**Lab Report-01**

**Problem** : Write a program to implement First Come First Served algorithm.

**Theory :** First Come First Served (FCFS) is a type of scheduling algorithm used by operating systems and networks to efficiently and automatically execute queued tasks, processes and requests by the order of their arrival.

**Code:**

package com.mycompany.operating\_sustem;

import java.util.\*;

public class FCFS {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter no of process: ");

int n = sc.nextInt();

int pid[] = new int[n];

int ar[] = new int[n];

int bt[] = new int[n];

int ct[] = new int[n];

int ta[] = new int[n];

int wt[] = new int[n];

int temp;

float avgwt=0,avgta=0;

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

System.out.println("Enter process " + (i+1) + " arrival time: ");

ar[i] = sc.nextInt();

System.out.println("Enter process " + (i+1) + " brust time: ");

bt[i] = sc.nextInt();

pid[i] = i+1; }

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

{

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

{

if( ar[j] > ar[j+1] )

{

temp = ar[j];

ar[j] = ar[j+1];

ar[j+1] = temp;

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

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

if( i == 0){

ct[i] = ar[i] + bt[i];}

else{

if( ar[i] > ct[i-1]){

ct[i] = ar[i] + bt[i];}

else

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

}

ta[i] = ct[i] - ar[i] ;

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

avgwt += wt[i] ;

avgta += ta[i] ;

}

System.out.println("\npid arrival brust complete turn waiting");

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

{

System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + a[i] + "\t" + wt[i] ) ;

}

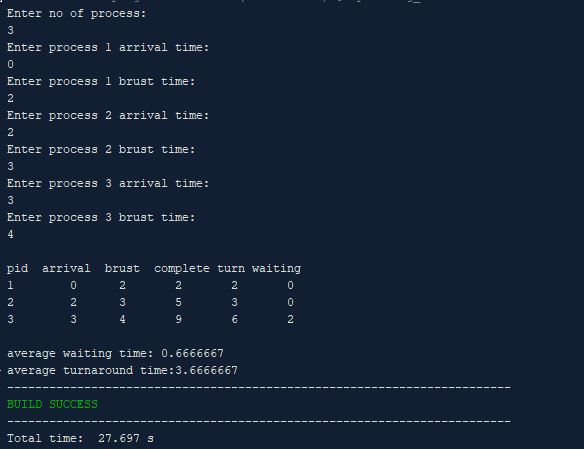
sc.close();

System.out.println("\naverage waiting time: "+ (avgwt/n));

System.out.println("average turnaround time:"+(avgta/n));

}

}



**Lab Report-02**

**Problem** : Write a program to implement Shortest Job First algorithm.

**Theory :** Shortest job first is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Shortest job first can be either preemptive or non-preemptive.

**Code:**

package com.mycompany.operating\_sustem;

import java.util.\*;

public class SJF {

public static void main (String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println ("enter no of process:");

int n= sc.nextInt();

int pid[] = new int[n];

int at[] = new int[n];

int bt[] = new int[n];

int ct[] = new int[n];

int ta[] = new int[n];

int wt[] = new int[n];

int f[] = new int[n];

int k[]= new int[n]; // it is also stores brust time

int i, st=0, tot=0;

float avgwt=0, avgta=0;

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

{

pid[i]= i+1;

System.out.println ("enter process " +(i+1)+ " arrival time:");

at[i]= sc.nextInt();

System.out.println("enter process " +(i+1)+ " burst time:");

bt[i]= sc.nextInt();

k[i]= bt[i];

f[i]= 0;

}

while(true){

int min=99,c=n;

if (tot==n)

break;

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

{

if ((at[i]<=st) && (f[i]==0) && (bt[i]<min))

{

min=bt[i];

c=i;

}

}

if (c==n)

st++;

else

{

bt[c]--;

st++;

if (bt[c]==0)

{

ct[c]= st;

f[c]=1;

tot++;

}

}

}

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

{

ta[i] = ct[i] - at[i];

wt[i] = ta[i] - k[i];

avgwt+= wt[i];

avgta+= ta[i];

}

System.out.println("pid arrival burst complete turn waiting");

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

{

System.out.println(pid[i] +"\t"+ at[i]+"\t"+ k[i] +"\t"+ ct[i] +"\t"+ ta[i] +"\t"+ wt[i]);

}

System.out.println("\naverage tat is "+ (float)(avgta/n));

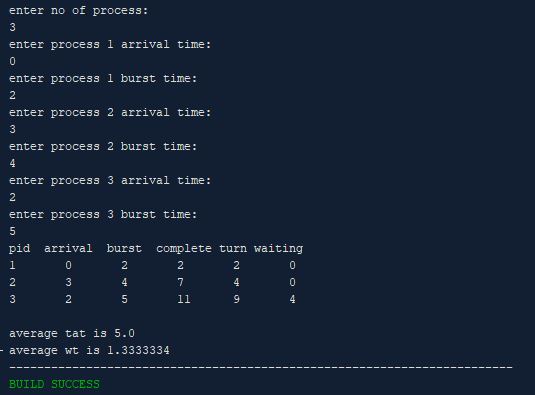
System.out.println("average wt is "+ (float)(avgwt/n));

sc.close();

}

}

**Output :**

****

**Lab Report-03**

**Problem** : Write a program to implement Priority Scheduling algorithm.

**Theory :** In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

**Code:**

package com.mycompany.operating\_sustem;

import java.util.\*;

public class PriorityScheduler{

public static void main(String args[]) {

Scanner s = new Scanner(System.in);

System.out.print("Enter the number of process : ");

int n = s.nextInt();

int i,x;

int p[] = new int[20];

int pp[] = new int[20];

int bt [] = new int[20];

int w [] = new int[20];

int t[] = new int[20];

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

{

System.out.print("Enter process " + (i+1) + " burst time: ");

bt[i] = s.nextInt();

System.out.print("Enter process " + (i+1) +" priorities: ");

pp[i] = s.nextInt();

p[i]=i+1;

}

for(i=0;i<n-1;i++)

{

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

{

if(pp[i]>pp[j])

{

x=pp[i];

pp[i]=pp[j];

pp[j]=x;

x=bt[i];

bt[i]=bt[j];

bt[j]=x;

x=p[i];

p[i]=p[j];

p[j]=x;

}

}

}

w[0]=0;

float awt=0;

t[0]=bt[0];

float atat=t[0];

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

{

w[i]=t[i-1];

awt+=w[i];

t[i]=w[i]+bt[i];

atat+=t[i];

}

w[0]=0;

awt=0;

t[0]=bt[0];

atat=t[0];

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

{

w[i]=t[i-1];

awt+=w[i];

t[i]=w[i]+bt[i];

atat+=t[i];

}

System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time Priority \n");

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

System.out.print("\n "+p[i]+"\t\t "+bt[i]+"\t\t "+w[i]+"\t\t "+t[i]+"\t\t "+pp[i]+"\n");

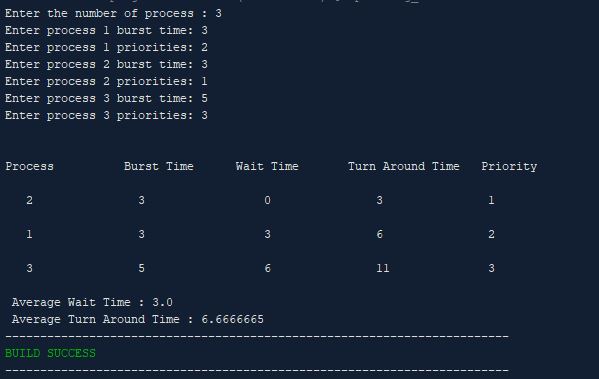
System.out.print("\n Average Wait Time : "+awt/n);

System.out.print("\n Average Turn Around Time : "+atat/n);

}

}

**Output:**



**Lab Report-04**

**Problem** : Write a program to implement Round Robin Scheduling algorithm.

**Theory :** The Round-robin scheduling algorithm is a kind of preemptive First come, First Serve CPU Scheduling algorithm where each process in the ready state gets the CPU for a fixed time in a cyclic way (turn by turn). It is the oldest scheduling algorithm, which is mainly used for multitasking.

**Code:**

package com.mycompany.operating\_sustem;

import java.util.Scanner;

public class RoundRobinS {

public static void main(String args[])

{

int n,i,qt,count=0,temp,sq =0,bt[],wt[],tat[],rem\_bt[];

float awt=0,atat=0;

bt = new int[10];

wt = new int[10];

tat = new int[10];

rem\_bt = new int[10];

Scanner s=new Scanner(System.in);

System.out.print("Enter the number of process = ");

n = s.nextInt();

System.out.print("Enter the burst time of the process\n");

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

{

System.out.print("P"+i+" = ");

bt[i] = s.nextInt();

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

}

System.out.print("Enter the quantum time: ");

qt = s.nextInt();

while(true)

{

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

{

temp = qt;

if(rem\_bt[i] == 0)

{

count++;

continue;

}

if(rem\_bt[i]>qt)

rem\_bt[i]= rem\_bt[i] - qt;

else

if(rem\_bt[i]>=0)

{

temp = rem\_bt[i];

rem\_bt[i] = 0;

}

sq = sq + temp;

tat[i] = sq;

}

if(n == count)

break;

}

System.out.print("\nProcess Burst Time Turnaround Time Waiting Time\n");

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

{

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

awt=awt+wt[i];

atat=atat+tat[i];

System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t "+wt[i]+"\n");

}

awt=awt/n;

atat=atat/n;

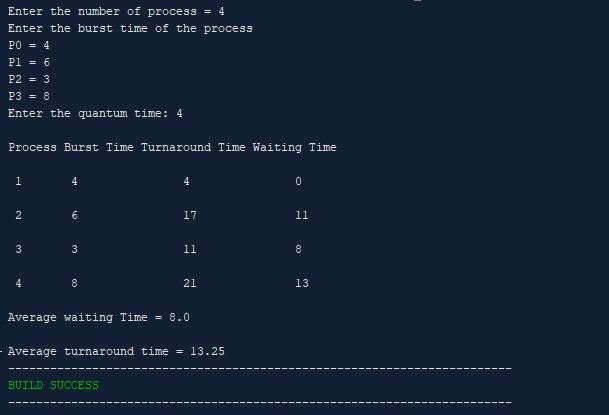
System.out.println("\nAverage waiting Time = "+awt+"\n");

System.out.println("Average turnaround time = "+atat);

}

}

**Output:**

****

**Lab Report-05**

# Problem : How to install Ubuntu on VirtualBox.

# Theory :

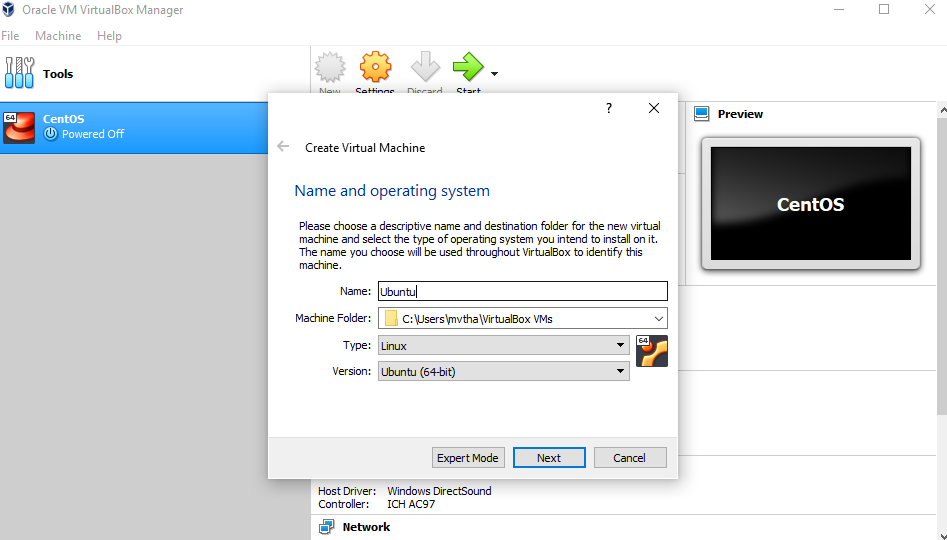
# Virtual Box: Oracle VM VirtualBox is a cross-platform virtualization application developed by the Oracle Corporation. It allows users to install operating systems on virtual hard disks such as Windows, macOS, Solaris and Linux.

# Ubuntu Install steps :

# First, open VirtualBox, then click "New" to create a virtual machine.

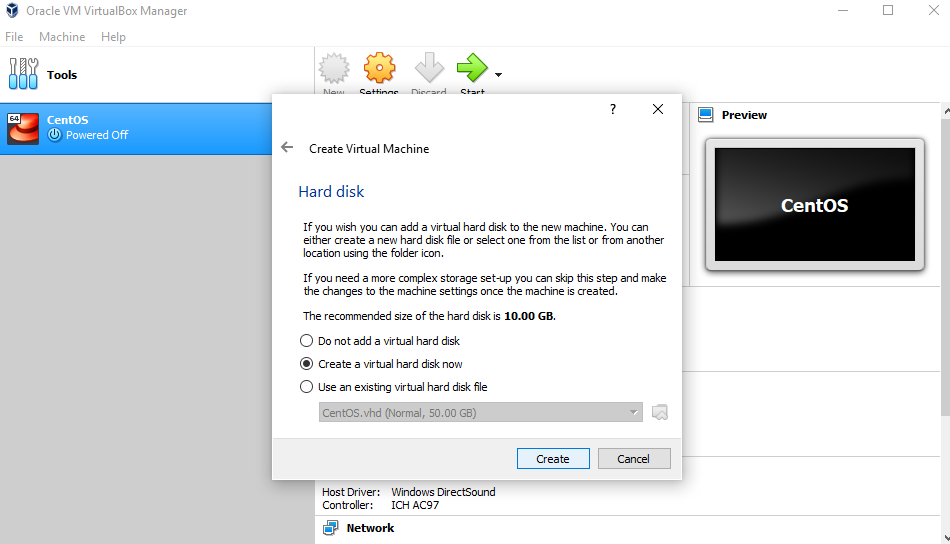
# 

Enter "Ubuntu" as the name, select "Linux" as the type, and select Ubuntu (64-bit) as the version.

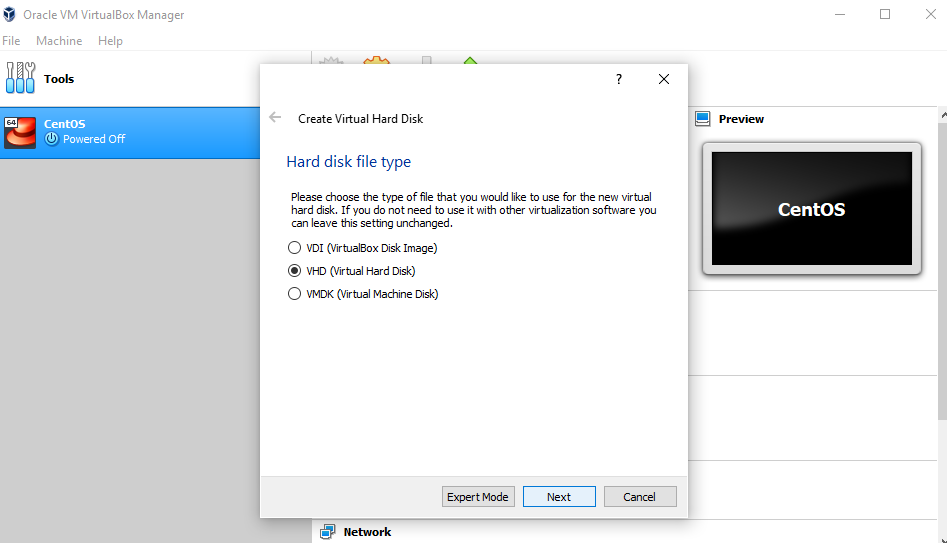
****

Select any amount of memory you wish, but don't add more than 50 percent of your total RAM.

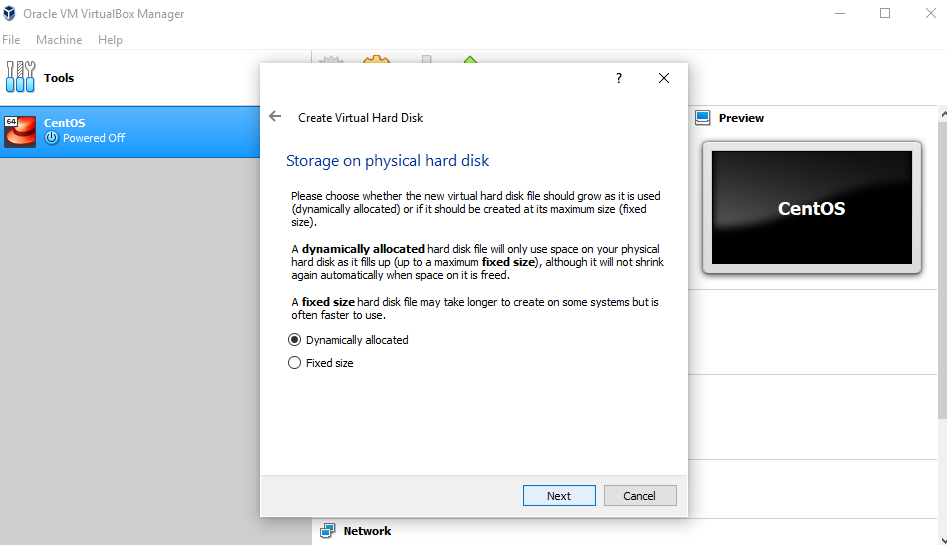
Check the "Create a virtual hard disk now" option so we can later define our Ubuntu OS virtual hard disk size.



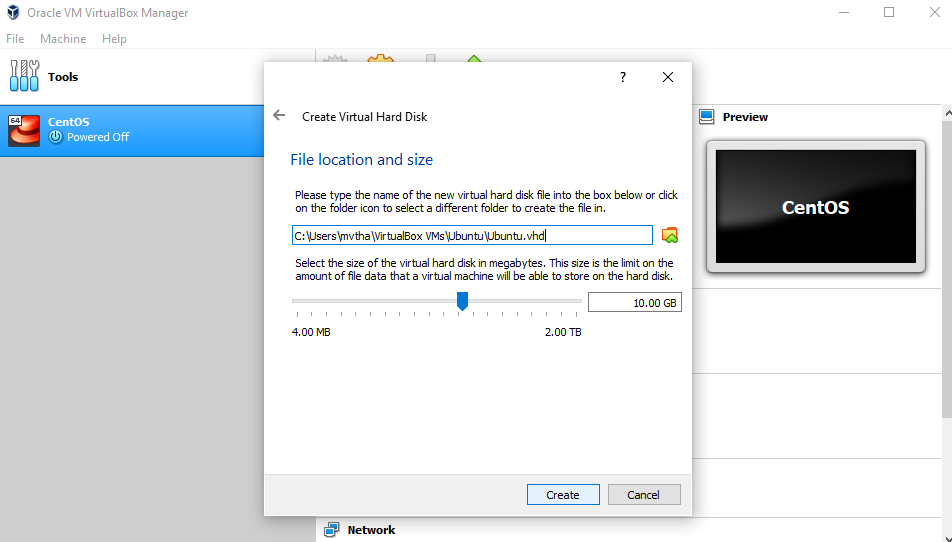
Now, we want to select "VHD (Virtual Hard Disk)".

****

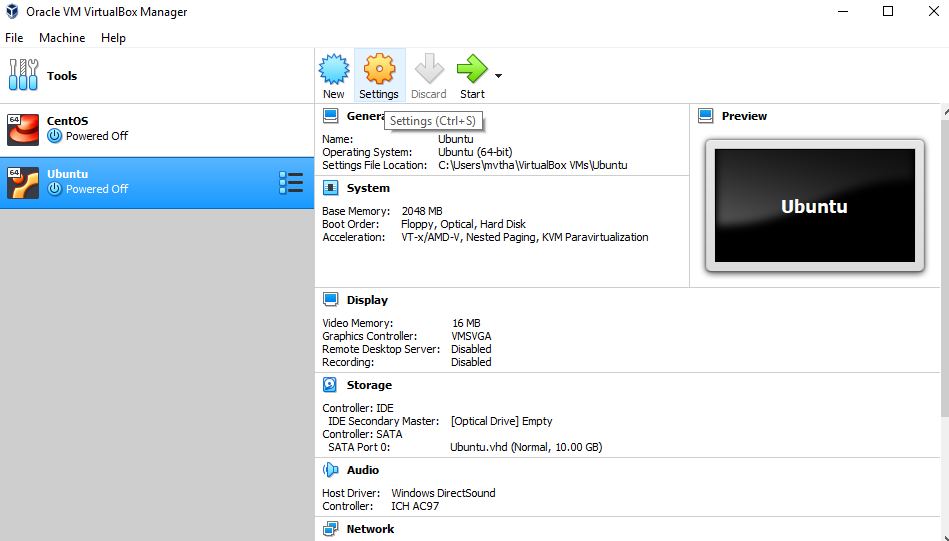
Next, we'll dynamically allocate storage on our physical hard disk.

****

We want to specify our Ubuntu OS's size. The recommended size is 10 GB, but you can increase the size if you wish.

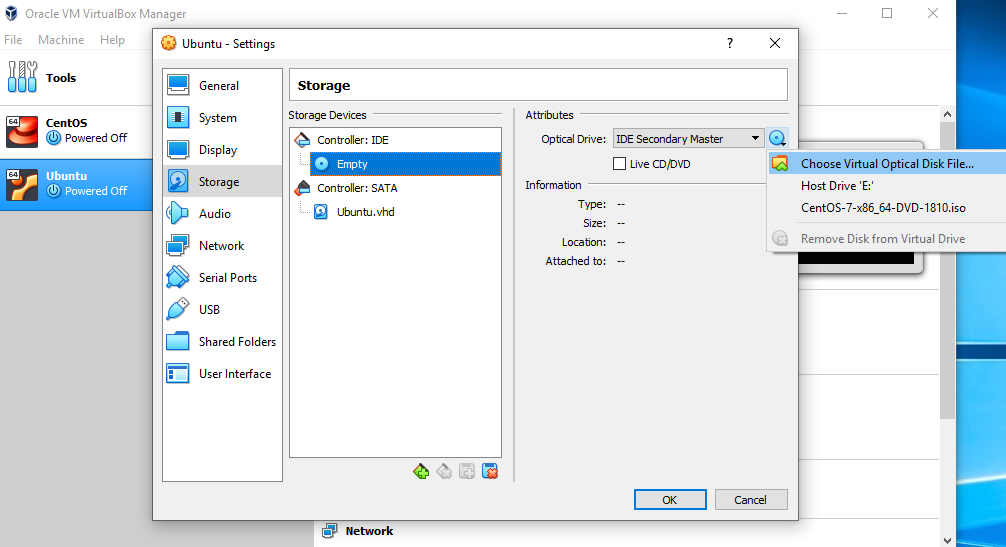
****

After creating a virtual hard disk, you'll see Ubuntu in your dashboard.

****

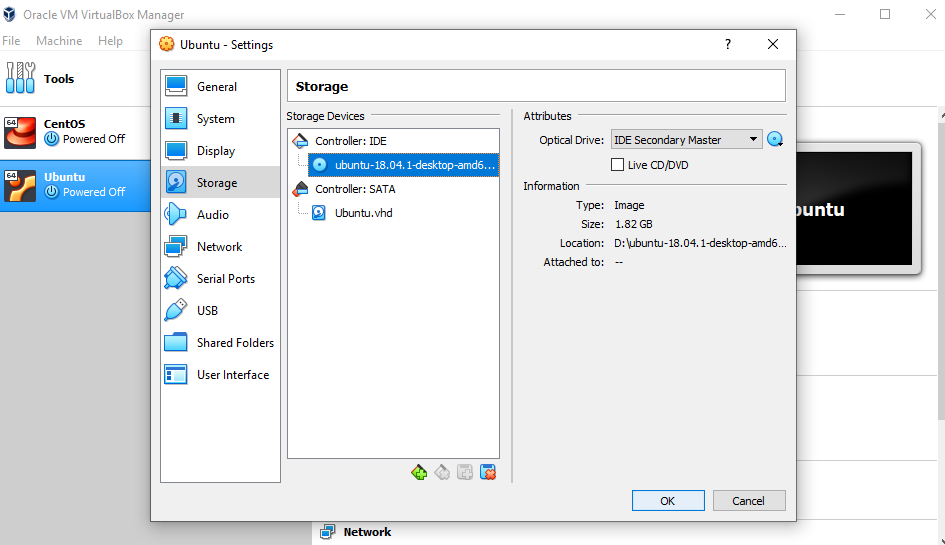
Now, we have to set up the Ubuntu disk image file (.iso).

The Ubuntu disk image file can be downloaded here: [Ubuntu OS download](https://ubuntu.com/#download).



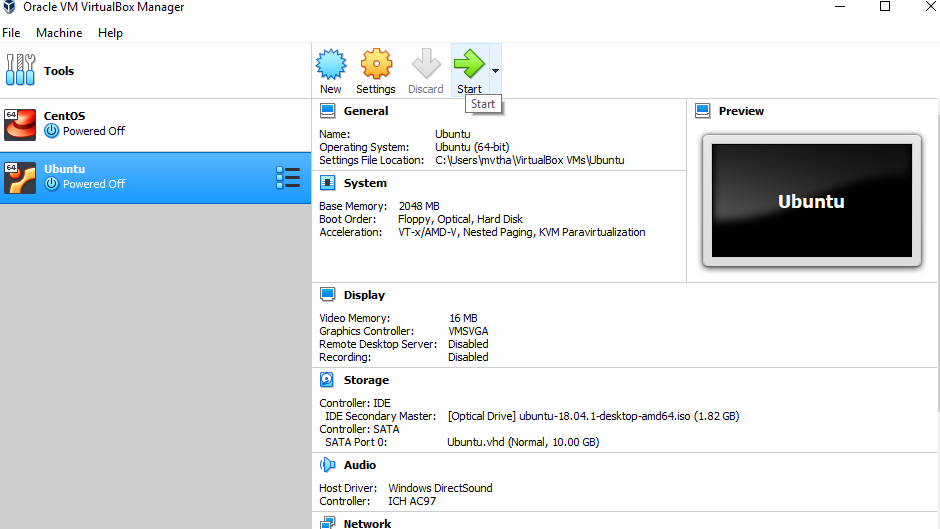
To set up the Ubuntu disk image file, go to settings and follow these steps:

1. Click "Storage"
2. In storage devices, click "Empty"
3. In attributes, click the disk image and "Choose Virtual Optical Disk File"
4. Select the Ubuntu disk image file and open it.



Click OK.

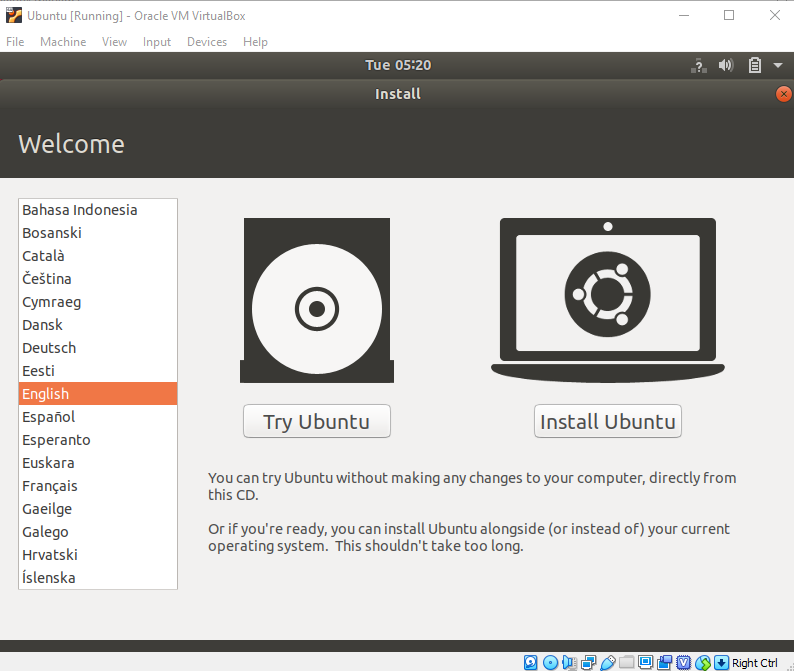
Your Ubuntu OS is ready to install in VirtualBox. Let's start!



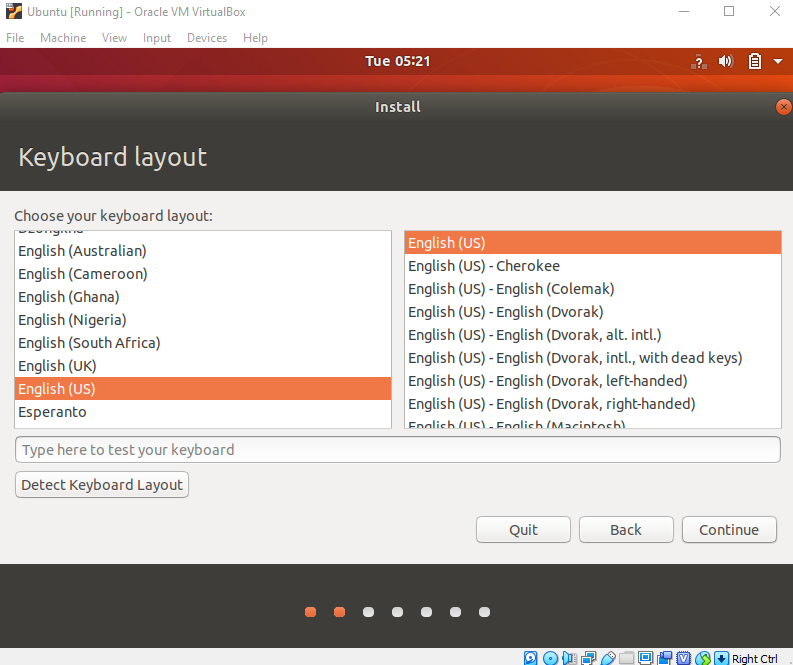
**NOTE:**Ubuntu VirtualBox installation and actual OS installation steps may vary. This guide helps you to install Ubuntu in VirtualBox only.

## Let's install Ubuntu!

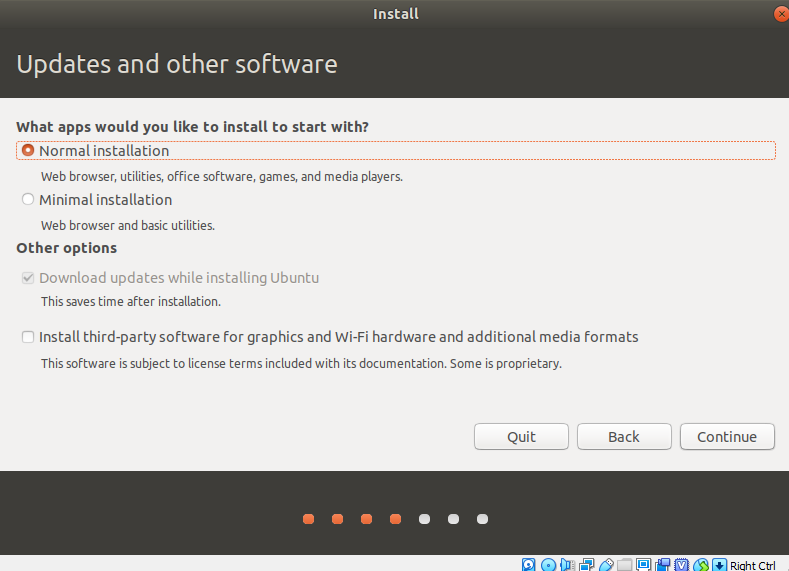
Click Install Ubuntu



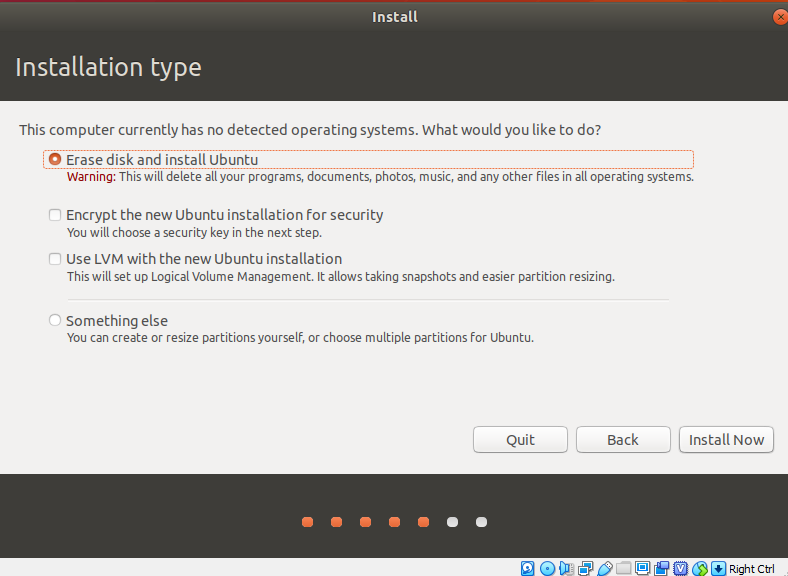
Select your keyboard layout.



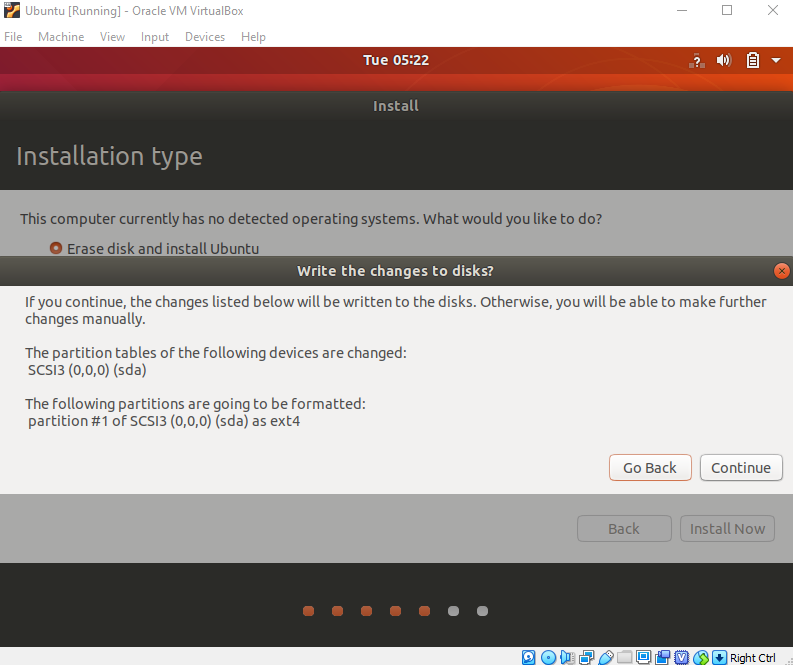
In the "Updates and other software" section, check "Normal installation" and continue.



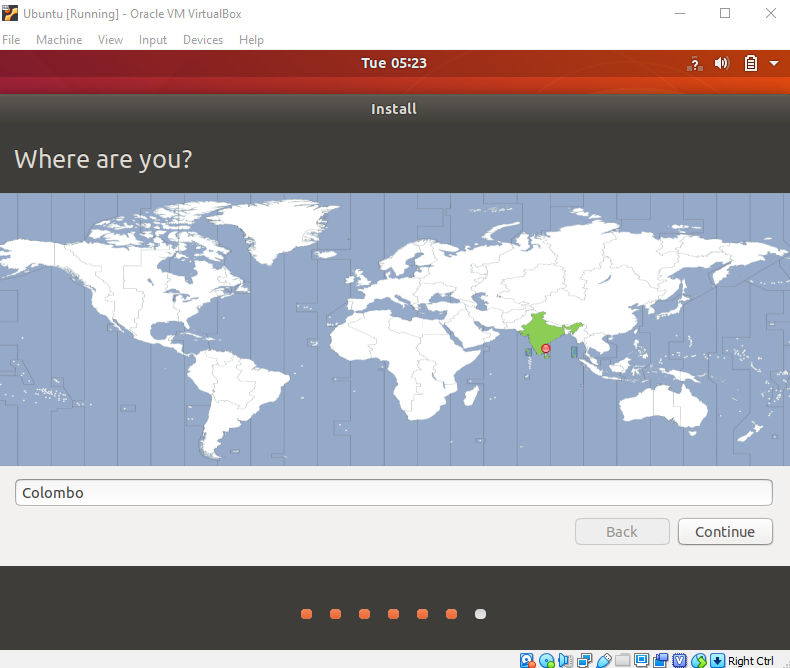
In "Installation type", check "Erase disk and install Ubuntu".



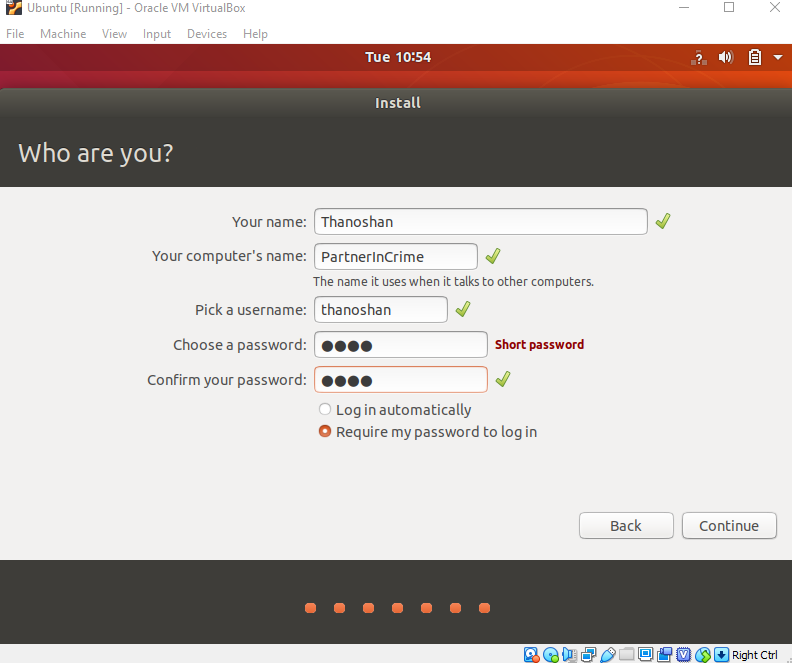
Click "Continue".



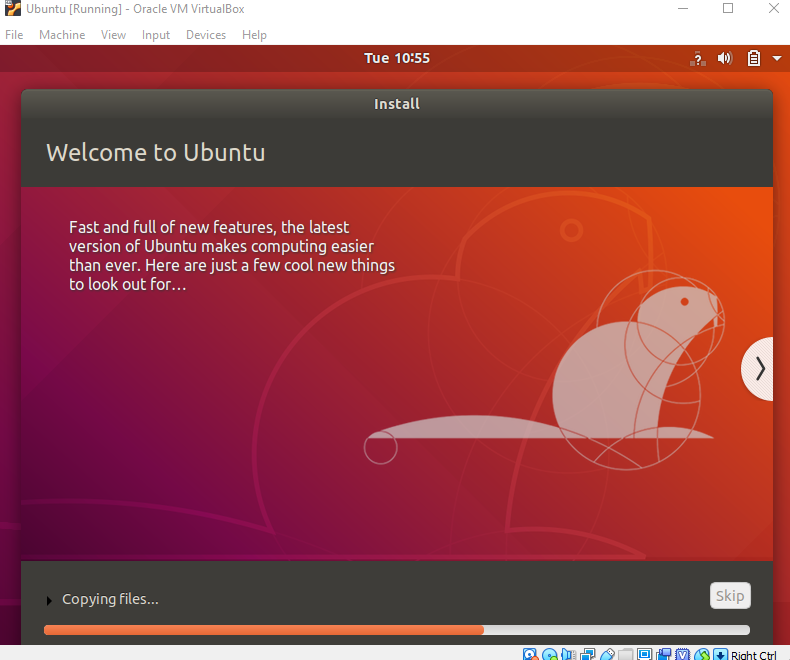
Choose your current location.



Now, set up your profile.

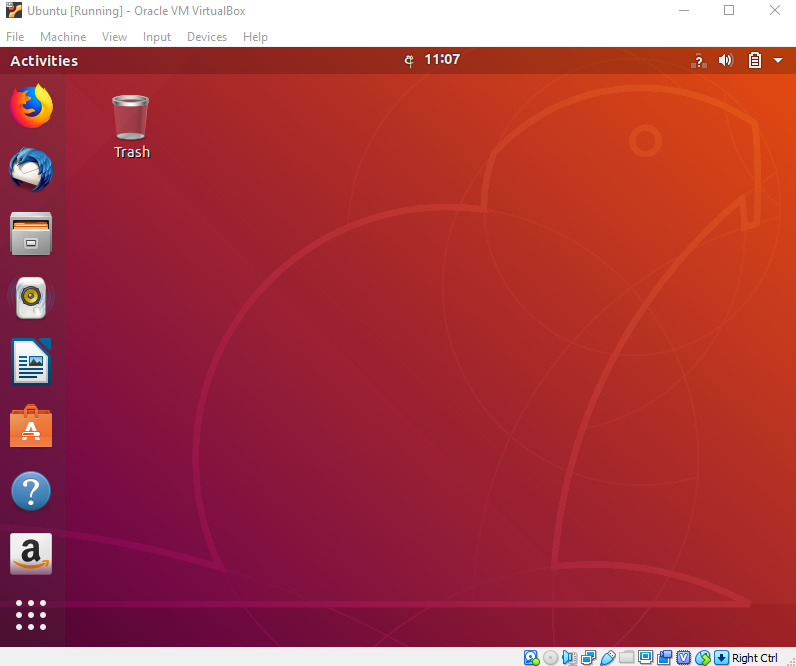


You'll see Ubuntu installing.



After the installation, restart it.

After logging in, you'll see the Ubuntu desktop.



**Conclusion :**VirtualBox is free and is a great tool for running multiple operating systems on a single OS. Ubuntu has many benefits. If you're a beginner to Linux, I would recommend you use Ubuntu as it's beginner friendly.