin array.

4. According to the array value the waiting time for each process and the average time are calculated as line the other scheduling.

5. The waiting time for each process and average times are displayed.

6. Stop the program.

**PROGRAM:**

#include<iostream>

using namespace std;

void findWaitingTime(int processes[], int n,

int bt[], int wt[], int quantum)

{

int rem\_bt[n];

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

rem\_bt[i] = bt[i];

int t = 0; // Current time

while (1)

{

bool done = true;

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

{

if (rem\_bt[i] > 0)

{

done = false; // There is a pending process

if (rem\_bt[i] > quantum)

{

t += quantum;

rem\_bt[i] -= quantum;

}

else

{

t = t + rem\_bt[i];

wt[i] = t - bt[i];

rem\_bt[i] = 0;

}

}

}

if (done == true)

break;

}

}

// Function to calculate turn around time

void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[])

{

// calculating turnaround time by adding

// bt[i] + wt[i]

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

tat[i] = bt[i] + wt[i];

}

void findavgTime(int processes[], int n, int bt[],int quantum)

{

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

findWaitingTime(processes, n, bt, wt, quantum);

findTurnAroundTime(processes, n, bt, wt, tat);

cout << "Processes "<< " Burst time "

<< " Waiting time " << " Turn around time\n";

// Calculate total waiting time and total turn

// around time

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

cout << " " << i+1 << "\t\t" << bt[i] <<"\t "

<< wt[i] <<"\t\t " << tat[i] <<endl;

}

cout << "Average waiting time = "

<< (float)total\_wt / (float)n;

cout << "\nAverage turn around time = "

<< (float)total\_tat / (float)n;

}

int main()

{

int processes[] = { 1, 2, 3};

int n = sizeof processes / sizeof processes[0];

int burst\_time[] = {10, 5, 8};

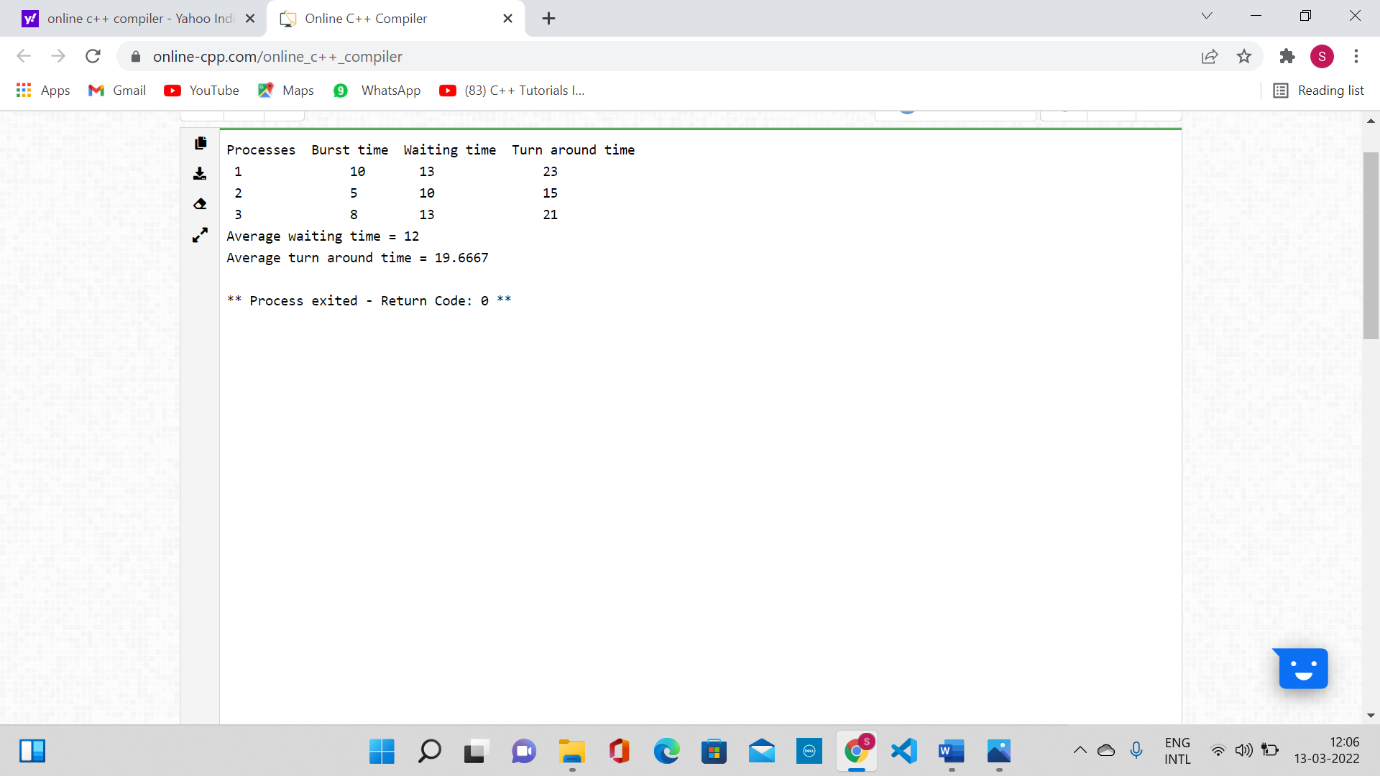
int quantum = 2;

findavgTime(processes, n, burst\_time, quantum);

return 0;

}

**OUTPUT:**



**Result:** Thus, the Round Robin scheduling program was executed and verified successfully.

**Outcome:** Students will learn about the execution of Round Robin Program.

**EXPERIMENT - 05**

**Aim**: Implementation of CPU Shortest Job First Scheduling algorithm(preemptive) to find turnaround time and waiting time.

**Objective:** To write the program to implement CPU & scheduling algorithm for shortest job first scheduling(preemptive).

**SHORTEST JOB FIRST SCHEDULING(PREEMPTIVE)**

**PROGRAM:**

#include <bits/stdc++.h>

using namespace std;

// structure for every process

struct Process

{

int pid; // Process ID

int bt; // Burst Time

int art; // Arrival Time

};

void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])

{

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

tat[i] = proc[i].bt + wt[i];

}

// waiting time of all process

void findWaitingTime(Process proc[], int n, int wt[])

{

int rt[n];

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

rt[i] = proc[i].bt;

int complete = 0, t = 0, minm = INT\_MAX;

int shortest = 0, finish\_time;

bool check = false;

while (complete != n)

{

for (int j = 0; j < n; j++)

{

if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0)

{

minm = rt[j];

shortest = j;

check = true;

}

}

if (check == false)

{

t++;

continue;

}

// decrementing the remaining time

rt[shortest]--;

minm = rt[shortest];

if (minm == 0)

minm = INT\_MAX;

// If a process gets completely

// executed

if (rt[shortest] == 0)

{

complete++;

check = false;

finish\_time = t + 1;

// Calculate waiting time

wt[shortest] = finish\_time -

proc[shortest].bt -

proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

// Increment time

t++;

}

}

// Function to calculate average time

void findavgTime(Process proc[], int n)

{

int wt[n], tat[n], total\_wt = 0,

total\_tat = 0;

// Function to find waiting time of all

// processes

findWaitingTime(proc, n, wt);

// Function to find turn around time for

// all processes

findTurnAroundTime(proc, n, wt, tat);

cout << "Processes "

<< " Burst time "

<< " Waiting time "

<< " Turn around time\n";

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t " << wt[i] << "\t\t " << tat[i] << endl;

}

cout << "\nAverage waiting time = " << (float)total\_wt / (float)n;

cout << "\nAverage turn around time = " << (float)total\_tat / (float)n;

}

// main function

int main()

{

Process proc[] = {{1, 5, 1}, {2, 3, 1}, {3, 6, 2}, {4, 5, 3}};

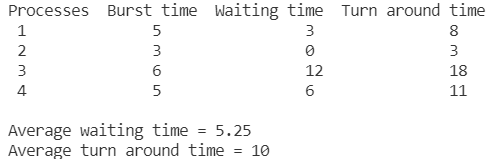
int n = sizeof(proc) / sizeof(proc[0]);

findavgTime(proc, n);

return 0;

}

**Output:**

****

**Result:** Thus, the SJF (preemptive) program was executed and verified successfully.

**Outcome:** Students will learn about the execution of SJF Program(preemptive).

**EXPERIMENT – 06**

**Aim**: Implementation of CPU Priority Scheduling algorithm to find turnaround time and waiting time.

**Objective:** To write the program to implement CPU & scheduling algorithm for priority scheduling.

**PRIORITY SCHEDULING**

**ALGORITHM:**

1. Start the program.

2. Read burst time, waiting time, turn the around time and priority.

3. Initialize the waiting time for process 1 and 0.

4. Based up on the priority process are arranged

5. The waiting time of all the processes is summed and then the average waiting time

6. The waiting time of each process and average waiting time are displayed based on the priority.

7. Stop the program.

**PROGRAM:**

// C++ implementation for Priority Scheduling with

// Different Arrival Time priority scheduling

/\*1. sort the processes according to arrival time

2. if arrival time is same the acc to priority

3. apply fcfs

\*/

#include <bits/stdc++.h>

using namespace std;

#define totalprocess 5

// Making a struct to hold the given input

struct process

{

int at, bt, pr, pno;

};

process proc[50];

/\*

Writing comparator function to sort according to priority if

arrival time is same

\*/

bool comp(process a, process b)

{

if (a.at == b.at)

{

return a.pr < b.pr;

}

else

{

return a.at < b.at;

}

}

// Using FCFS Algorithm to find Waiting time

void get\_wt\_time(int wt[])

{

// declaring service array that stores cumulative burst time

int service[50];

// Initialising initial elements of the arrays

service[0] = proc[0].at;

wt[0] = 0;

for (int i = 1; i < totalprocess; i++)

{

service[i] = proc[i - 1].bt + service[i - 1];

wt[i] = service[i] - proc[i].at;

// If waiting time is negative, change it into zero

if (wt[i] < 0)

{

wt[i] = 0;

}

}

}

void get\_tat\_time(int tat[], int wt[])

{

// Filling turnaroundtime array

for (int i = 0; i < totalprocess; i++)

{

tat[i] = proc[i].bt + wt[i];

}

}

void findgc()

{

// Declare waiting time and turnaround time array

int wt[50], tat[50];

double wavg = 0, tavg = 0;

// Function call to find waiting time array

get\_wt\_time(wt);

// Function call to find turnaround time

get\_tat\_time(tat, wt);

int stime[50], ctime[50];

stime[0] = proc[0].at;

ctime[0] = stime[0] + tat[0];

// calculating starting and ending time

for (int i = 1; i < totalprocess; i++)

{

stime[i] = ctime[i - 1];

ctime[i] = stime[i] + tat[i] - wt[i];

}

cout << "Process\_no\tStart\_time\tComplete\_time\tTurn\_Around\_Time\tWaiting\_Time" << endl;

// display the process details

for (int i = 0; i < totalprocess; i++)

{

wavg += wt[i];

tavg += tat[i];

cout << proc[i].pno << "\t\t" << stime[i] << "\t\t" << ctime[i] << "\t\t" << tat[i] << "\t\t\t" << wt[i] << endl;

}

// display the average waiting time

// and average turn around time

cout << "Average waiting time is : ";

cout << wavg / (float)totalprocess << endl;

cout << "average turnaround time : ";

cout << tavg / (float)totalprocess << endl;

}

int main()

{

int arrivaltime[] = {1, 2, 3, 4, 5};

int bursttime[] = {3, 5, 1, 7, 4};

int priority[] = {3, 4, 1, 7, 8};

for (int i = 0; i < totalprocess; i++)

{

proc[i].at = arrivaltime[i];

proc[i].bt = bursttime[i];

proc[i].pr = priority[i];

proc[i].pno = i + 1;

}

// Using inbuilt sort function

sort(proc, proc + totalprocess, comp);

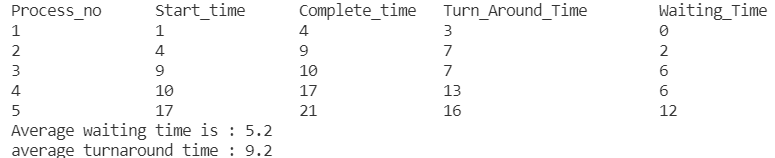
// Calling function findgc for finding Gantt Chart

findgc();

return 0;

}

**Output:**

****

**Result:** Thus, the priority scheduling program was executed and verified successfully.

**Outcome:** Students will learn about the execution of Priority Scheduling Program.

**EXPERIMENT – 07**

**Aim**: Virtualization, Installation of Virtual Machine Software and installation of Operating System on Virtual Machine.

**Objectives:** To study about virtualization, installation of Virtual Machine and Installation of Linux operating system using Virtual Machine.

**Introduction**  
VirtualBox allows you to run an entire operating system inside another operating system. Please be aware that you should have a minimum of 512 MB of RAM. 1 GB of RAM or more is recommended.

**Comparison to Dual-Boot**  
Many websites (including the one you're reading) have tutorials on setting up dual-boots between Windows and Ubuntu. A dual-boot allows you, at boot time, to decide which operating system you want to use. Installing Ubuntu on a virtual machine inside of Windows has a lot advantages over a dual-boot (but also a few disadvantages).

Advantages of virtual installation

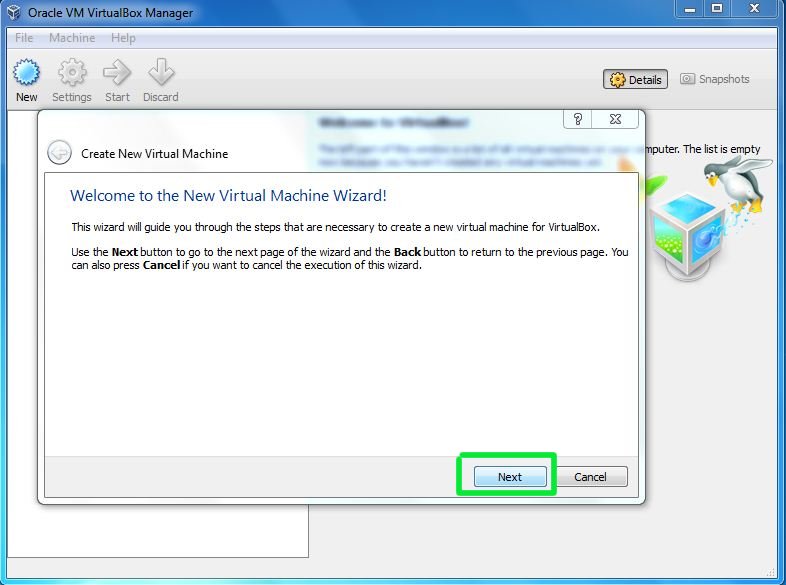
* The size of the installation doesn't have to be predetermined. It can be a dynamically resized virtual hard drive.
* You do not need to reboot in order to switch between Ubuntu and Windows.
* The virtual machine will use your Windows internet connection, so you don't have to worry about Ubuntu not detecting your wireless card, if you have one.
* The virtual machine will set up its own video configuration, so you don't have to worry about installing proprietary graphics drivers to get a reasonable screen resolution.
* You always have Windows to fall back on in case there are any problems. All you have to do is press the right Control key instead of rebooting your entire computer.
* For troubleshooting purposes, you can easily take screenshots of any part of Ubuntu (including the boot menu or the login screen).
* It's low commitment. If you later decide you don't like Ubuntu, all you have to do is delete the virtual hard drive and uninstall VirtualBox.

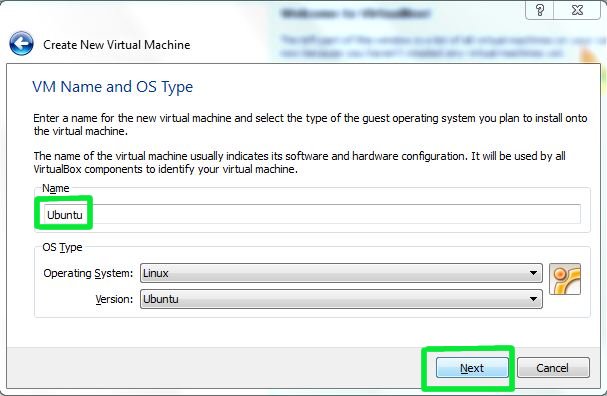
Disadvantages of virtual installation

* In order to get any kind of decent performance, you need at least 512 MB of RAM, because you are running an entire operating system (Ubuntu) inside another entire operating system (Windows). The more memory, the better. I would recommend at least 1 GB of RAM.
* Even though the low commitment factor can seem like an advantage at first, if you later decide you want to switch to Ubuntu and ditch Windows completely, you cannot simply delete your Windows partition. You would have to find some way to migrate out your settings from the virtual machine and then install Ubuntu over Windows outside the virtual machine.
* Every time you want to use Ubuntu, you have to wait for two boot times (the time it takes to boot Windows, and then the time it takes to boot Ubuntu within Windows).

**Installation Process**  
Step 1: The first thing you have to do is to install oracle VirtualBox. Install it the same way you would any normal Windows program. Then get Ubuntu disk image (.iso file).

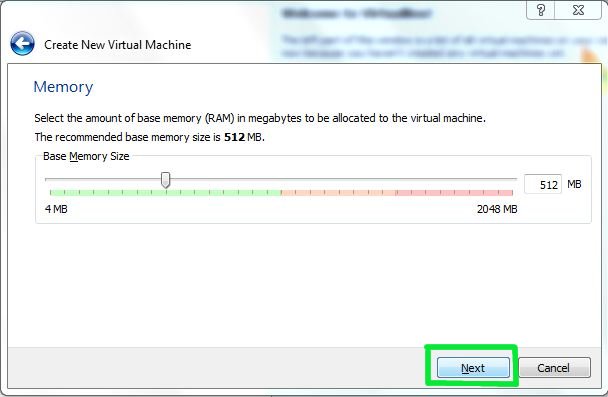
Step 2: After you launch VirtualBox from the Windows Start menu, click on **New** to create a new virtual machine. When the New Virtual Machine Wizard appears, click **Next**.



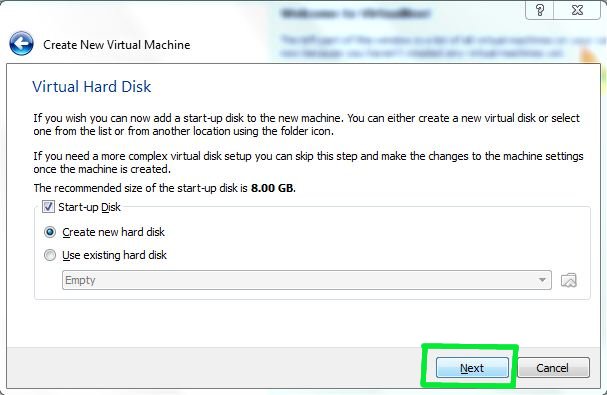
Step 3: You can call the machine whatever you want. If you're installing Ubuntu, it makes sense to call it **Ubuntu**, I guess. You should also specify that the operating system is **Linux**.  


Step 4: VirtualBox will try to guess how much of your memory (or RAM) to allocate for the virtual machine. If you have 1 GB or less of RAM, I would advise you stick with the recommendation. If, however, you have over 1 GB, about a quarter your RAM or less should be fine. For example, if you have 2 GB of RAM, 512 MB is fine to allocate. If you have 4 GB of RAM, 1 GB is fine to allocate. If you have no idea what RAM is or how much of it you have, just go with the default.

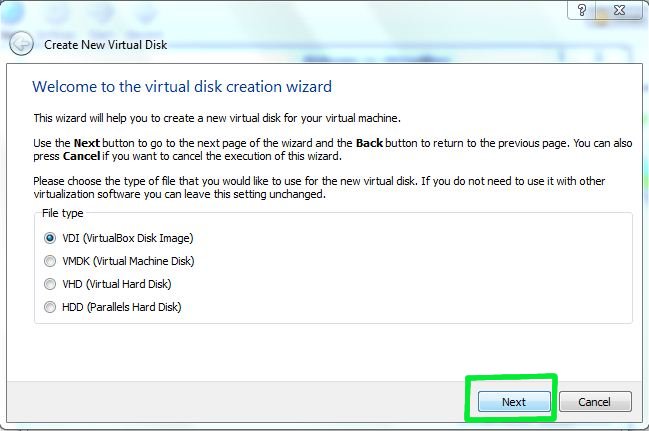
Click **Next**.



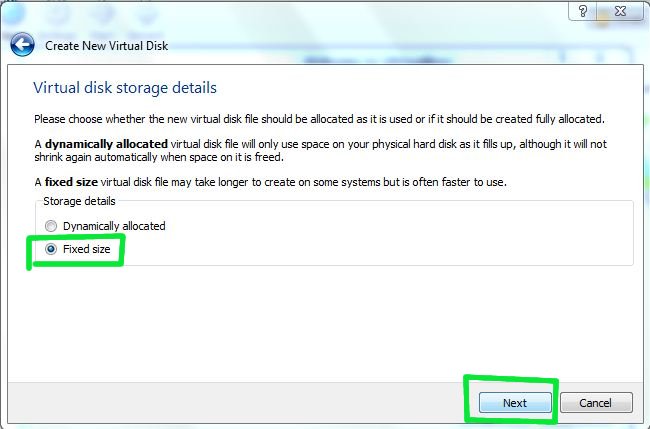
Step 5: If this is your first time using VirtualBox (which it probably is if you need a tutorial on how to use it), then you do want to Create new hard disk and then click **Next**.



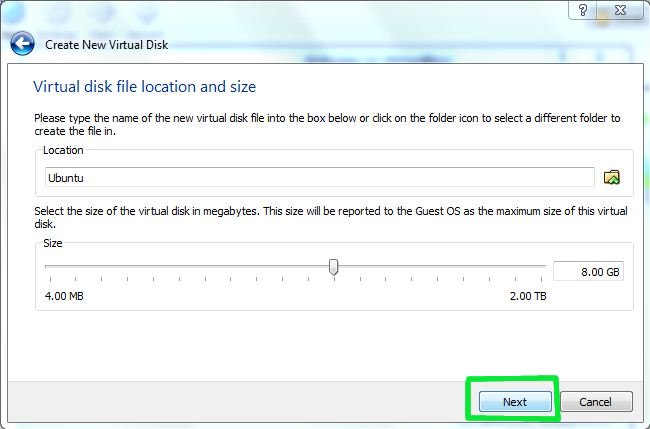
Step 6: Click **Next** again.



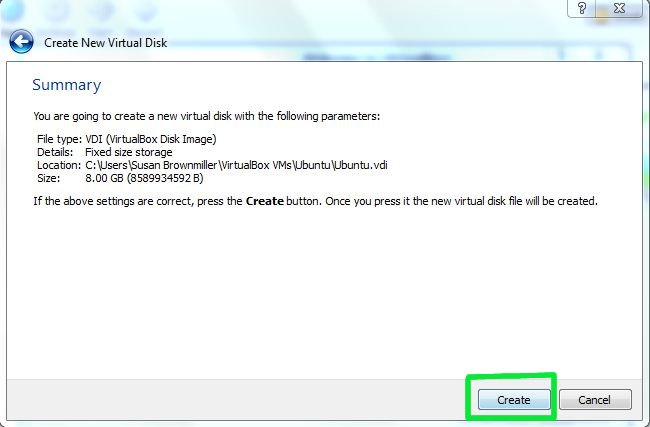
Step 7: Theoretically, a dynamically expanding virtual hard drive is best, because it'll take up only what you actually use. I have come upon weird situations, though, when installing new software in a virtualized Ubuntu, in which the virtual hard drive just fills up instead of expanding. So I would actually recommend picking a **Fixed-size storage**.

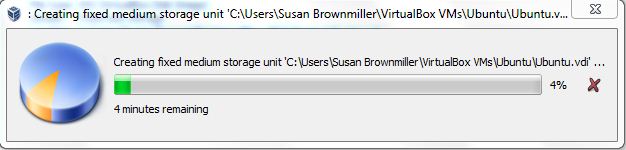


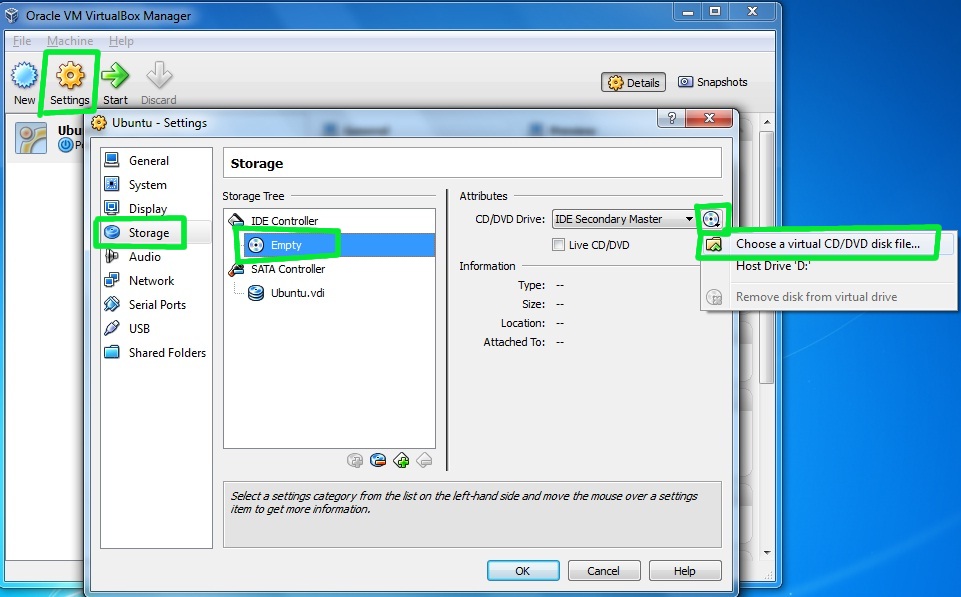
Step 8: Ubuntu's default installation is less than 3 GB. If you plan on adding software or downloading large files in your virtualized Ubuntu, you should tack on some buffer.



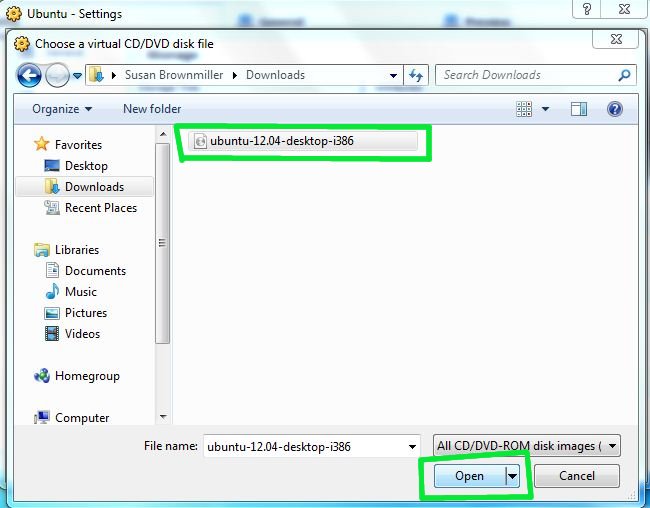
Step 9: Click **Create** and wait for the virtual hard drive to be created. This is actually just a very large file that lives inside of your Windows installation.





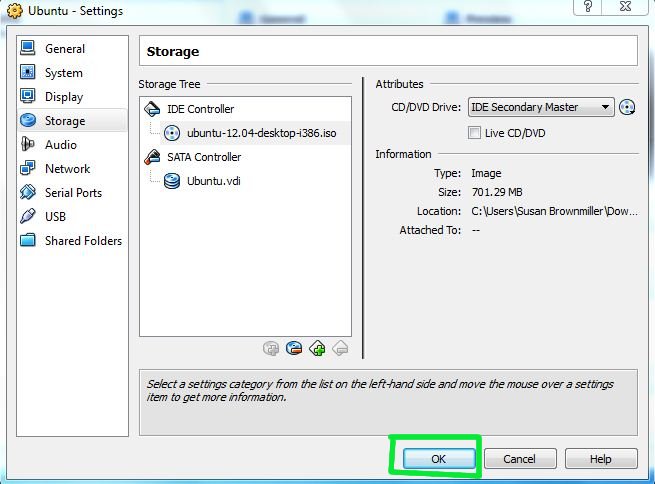
Step 10: The next thing to do to make the (currently blank) virtual hard drive useful is to add the downloaded Ubuntu disk image (the .iso) boot on your virtual machine. Click on **Settings** and **Storage**. Then, under CD/DVD Device, next to Empty, you'll see a little folder icon. Click that.  


Step 11:Select the Ubuntu .iso you downloaded earlier.

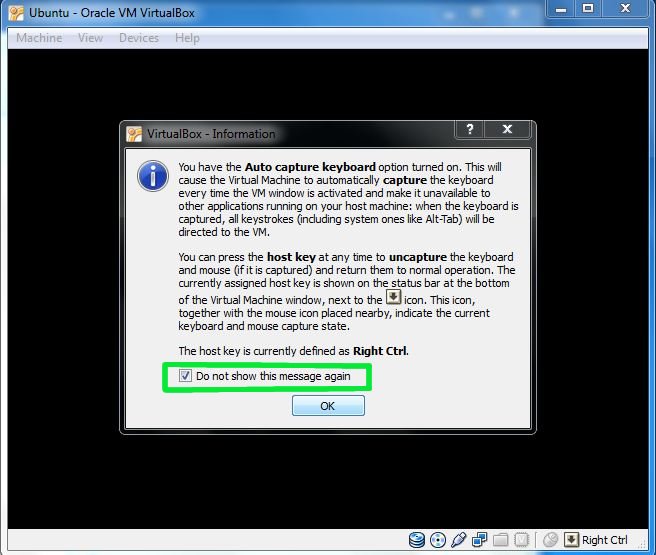


Step 12: Once you've selected it, click **OK**.

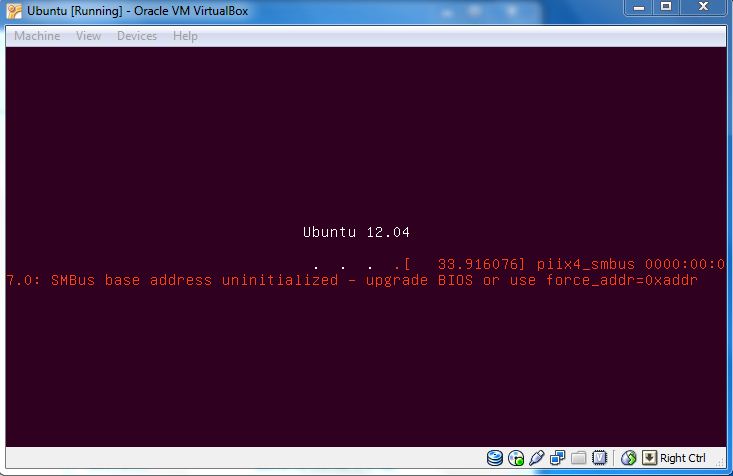
Then double-click your virtual machine to start it up.



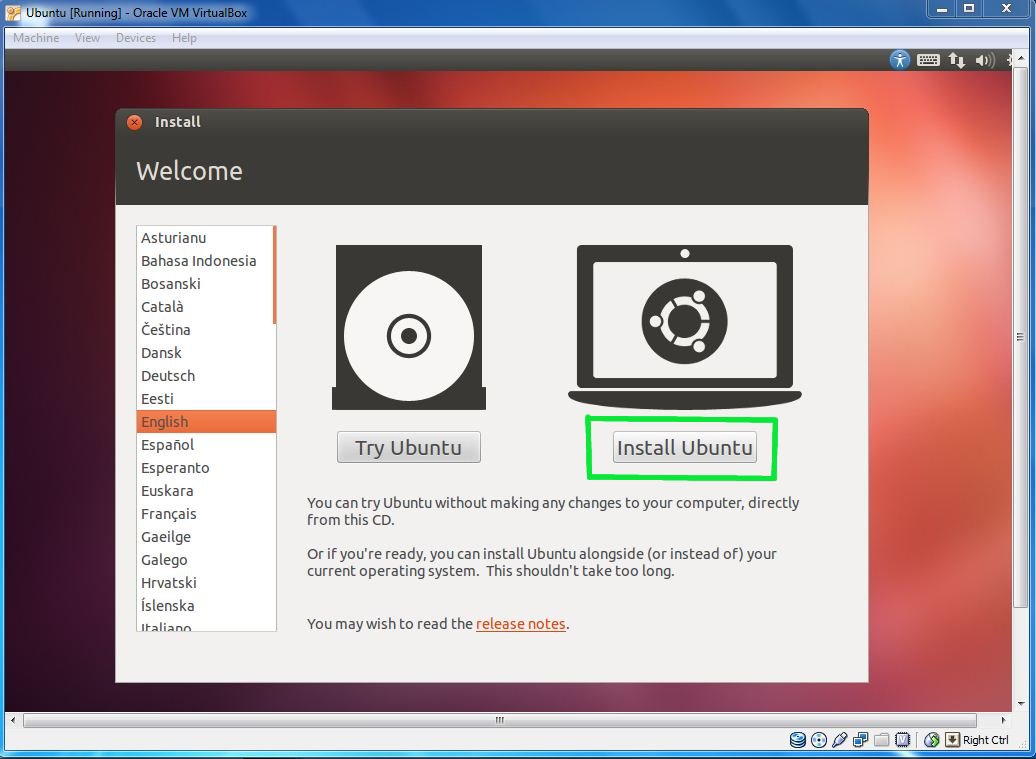
Step 13: You may get a bunch of random warnings/instructions about how to operate the guest operating system within VirtualBox. Read those, and then you may also want to mark not to see those again.



Step 14: Wait for Ubuntu to boot up.



Step 15: Once it's started up, just follow the [regular installation procedure](http://www.psychocats.net/ubuntu/installing) as if you were installing.



**Result:** Virtual Machine is installed successfully and Linux operating is also successfully installed using virtual machine.

**Outcome:** Students will learn about the virtualization and installing of operating system using virtual machine.