

Algorithm Design and Programming Fundamentals MCA SEM-1

Environment Setup and Basic C Programs - I

Part-I - Environment Setup - Windows with VS Code

- Download the GCC Compiler zip file. Alternatively you can download the exe version for the same.

You can get the GCC compiler from any of the following:

→ <https://winlibs.com/#download-release>

Other alternatives,

→ <https://www.mingw-w64.org/downloads/> [Go to WinLibs.com]

→ <https://sourceforge.net/projects/mingw-w64/> [Search for Online installer file]

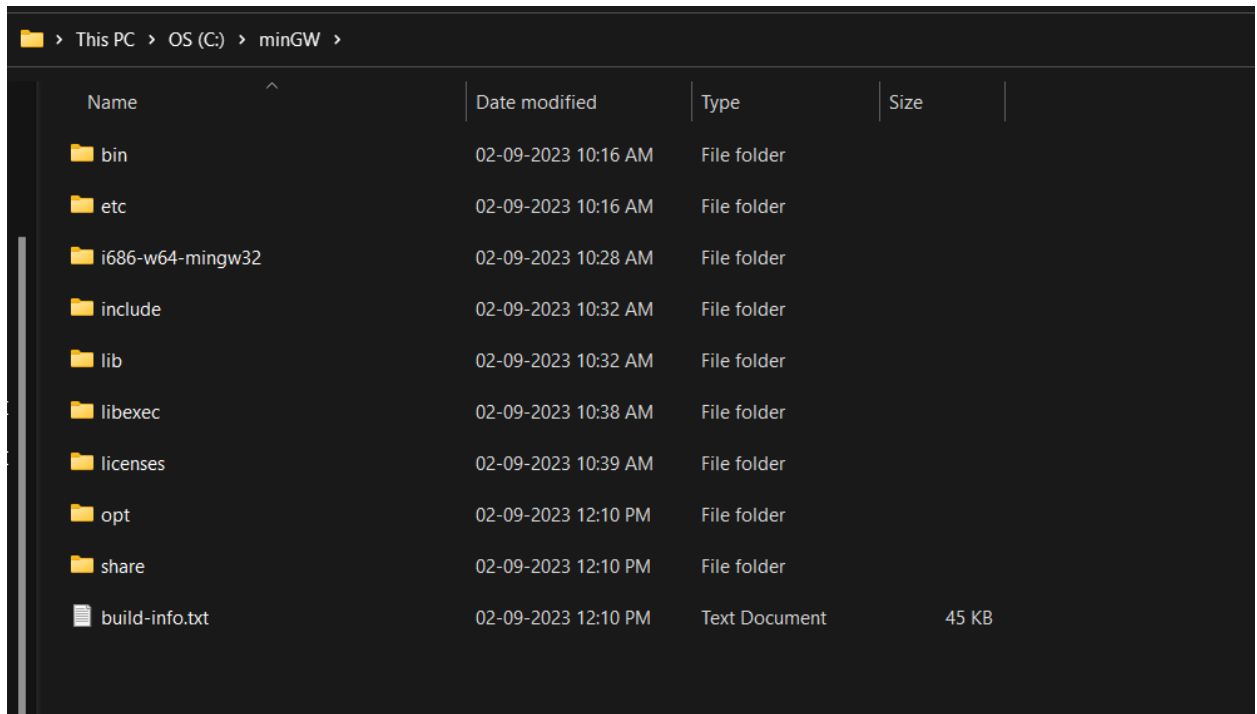
→ <https://sourceforge.net/projects/mingw-w64/files/mingw-w64/mingw-w64-release/>

→ <https://sourceforge.net/projects/mingw/>

→ etc...

- Extract the Zip folder to a path in your drive.

Here it is extracted at windows drive (C:)



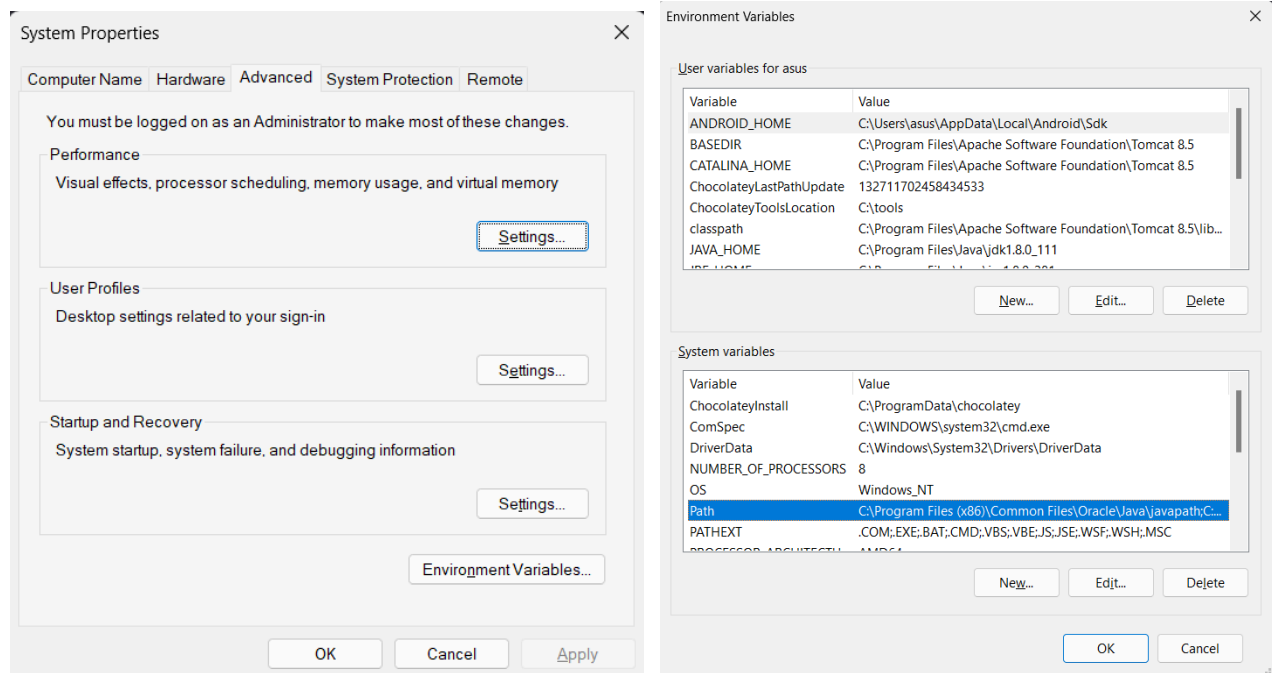
- Point the location of GCC into your system Environment Variable under “PATH”/ “Path”

Now the copy the location of bin folder - C:\minGW\bin

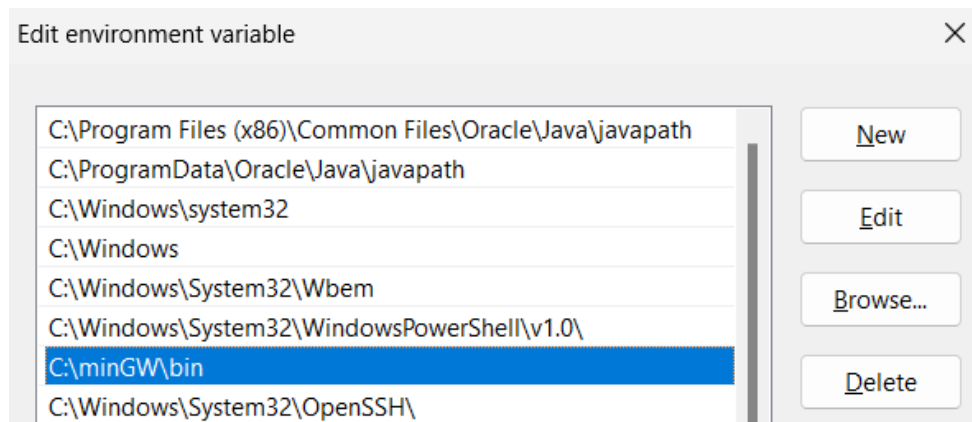
And set it to the environment variable of your machine.

You can access the Environment variable via going to properties by right clicking on “THIS PC”.

This PC => Properties => Advanced System Settings=> Environment Variables under Advance tab



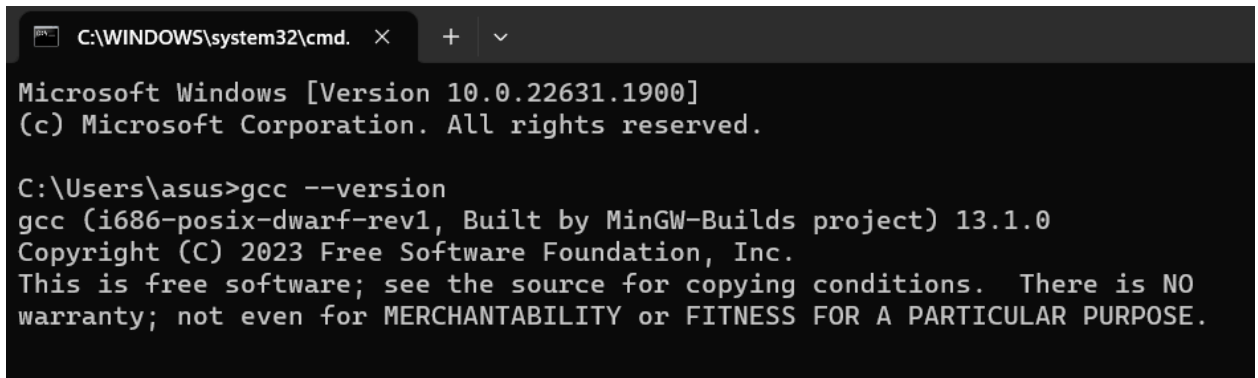
Set the path C:\minGW\bin in the path variable.



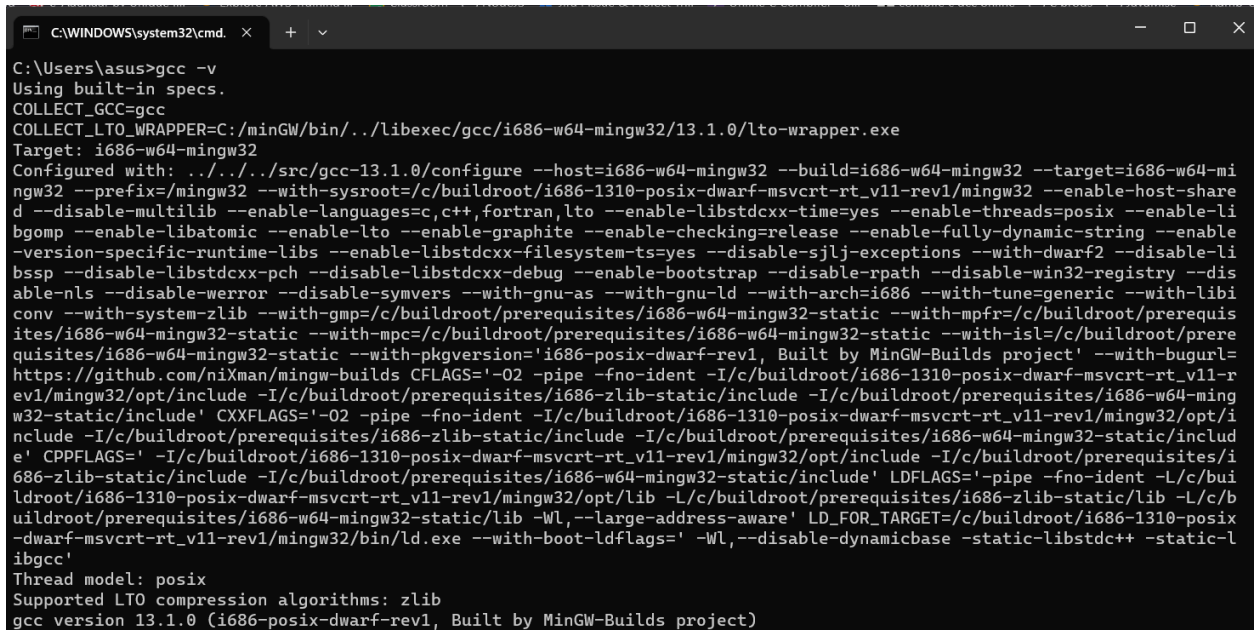
- Check the GCC version in your command prompt to validate the setup.

To check the version you can use the following command.

```
gcc --version  
gcc -v
```



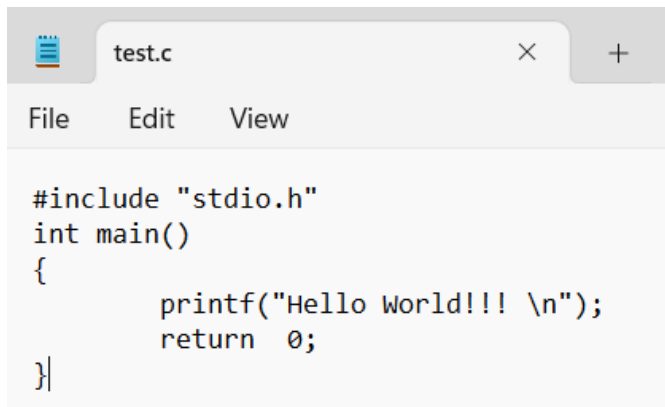
```
C:\WINDOWS\system32\cmd. X + v  
Microsoft Windows [Version 10.0.22631.1900]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\asus>gcc --version  
gcc (i686-posix-dwarf-rev1, Built by MinGW-Builds project) 13.1.0  
Copyright (C) 2023 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```



```
C:\WINDOWS\system32\cmd. X + v  
C:\Users\asus>gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=C:/minGW/bin/./libexec/gcc/i686-w64-mingw32/13.1.0/lto-wrapper.exe  
Target: i686-w64-mingw32  
Configured with: ../../src/gcc-13.1.0/configure --host=i686-w64-mingw32 --build=i686-w64-mingw32 --target=i686-w64-mingw32 --prefix=/mingw32 --with-sysroot=/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32 --enable-host-shared --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=posix --enable-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --enable-version-specific-runtime-libs --enable-libstdcxx-filesystem-ts=yes --disable-sjlj-exceptions --with-dwarf2 --disable-libssp --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=i686 --with-tune=generic --with-libiconv --with-system-zlib --with-gmp=/c/buildroot/prerequisites/i686-w64-mingw32-static --with-mpfr=/c/buildroot/prerequisites/i686-w64-mingw32-static --with-mpc=/c/buildroot/prerequisites/i686-w64-mingw32-static --with-isl=/c/buildroot/prerequisites/i686-w64-mingw32-static --with-pkgversion='i686-posix-dwarf-rev1, Built by MinGW-Builds project' --with-bugurl=https://github.com/nixman/mingw-builds CFLAGS='-O2 -pipe -fno-ident -I/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32/opt/include -I/c/buildroot/prerequisites/i686-zlib-static/include -I/c/buildroot/prerequisites/i686-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32/opt/include -I/c/buildroot/prerequisites/i686-zlib-static/include -I/c/buildroot/prerequisites/i686-w64-mingw32-static/include' CPPFLAGS='-I/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32/opt/include -I/c/buildroot/prerequisites/i686-zlib-static/include -I/c/buildroot/prerequisites/i686-w64-mingw32-static/include' LDFLAGS='-pipe -fno-ident -L/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32/opt/lib -L/c/buildroot/prerequisites/i686-zlib-static/lib -L/c/buildroot/prerequisites/i686-w64-mingw32-static/lib -Wl,--large-address-aware' LD_FOR_TARGET=/c/buildroot/i686-1310-posix-dwarf-msvcrt-rt_v11-rev1/mingw32/bin/ld.exe --with-boot-ldflags='-Wl,--disable-dynamicbase -static-libstdc++ -static-libgcc'  
Thread model: posix  
Supported LTO compression algorithms: zlib  
gcc version 13.1.0 (i686-posix-dwarf-rev1, Built by MinGW-Builds project)
```

This ensures that your gcc is set up properly.

- To compile a program, create file test.c with a simple “Hello World” message and save it in your drive.

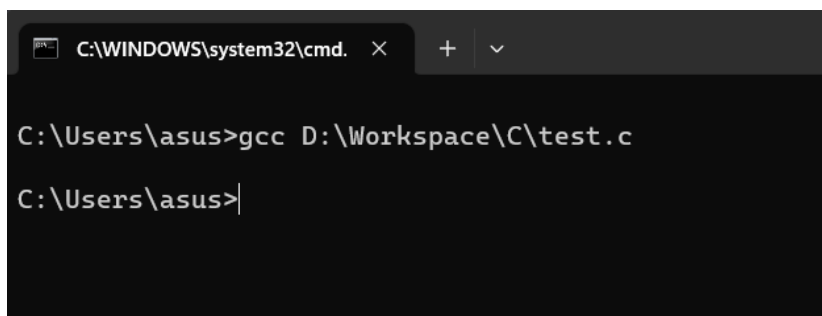
A screenshot of a code editor window titled 'test.c'. The window has a menu bar with 'File', 'Edit', and 'View'. The code inside is a simple C program that prints 'Hello World!!! \n' and returns 0.

```
#include "stdio.h"
int main()
{
    printf("Hello World!!! \n");
    return 0;
}
```

Open your command prompt and execute the following command

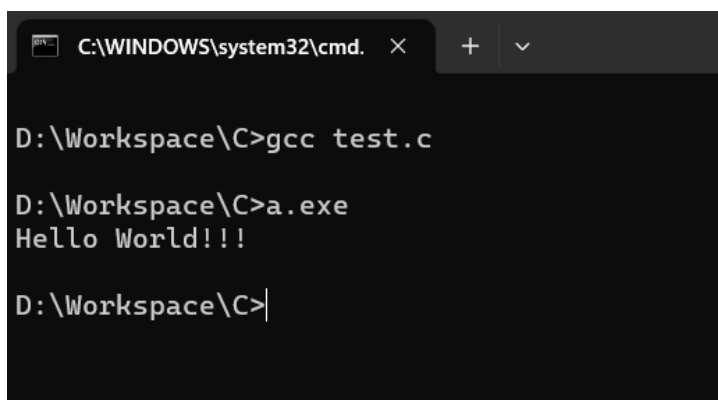
```
gcc <filename/ full file path> or
gcc <filename> -o outputname

gcc D:\Workspace\C\test.c or
gcc test.c -o test
```

A screenshot of a Windows command prompt window. The title bar shows 'C:\WINDOWS\system32\cmd.'. The prompt is at 'C:\Users\asus>'. The user has entered 'gcc D:\Workspace\C\test.c' and the prompt is now at 'C:\Users\asus>|' without any error messages.

```
C:\WINDOWS\system32\cmd.
C:\Users\asus>gcc D:\Workspace\C\test.c
C:\Users\asus>|
```

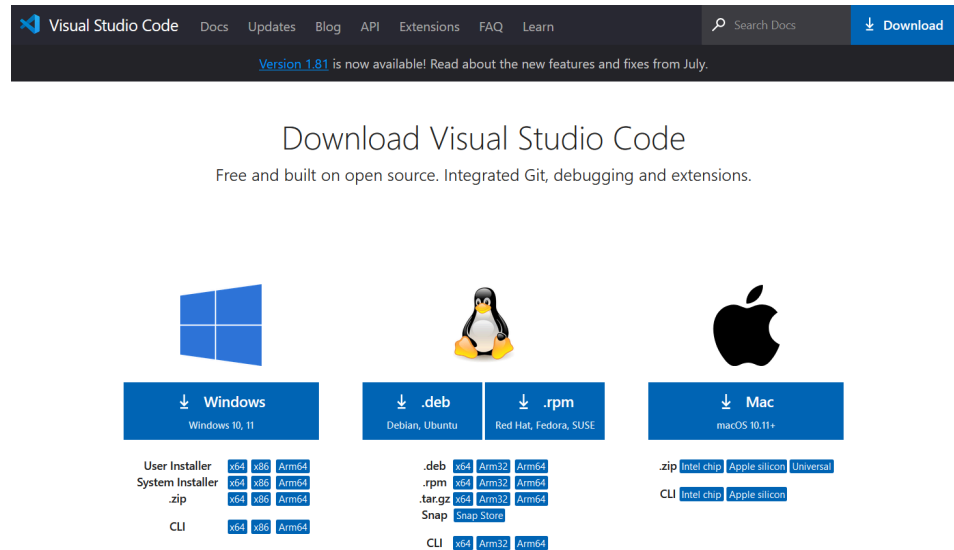
If there is no error message then your compiler has successfully compiled the file and you will have “a.exe” generated in your folder. You can check output by executing “a.exe”.

A screenshot of a Windows command prompt window. The title bar shows 'C:\WINDOWS\system32\cmd.'. The prompt is at 'D:\Workspace\C>'. The user has entered 'gcc test.c' and 'a.exe', and the output 'Hello World!!!' is displayed.

```
C:\WINDOWS\system32\cmd.
D:\Workspace\C>gcc test.c
D:\Workspace\C>a.exe
Hello World!!!
D:\Workspace\C>|
```

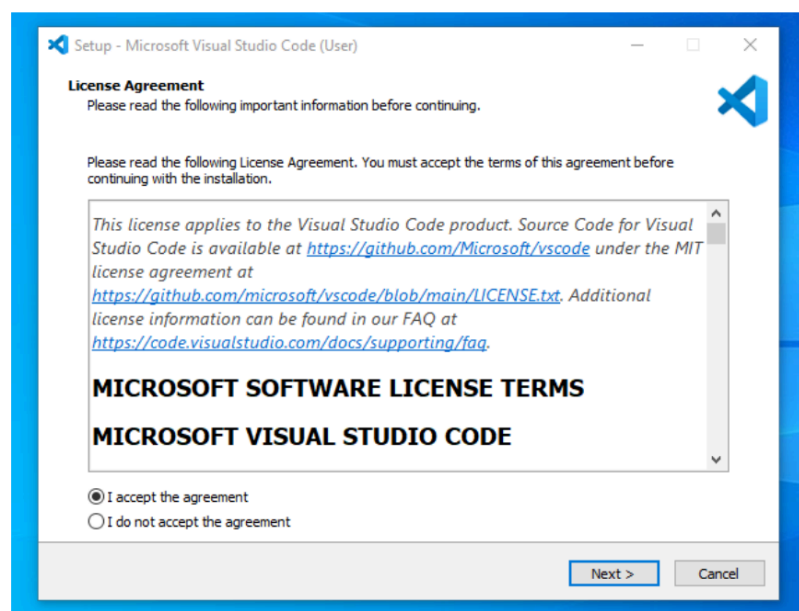
Setup your VS CODE editor

- Visit the <https://code.visualstudio.com/Download> to download the Visual Studio Code using any web browser. Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.

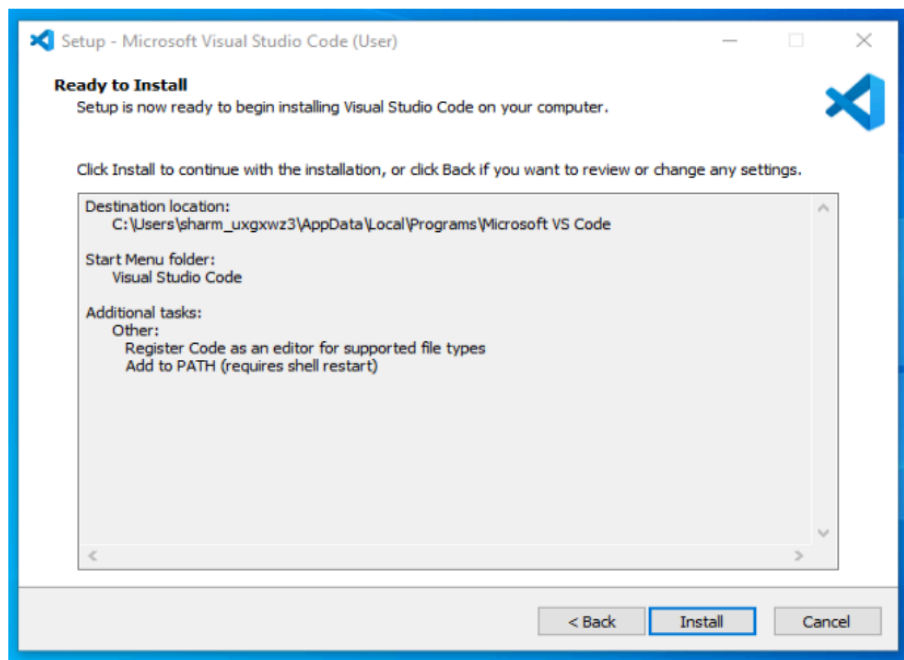
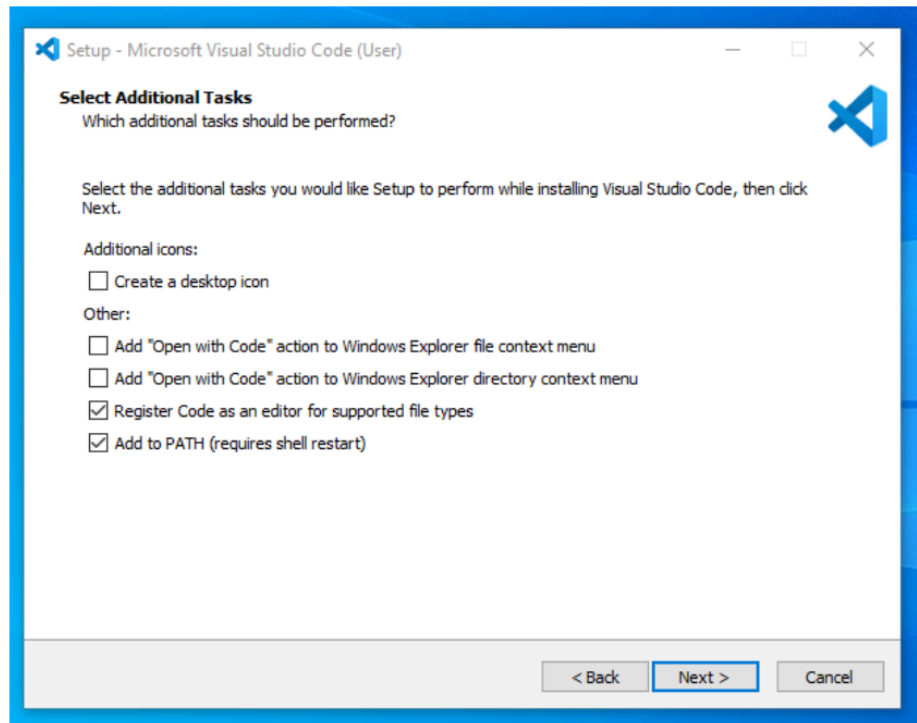


- When the download finishes, then the Visual Studio Code file appears in the downloads folder. Click on the installer to start the installation process of the Visual Studio Code.

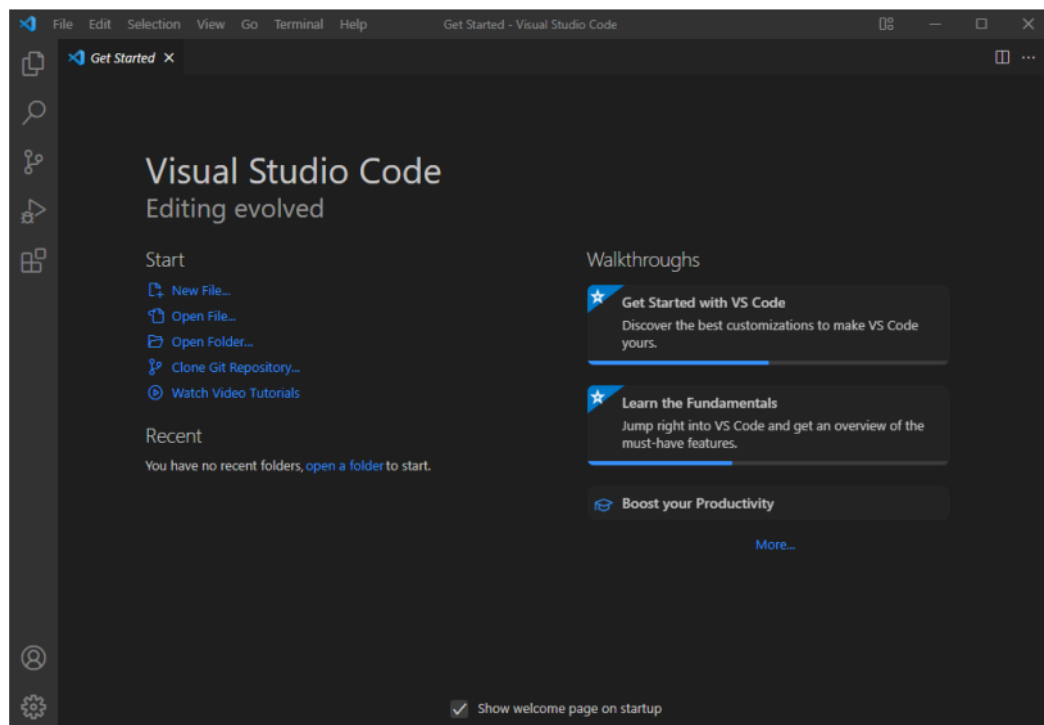
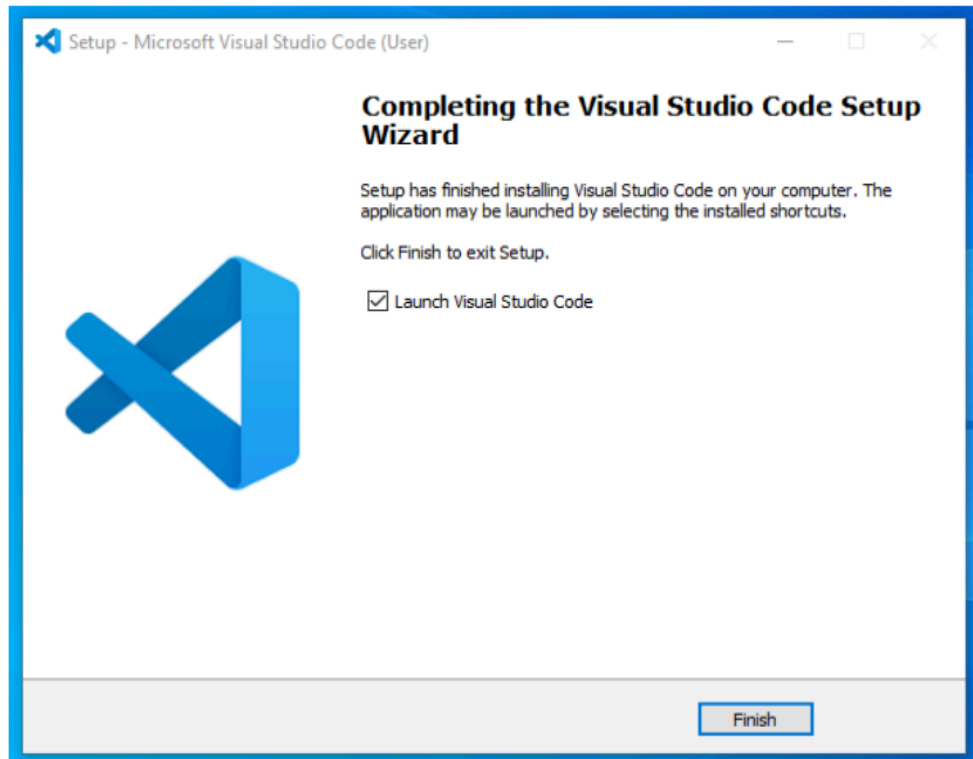
After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.



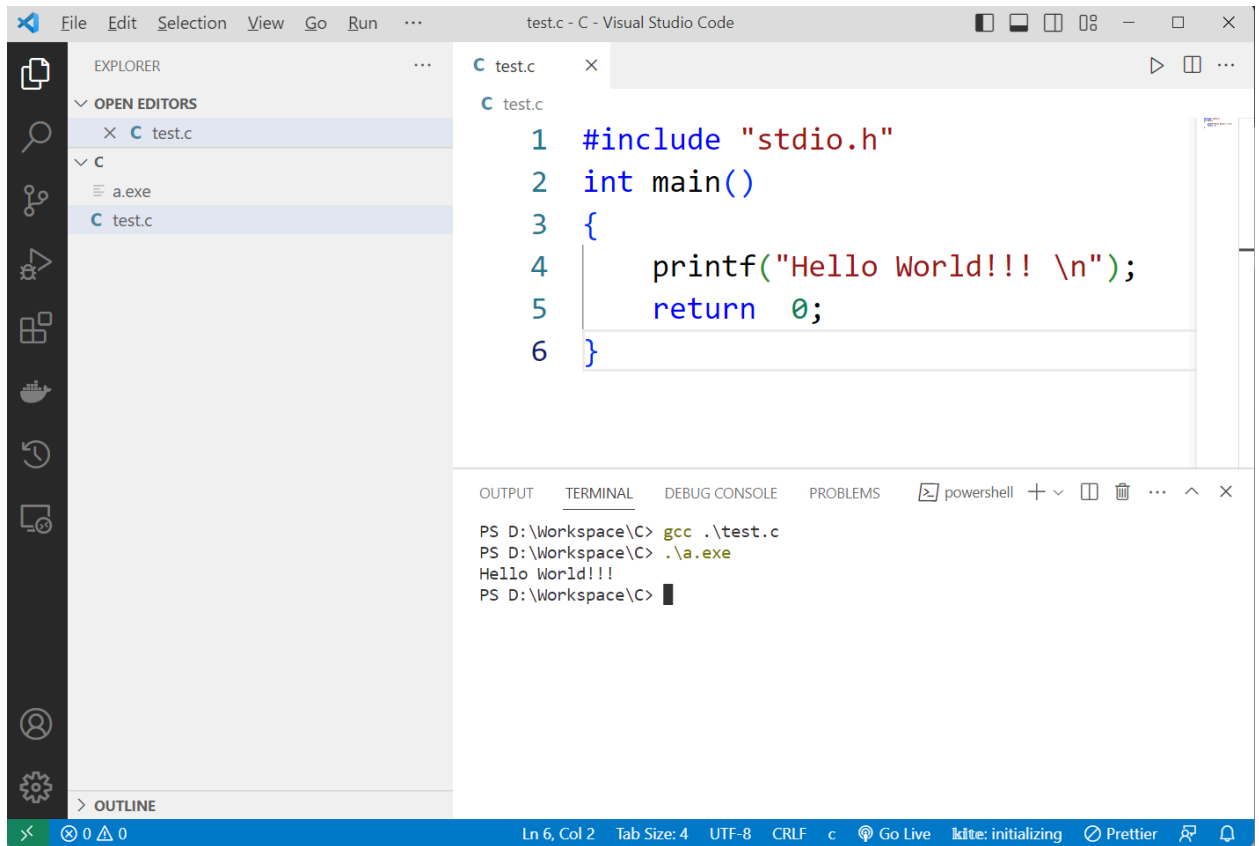
- Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the Next button. Then it will ask for the installing setup. Click on the Install button.



- After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.



- Here You can create your file and execute it under the terminal window.



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows the file structure with 'test.c' selected. The main editor window displays the following C code:

```
1 #include "stdio.h"
2 int main()
3 {
4     printf("Hello World!!! \n");
5     return 0;
6 }
```

Below the editor, the TERMINAL panel is active, showing the execution of the program:

```
PS D:\Workspace\C> gcc .\test.c
PS D:\Workspace\C> .\a.exe
Hello World!!!
PS D:\Workspace\C> █
```

The status bar at the bottom indicates the current line and column (Ln 6, Col 2), tab size (4), encoding (UTF-8), and other settings.

DO NOT INSTALL ANY EXTENSION FOR C/ C++ in VSCode.

Part - II

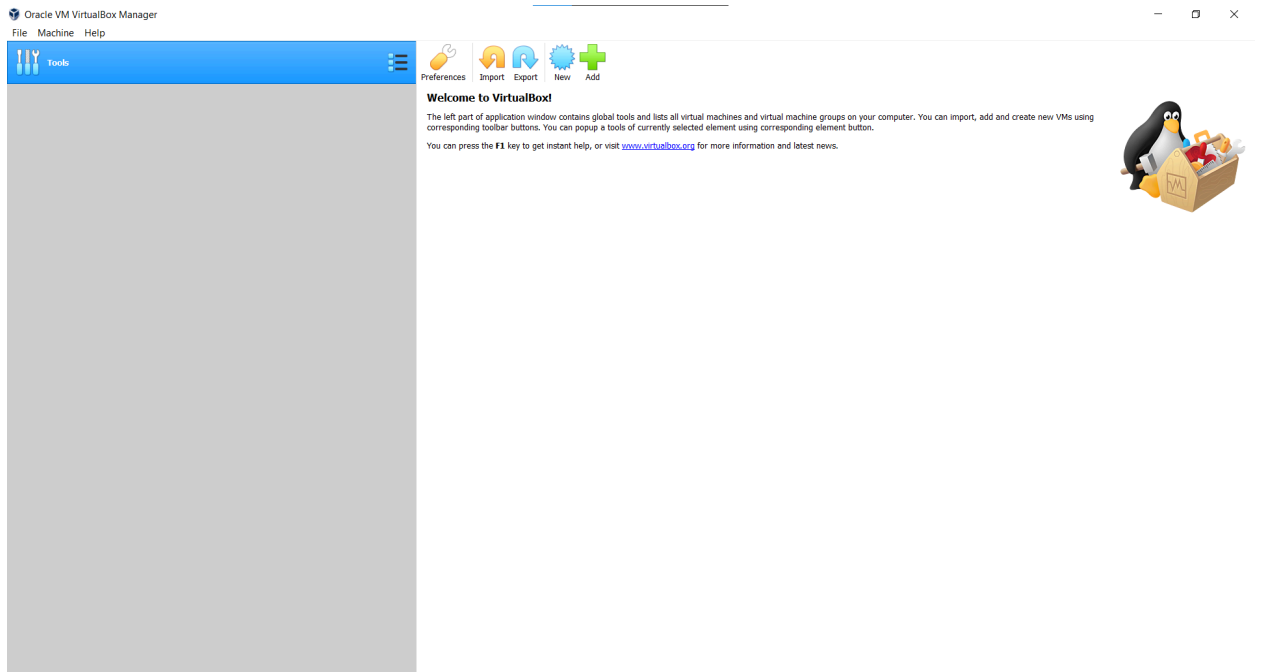
Basic C Programs

- A. Write down the steps to perform operations of following programs. Draw appropriate flowchart for the same.
- B. Perform the following programs using GCC. Write down the commands to compile and run the program.
 - Write a C program to print “Hello DDU”.
 - Write a simple C program to add two numbers.
 - Write a simple C program to subtract two numbers.
 - Write a simple C program to multiply two numbers.
 - Write a simple C program to divide two numbers.
 - Write a simple C program to divide two numbers and find the remainder.
 - Write a C program to print all your personal and academic details using new line separation.

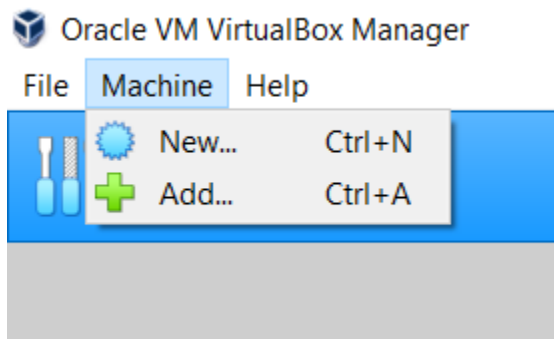
Appendix

Linux Machine Setup

1. Download Oracle Virtual Box [For setting up virtual machine in your computer]
<https://www.virtualbox.org/wiki/Downloads>
2. Download Ubuntu Operating System.
<https://ubuntu.com/download/desktop>
3. Start Oracle Virtual Box.



4. Click on New to create a new virtual machine.



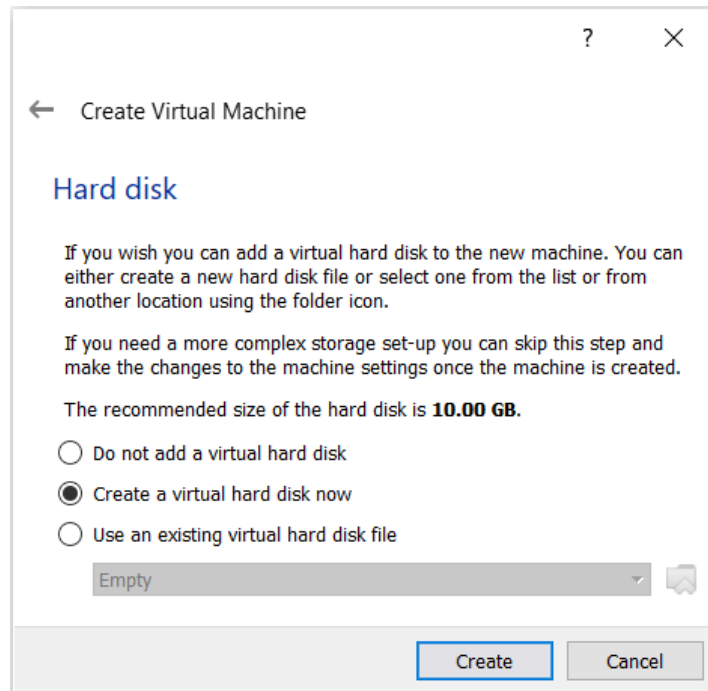
5. Give proper name to the virtual machine to be created. Select the Type as Linux and appropriate version.

The screenshot shows the 'Create Virtual Machine' dialog box in Oracle VM VirtualBox. The title bar has a question mark and a close button. Below the title bar is a back arrow and the text 'Create Virtual Machine'. The main heading is 'Name and operating system'. Below this is a paragraph: 'Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.' There are four input fields: 'Name:' with the text 'Ubuntu_GCC_GPP', 'Machine Folder:' with a folder icon and the path 'C:\Users\asus\VirtualBox VMs', 'Type:' with a dropdown menu showing 'Linux', and 'Version:' with a dropdown menu showing 'Ubuntu (64-bit)'. To the right of the 'Type' and 'Version' dropdowns is a small icon of a person. At the bottom are three buttons: 'Expert Mode', 'Next' (highlighted with a blue border), and 'Cancel'.

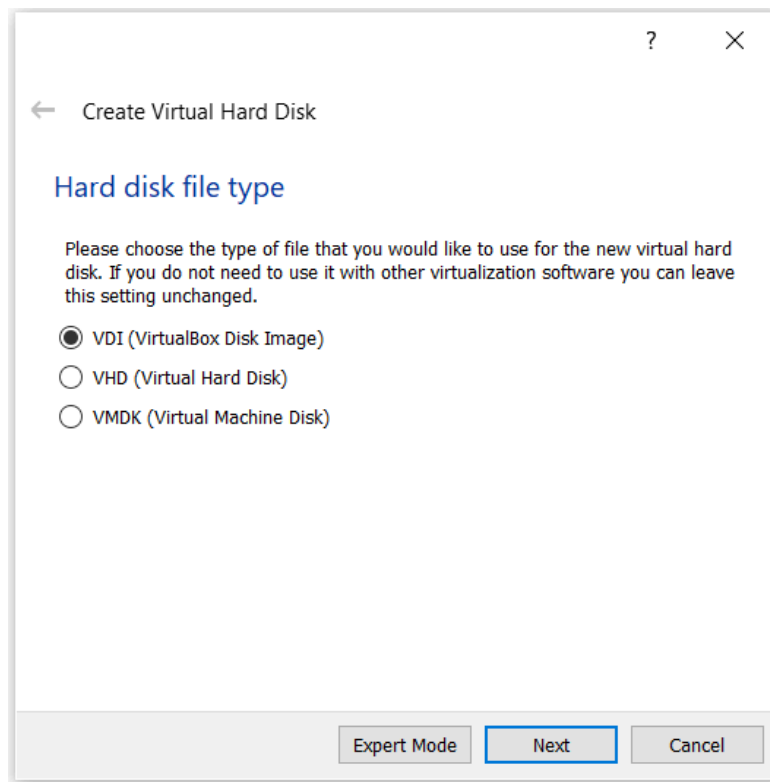
6. Select memory for the virtual machine then click next.

The screenshot shows the 'Create Virtual Machine' dialog box in Oracle VM VirtualBox, specifically the 'Memory size' step. The title bar has a question mark and a close button. Below the title bar is a back arrow and the text 'Create Virtual Machine'. The main heading is 'Memory size'. Below this is a paragraph: 'Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.' followed by 'The recommended memory size is **1024** MB.' There is a horizontal slider bar with a blue handle. Below the slider bar is a color-coded bar with green on the left and red on the right. The slider bar has '4 MB' at the left end and '8192 MB' at the right end. To the right of the slider bar is a text box containing '2048' and a unit 'MB'. At the bottom are two buttons: 'Next' (highlighted with a blue border) and 'Cancel'.

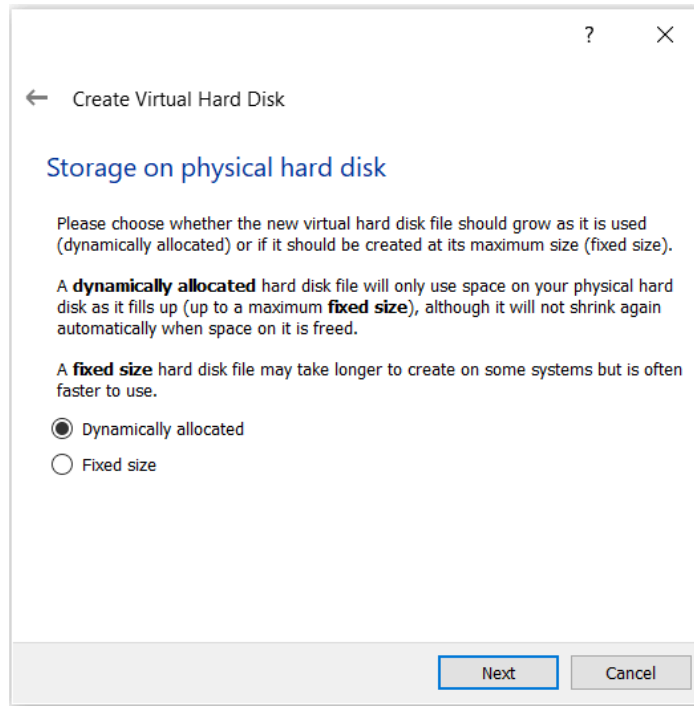
7. Choose the HDD type and then click Create. It is recommended to select a virtual hard drive.



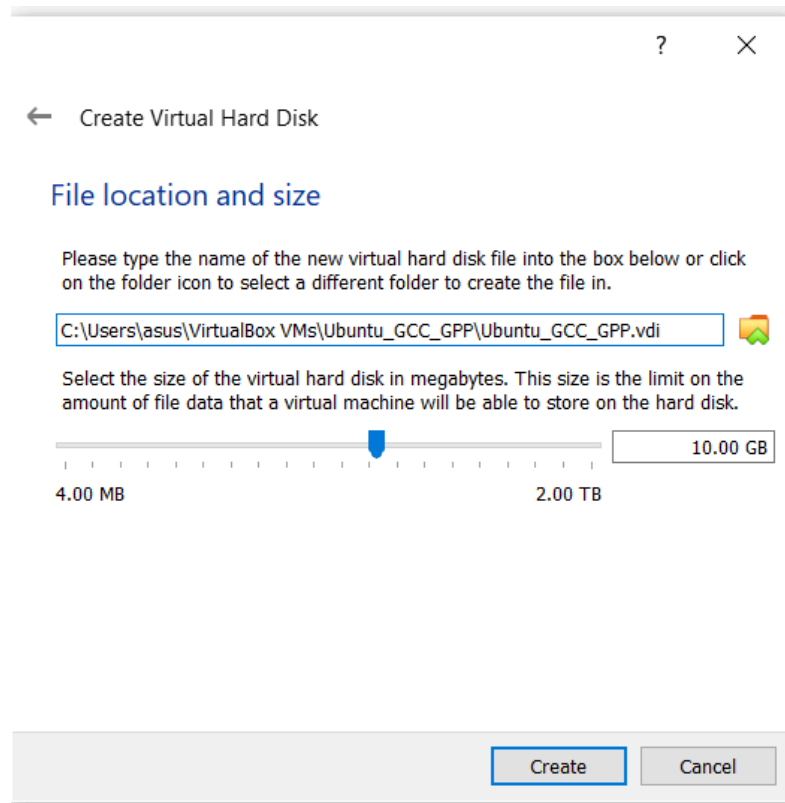
8. Choose the format for the virtual hard drive. (VDI is recommended)



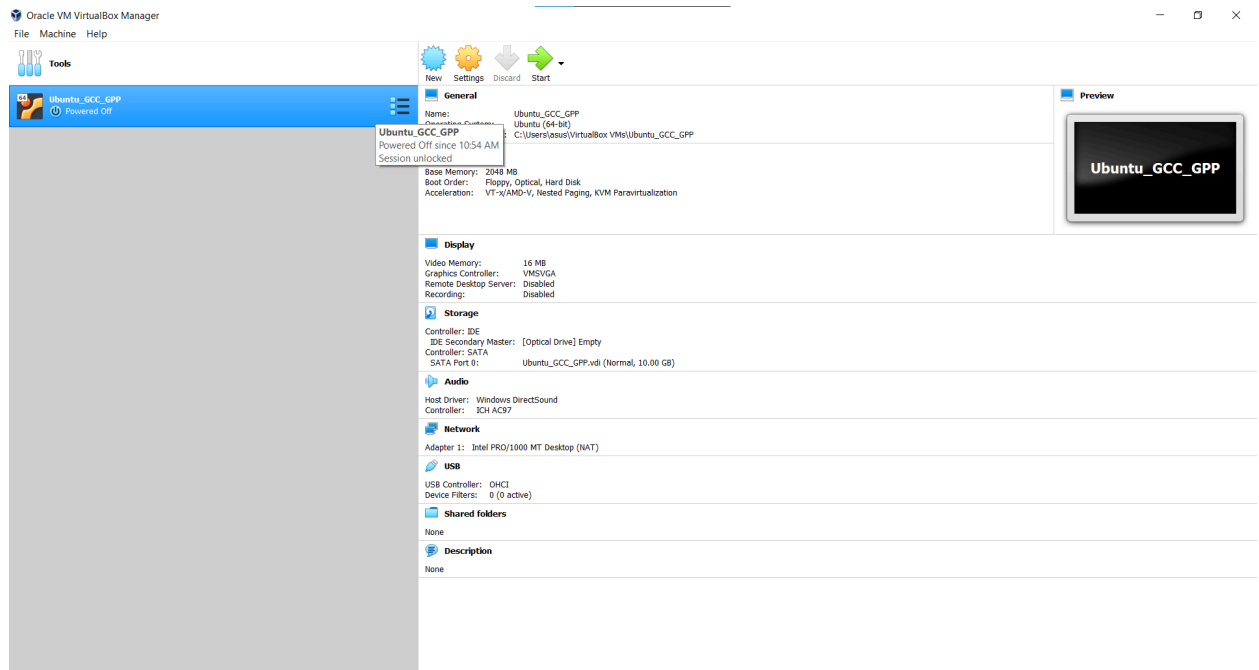
9. Choose allocation type for virtual hard drive. It is recommended to keep dynamic size.



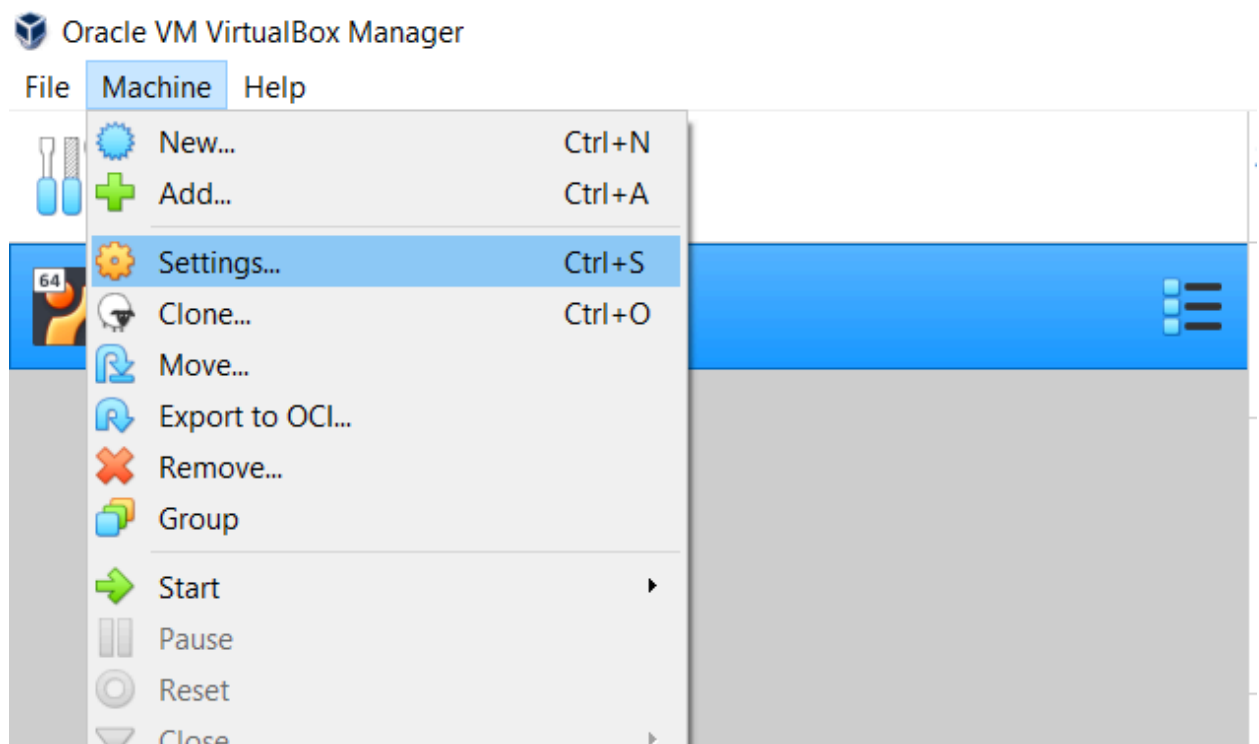
10. You can name the virtual hard drive and its size here then click create.



11. You will see a new virtual machine create now. Next we have to install operating system on the virtual machine.



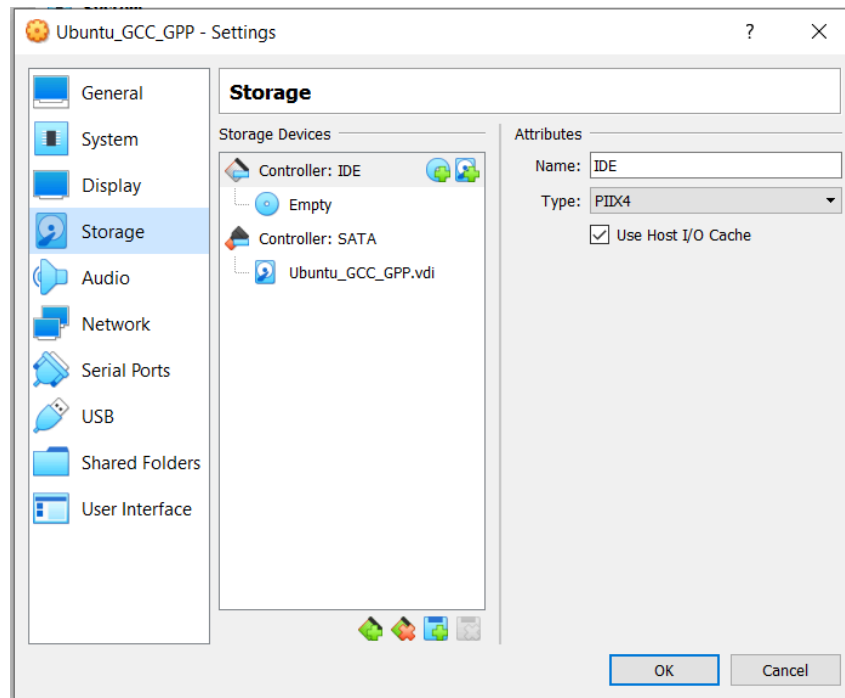
12. Click on Setting from the Machines menu. This will open a settings window for the selected virtual machine.



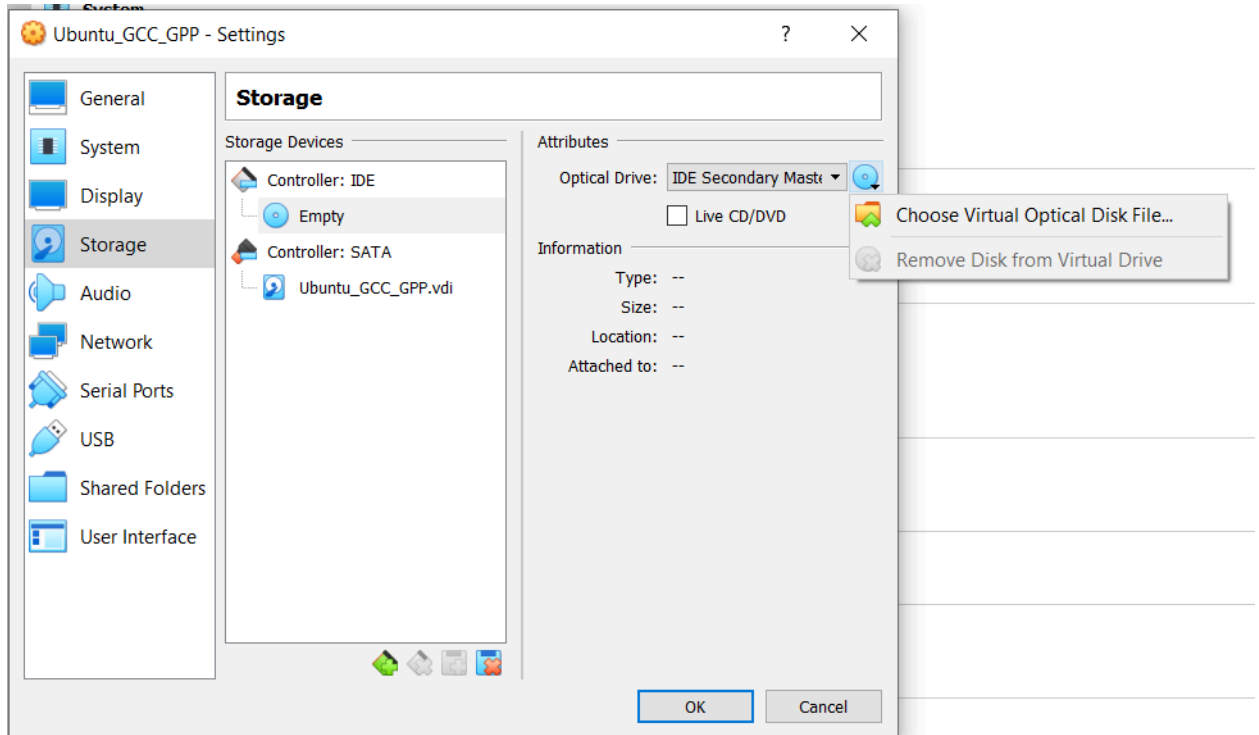
13. In the Settings window click on Storage option. Here you can see the virtual hard disk and an IDE drive option.

Now We have to load the operating system image into the optical drive.

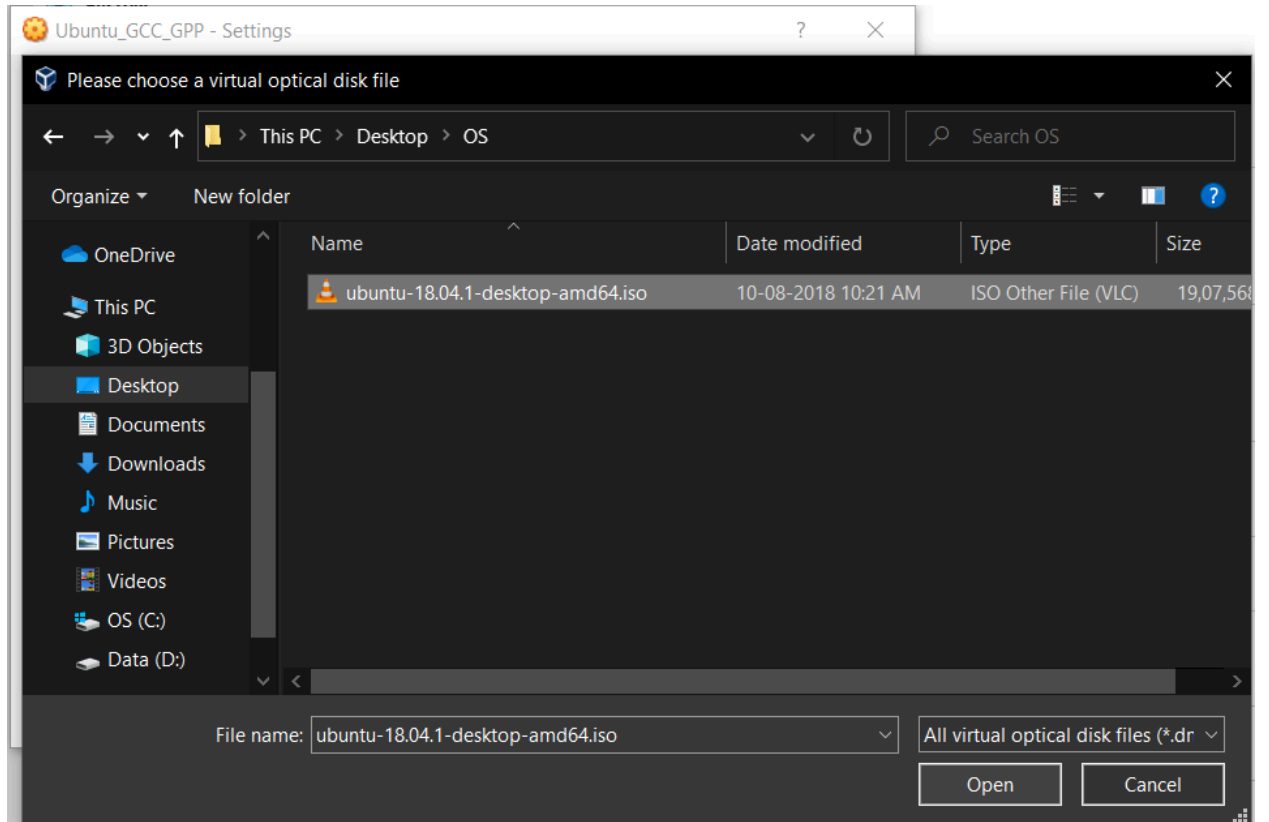
Select the Empty IDE controller.



14. Click on the disk icon. This will let you choose the ISO image of Ubuntu OS.

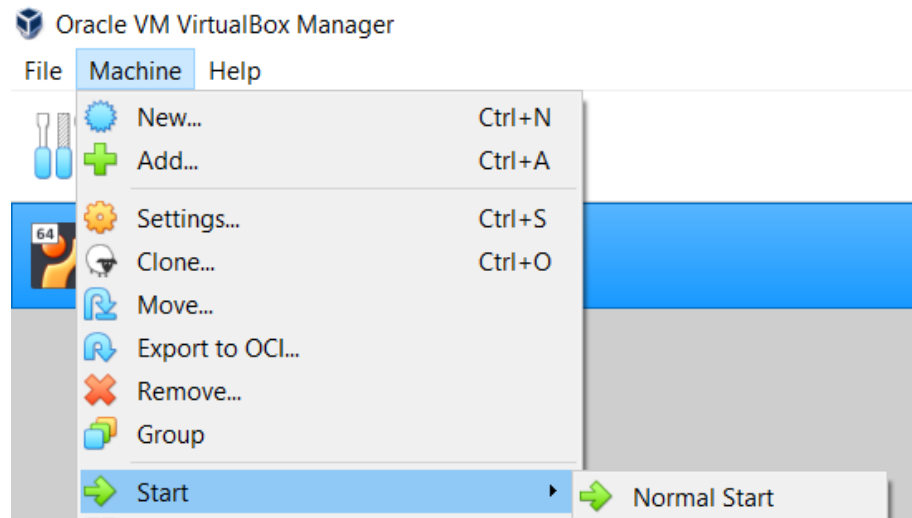


15. Choose the ISO of operating system. Click on Open and then OK.



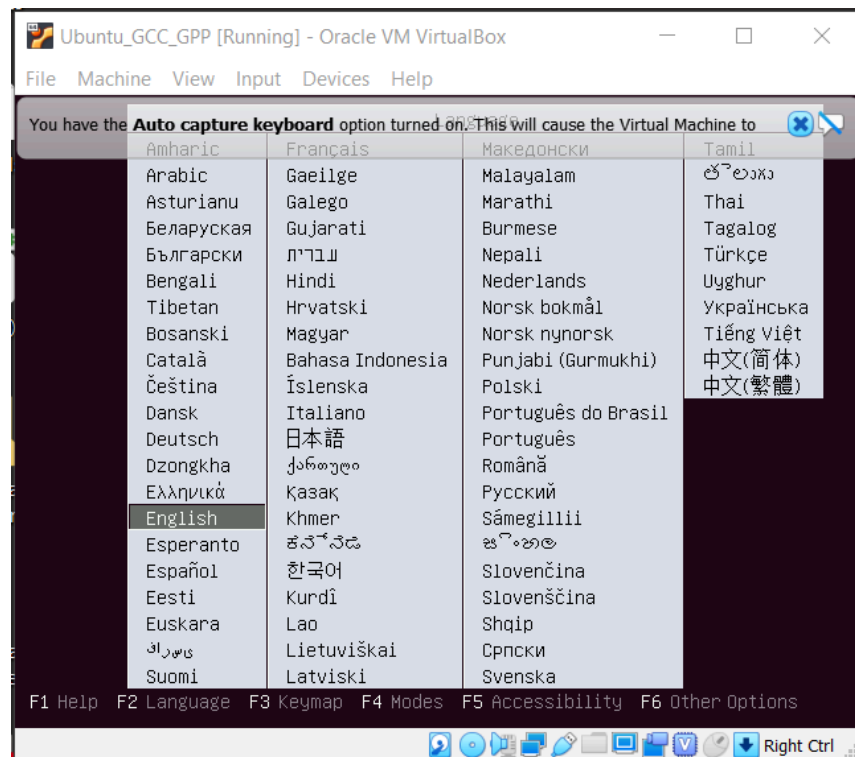
16. Now we can start the virtual machine and continue with the installation process of the operating system.

Click on Start [Normal Start] from the Machine Menu.

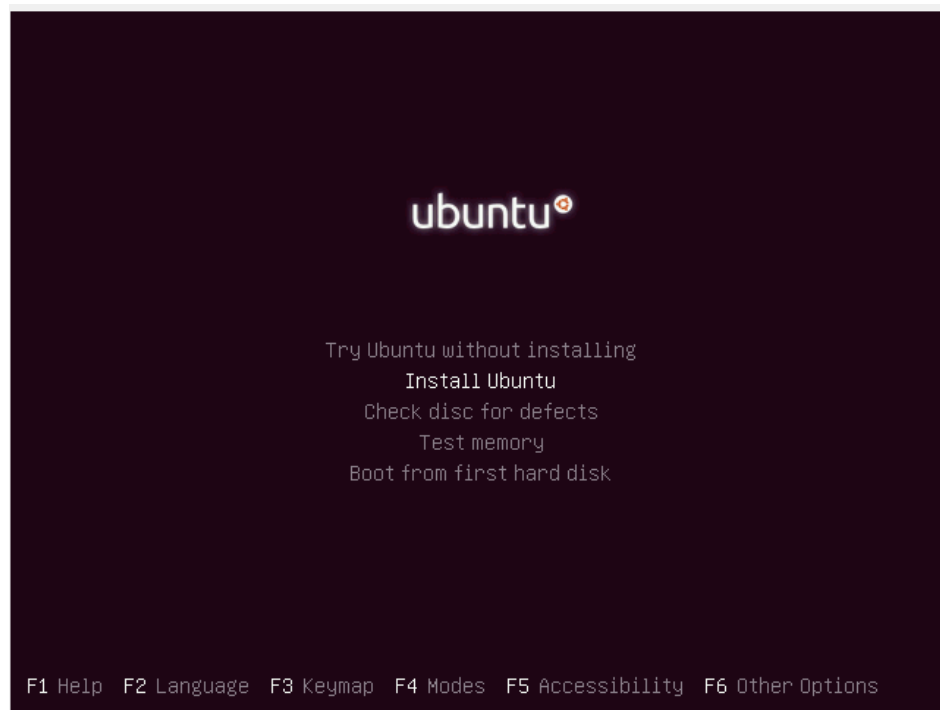


17. You will see the following screen once the machine has started. This is the process for installing Ubuntu OS.

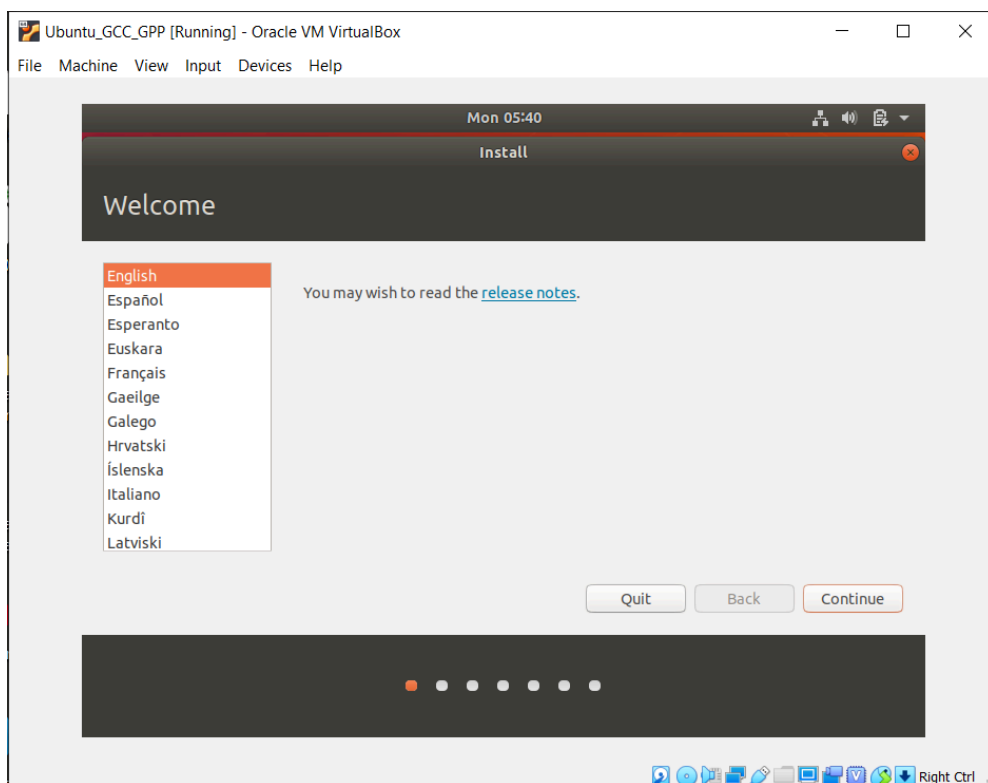
Select Language - English

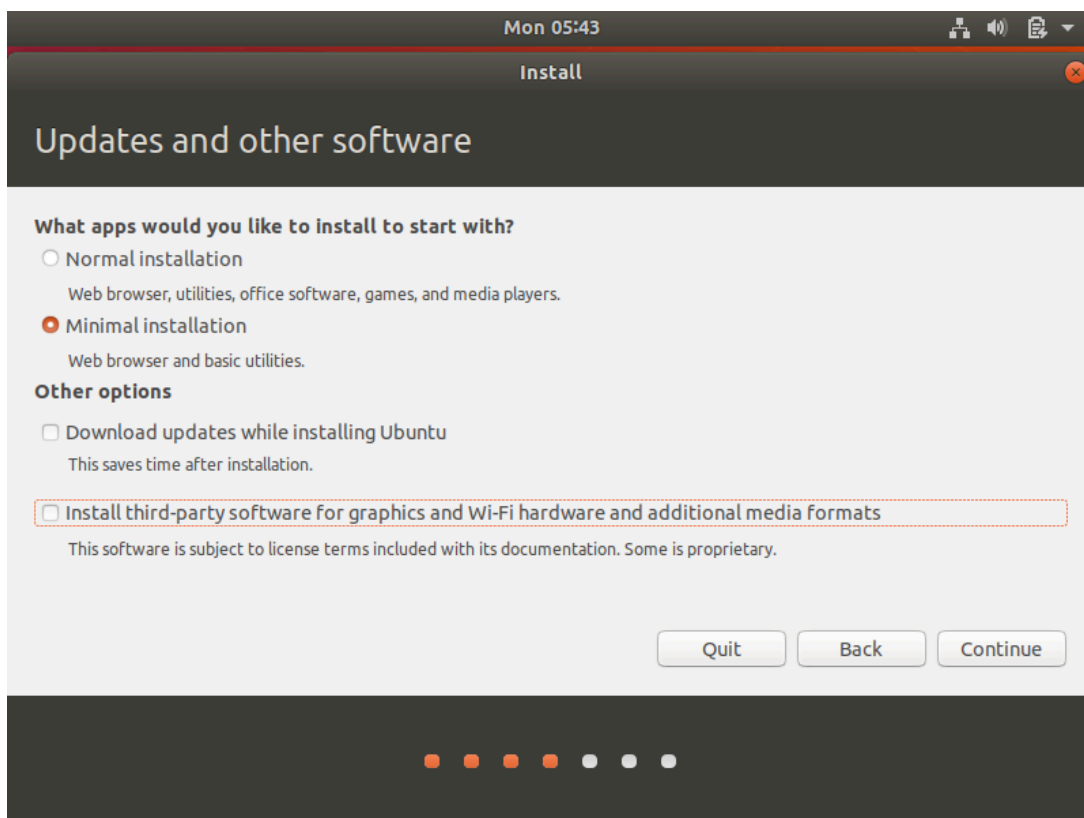
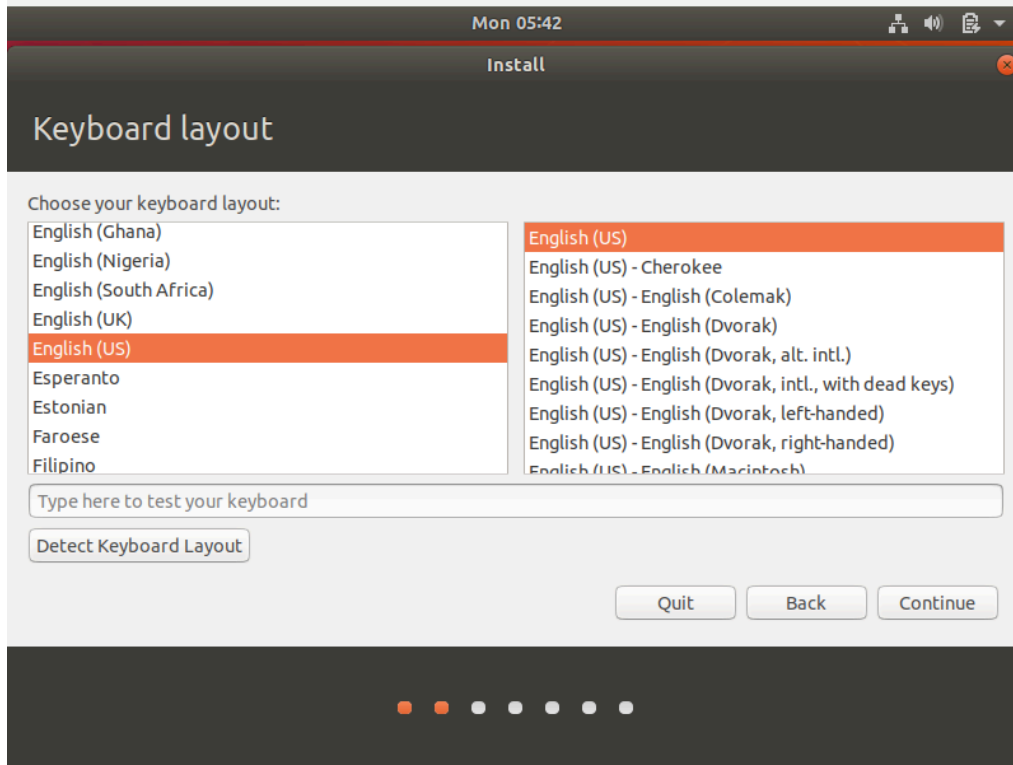


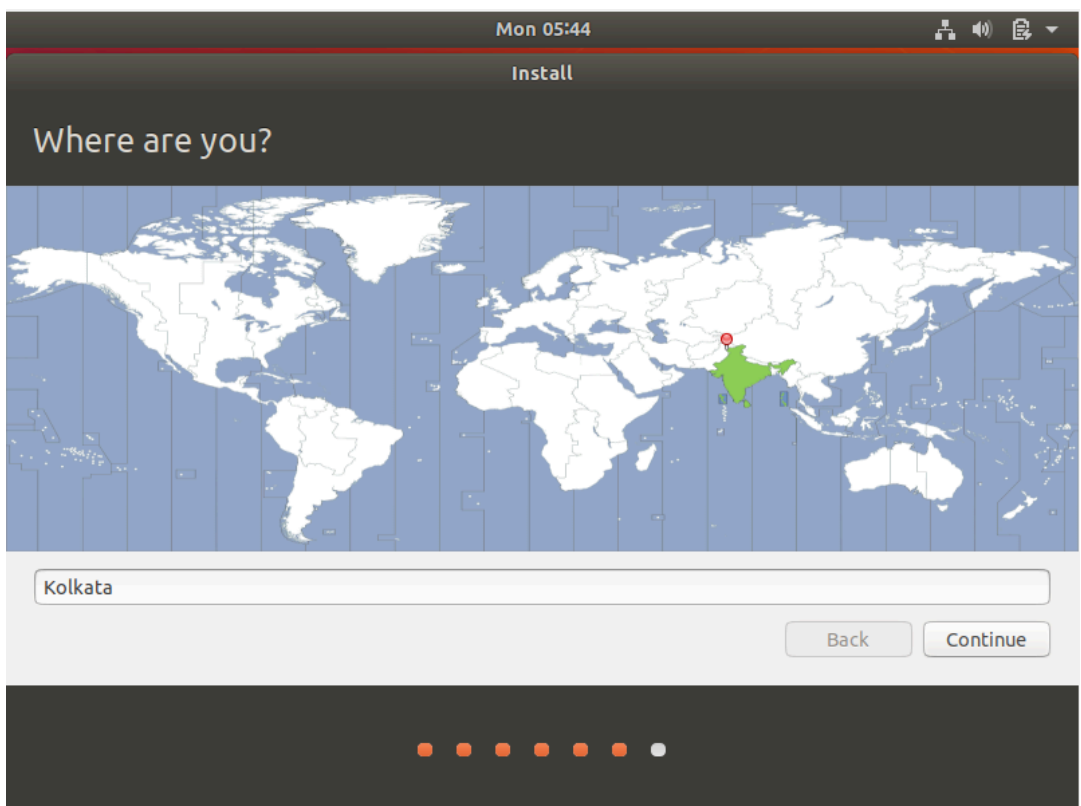
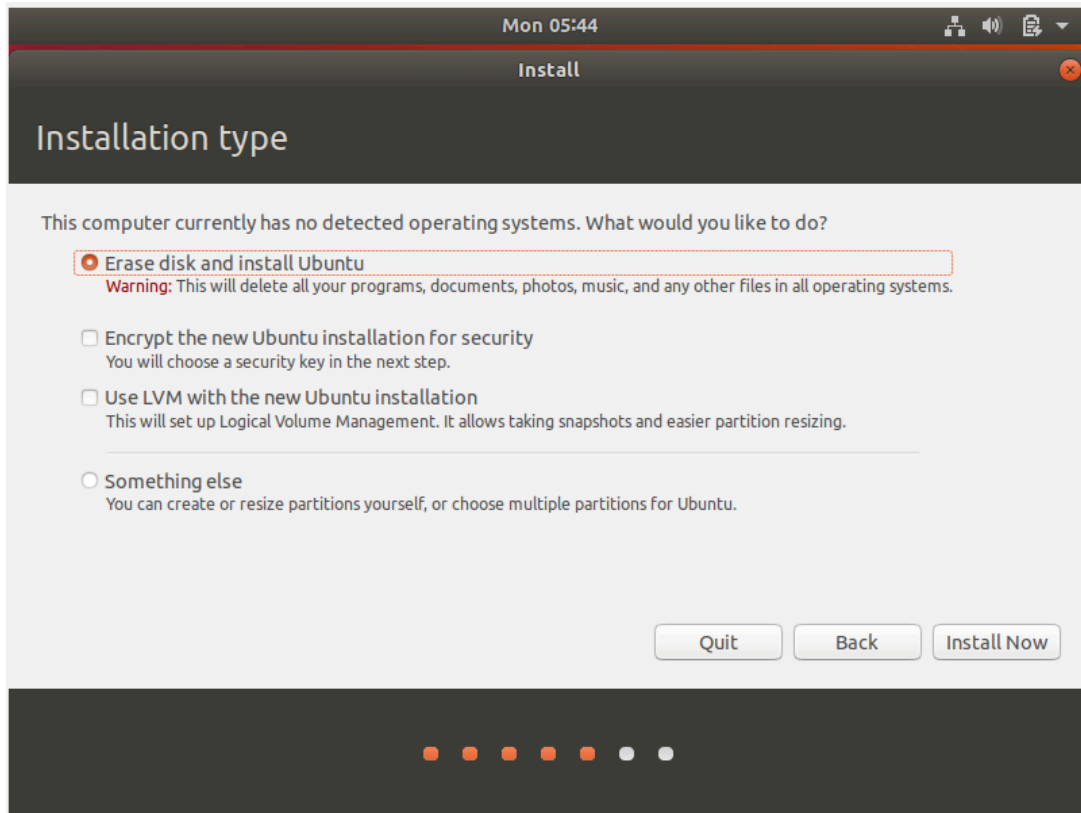
18. Click in Install Ubuntu

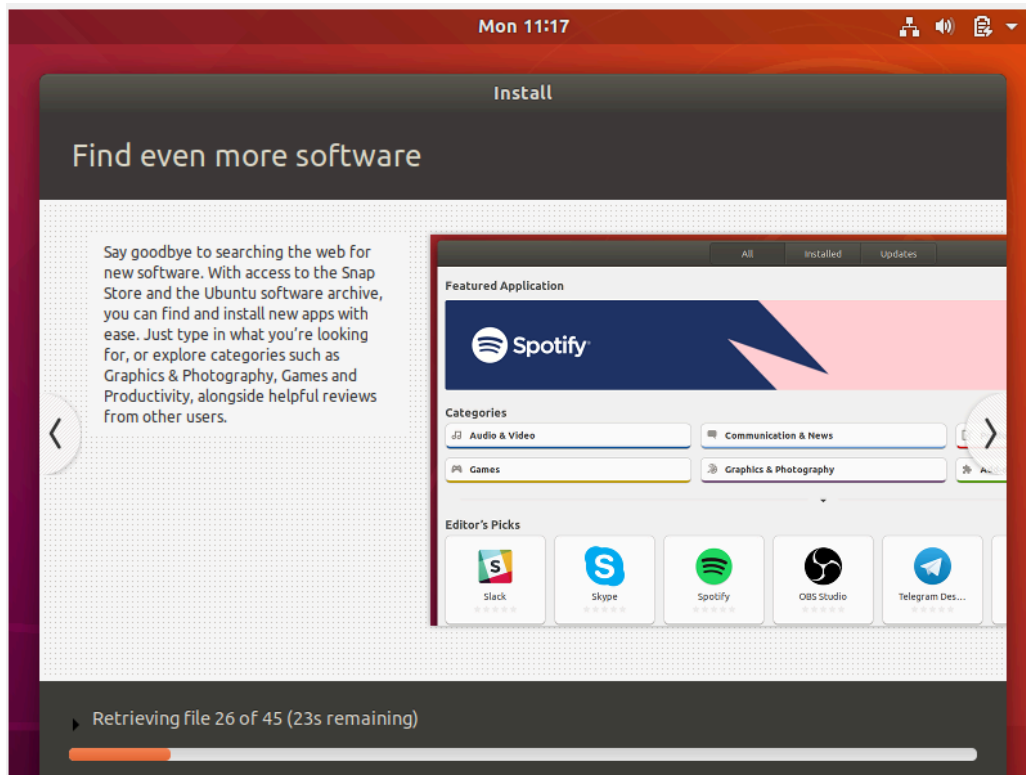
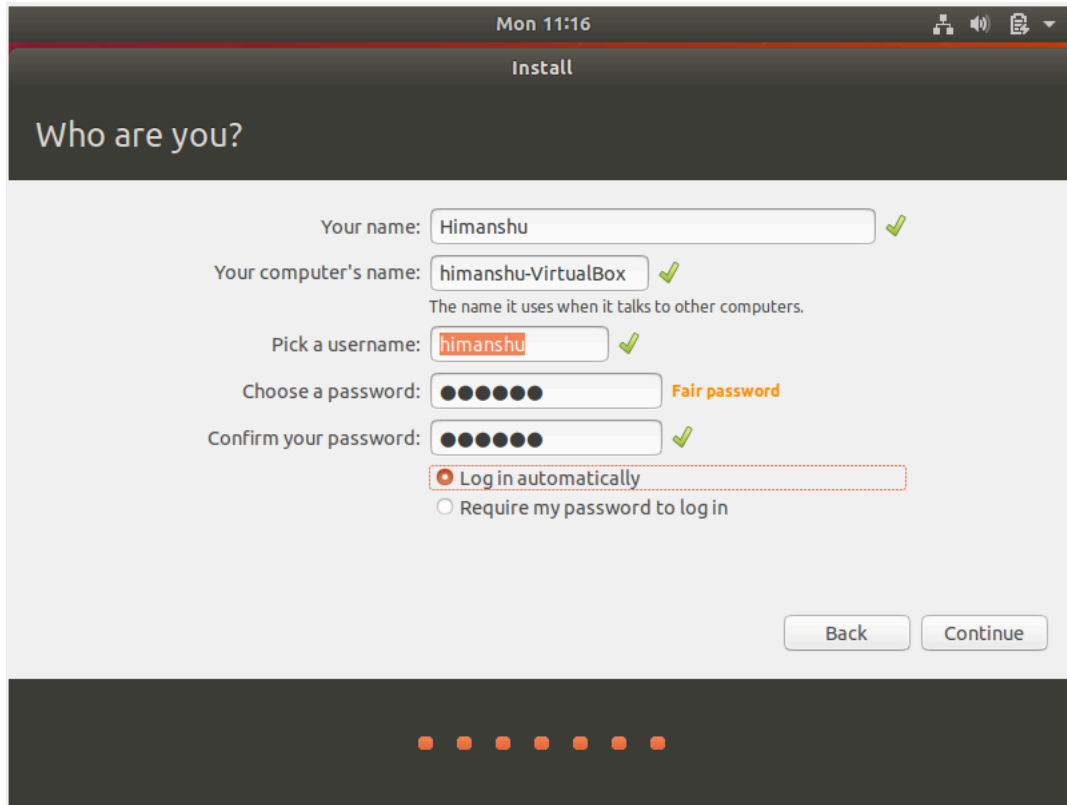


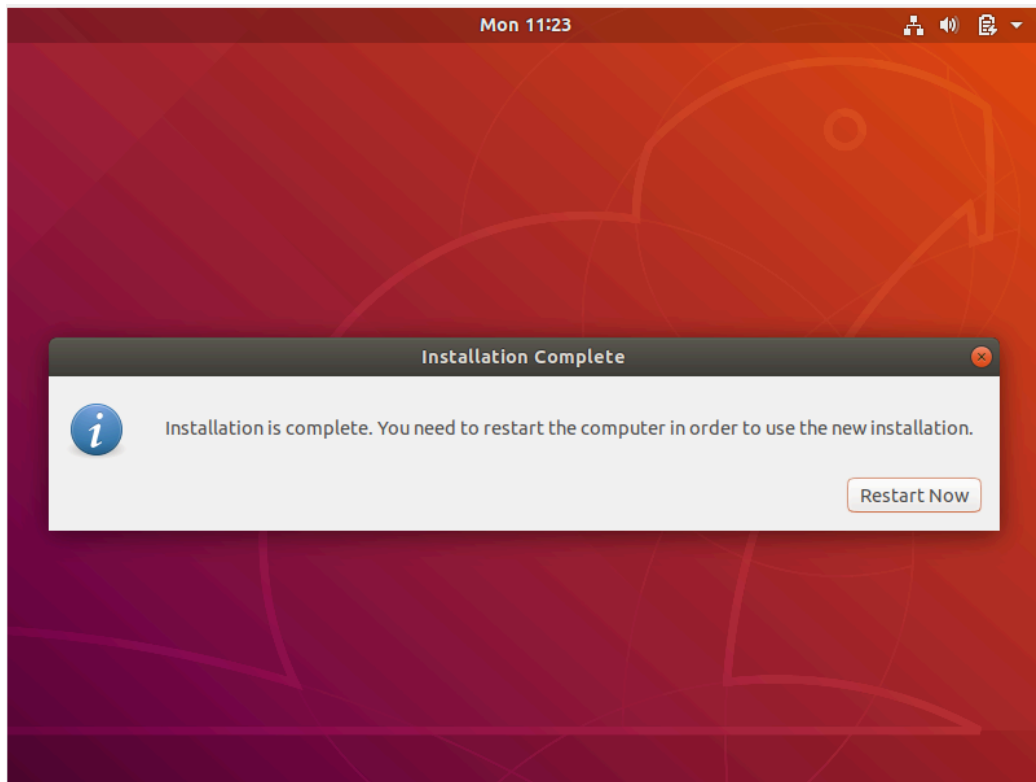
19. Go through the installation wizard for installation.
Below screens shows basic installation instructions.



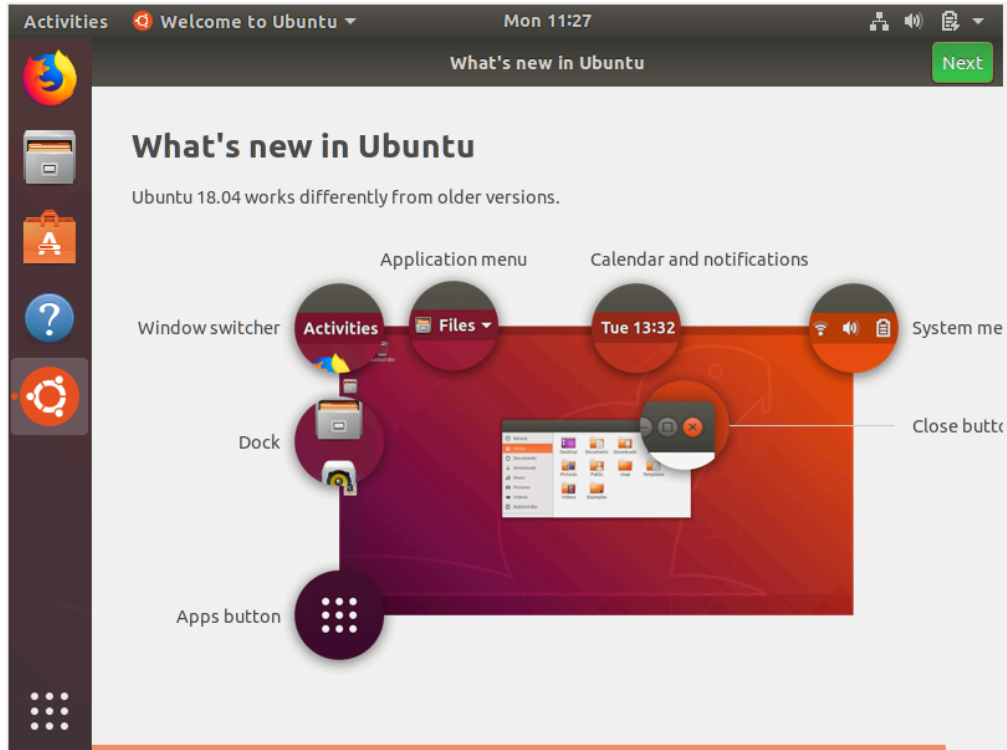




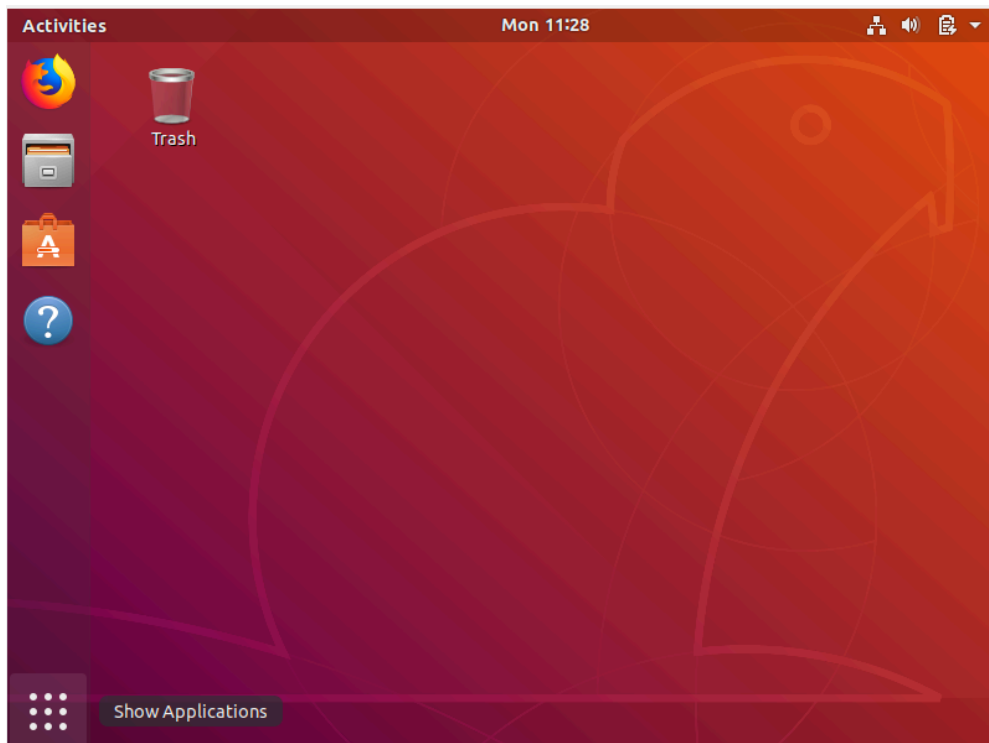




At last you can click enter after restarting the machine.



20. Open Terminal from the bottom left corner of the screen



21. Type the following command in the window to update packages in OS.

sudo apt-get update

```
himanshu@himanshu-VirtualBox:~$ sudo apt-get update
[sudo] password for himanshu:
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
```

22. Type following command to install GCC.

sudo apt-get install gcc

```
himanshu@himanshu-VirtualBox:~$ sudo apt-get install gcc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpp cpp-7 gcc-7 gcc-7-base gcc-8-base libasan4 libatomic1 libc-dev-bin
  libc6 libc6-dbg libc6-dev libcc1-0 libcilkrts5 libgcc-7-dev libgcc1
  libgomp1 libitm1 liblsan0 libmpx2 libquadmath0 libstdc++6 libtsan0
  libubsan0 linux-libc-dev manpages-dev
Suggested packages:
  cpp-doc gcc-7-locales gcc-multilib make autoconf automake libtool flex
  bison gcc-doc gcc-7-multilib gcc-7-doc libgcc1-dbg libgomp1-dbg libitm1-dbg
  libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg
  libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg glibc-doc
The following NEW packages will be installed:
  gcc gcc-7 libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5
  libgcc-7-dev libitm1 liblsan0 libmpx2 libquadmath0 libtsan0 libubsan0
  linux-libc-dev manpages-dev
The following packages will be upgraded:
  cpp cpp-7 gcc-7-base gcc-8-base libc6 libc6-dbg libcc1-0 libgcc1 libgomp1
  libstdc++6
10 upgraded, 16 newly installed, 0 to remove and 608 not upgraded.
Need to get 18.7 MB/36.0 MB of archives.
After this operation, 76.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```


23. To check if the installation of GCC is completed successfully you can use the following command.

`gcc -v`

```
himanshu@himanshu-VirtualBox:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
```

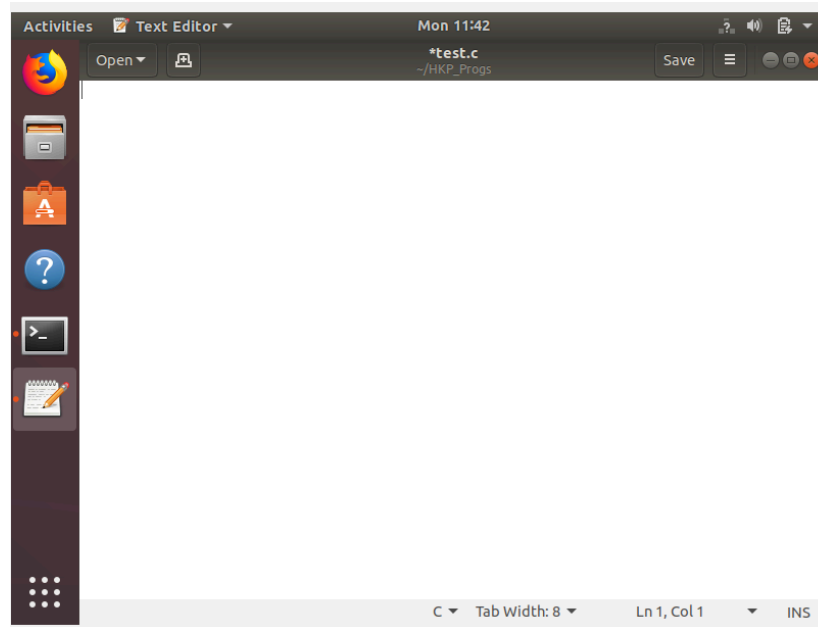
24. You can create your working directory and switch to it.

```
himanshu@himanshu-VirtualBox:~$ mkdir HKP_Progs
himanshu@himanshu-VirtualBox:~$ cd HKP_Progs/
```

25. To create any C program you can use editors for example gedit.

The following command will open a new file called test.c for editing.

```
himanshu@himanshu-VirtualBox:~/HKP_Progs$ gedit test.c &
[1] 6318
```



26. Type your program here.

```
Open ▾ *test.c ~/HKP_Progs Save ≡ ⌵ ⌵ ⌵ ⌵  
  
#include <stdio.h>  
  
int main()  
{  
    printf("Hello Students\n");  
    return 1;  
}
```

27. To compile the program in GCC you can use the following command.

```
gcc -o test test.c
```

This will compile the test.c file and name the store the compiled version in the name test.

To run the file use the following command.

```
./test
```

```
himanshu@himanshu-VirtualBox:~/HKP_Progs$ gcc -o test test.c  
himanshu@himanshu-VirtualBox:~/HKP_Progs$ ./test  
Hello Students  
himanshu@himanshu-VirtualBox:~/HKP_Progs$
```