**3. Use gcc to compile c-programs. Split the programs into different modules and create an application using the make command**

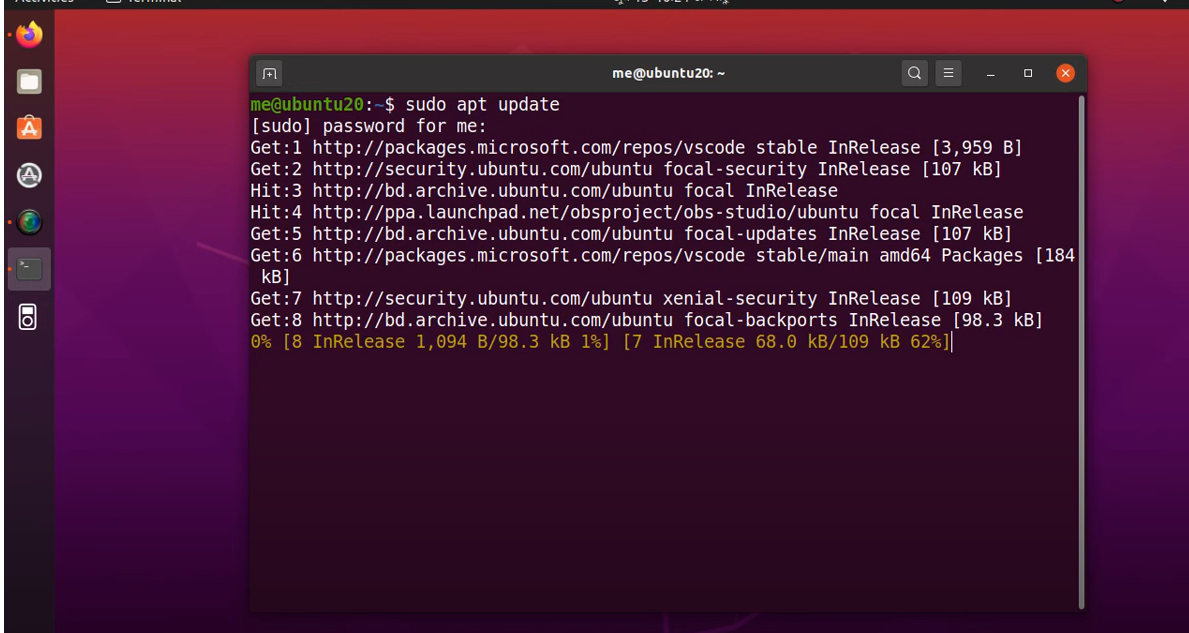
**3.1 Steps to install gcc compiler**

**Step 1:**

Login to the VM

Open the terminal and type the following command and the screen shows the output

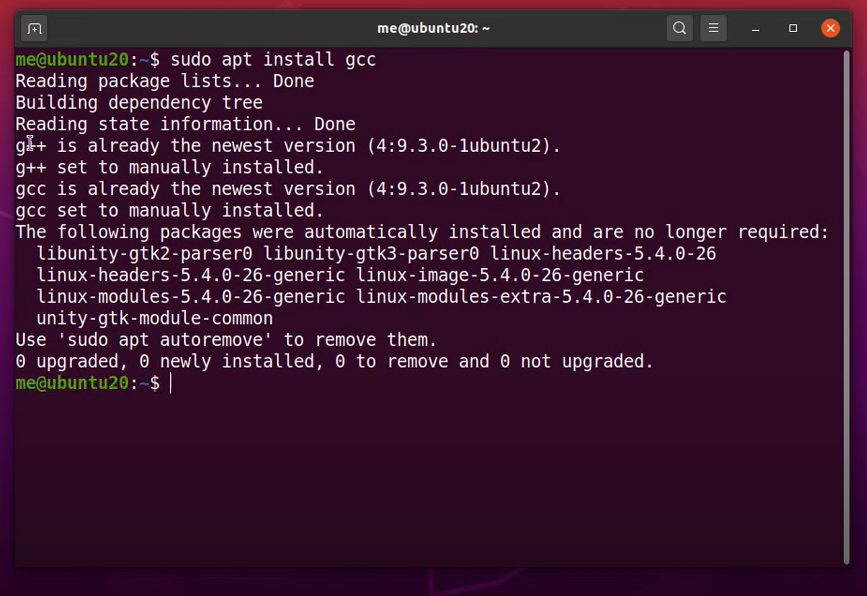
|  |
| --- |
| **$sudo apt update** |



**Step 2:**

Run the next command

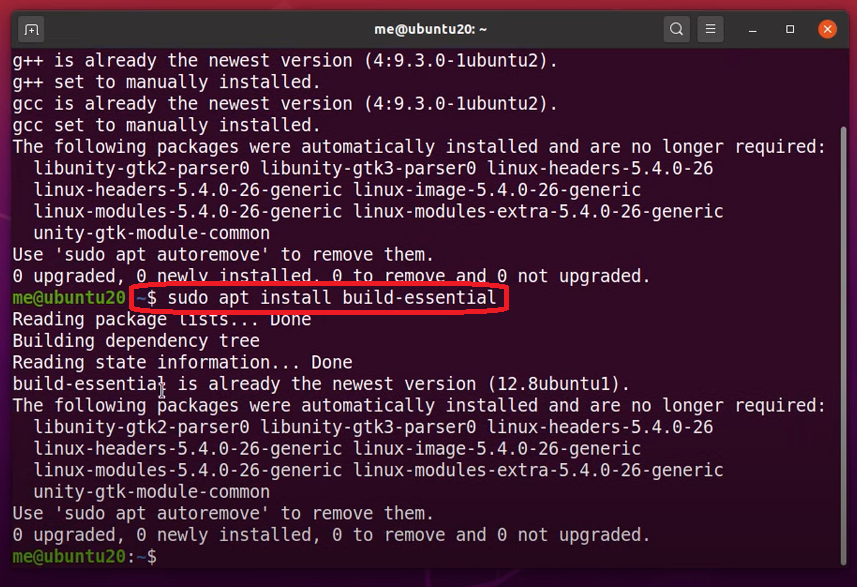
|  |
| --- |
| **$ sudo apt install gcc** |



**Step 3:**

Run the command

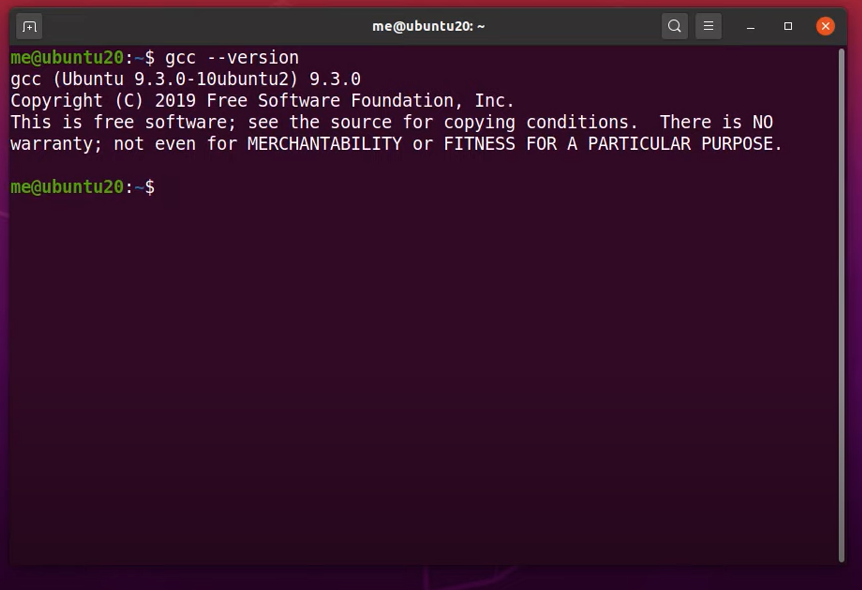
|  |
| --- |
| **$ sudo apt install build-essential** |



**Step 4:**

Check for successful installation by executing the command

|  |
| --- |
| **$ gcc - -version** |



**3.2 Splitting the programs into different modules and creating an application using the make command**

**Step 1:**

Create the following files with specified names and content using the editor in Ubuntu

|  |
| --- |
| **File name: mainp.c**  #include <stdio.h>  #include “functions.h”  int main()  {  display\_hello();  printf(“\n\n The factorial of 5 is %d”, factorial(5));  return 0;  } |
| **File name: functions.h**  void disp\_hello();  int factorial(int n); |
| **File name: factfunction.c**  #include <functions.h>  int factorial(int n)  {  if (n!=1)  return(n \* factorial(n-1));  else  return 1;  } |
| **File name: dispfunction.c**  #include <stdio.h>  #include “functions.h”  void disp\_hello()  {  printf(“\nHello! Welcome!\n\n”);  } |

**Step 2:**

Create a file to write code to execute **make** command as follows

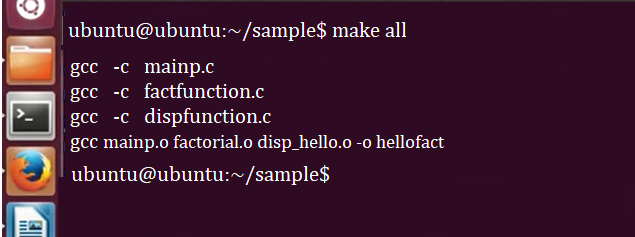
|  |  |
| --- | --- |
| **File name: Makefile** | |
| **Source code** | **Meaning** |
| #Variable declaration  CC=gcc  CFLAGS=-c -Wall  all: a.out  a,out: main1.o sum1.o  $ (CC) main1.o sum1.o -o a.out  Main1.o: main1.c  $(CC) $(CFLAGS) main1.c  Sum1.o: sum1.c  $(CC) $(CFLAGS) sum1.c  clean:  rm -rf \*o hellofact | Comment statement  CC is a variable that stores the type of compiler  CFLAGS is also a variable that stores -c is for compilation - Wall is for giving the warning  For all the created files, specify the name of the executable file. Here it is hellofact  This line describes the dependencies for the target file hellofact  mainp.c file is compiled and the executable code is stored in mainp.o  The same is happened here also  It removes all the files created with extension\*.o |

**Step 3:**

Run the following command in the terminal

|  |
| --- |
| **$ make all** |

The output of the above command is like as shown

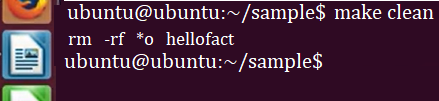


**Step 4:**

If we want to remove all the created object files with extension \*.o, run the command

|  |
| --- |
| **$ make clean** |

The output is as follows



**Reference:**

* Open this website [www.gnu.org](http://www.gnu.org) in the browser
* In the home page search for ‘makefile’
* Click the second link ‘GNU make’

Errors of sudo not root user:

**Su root**

**Apt-get install sudo –y**

**Su root**

**Adduser vboxuser sudo**

**Chmod 0440 /etc/sudoers**

**Reboot**

Makefile: is the file which automate Build of projects files

Instead of running all files separately and execute

We create make file (Makefile) which includes creation of objects files and executables file , With one single command we can run entire project

Target : dependencies

TabSpace Action

Main1.o : main1.c

gcc –c main1.c

sum1.o: sum1.c

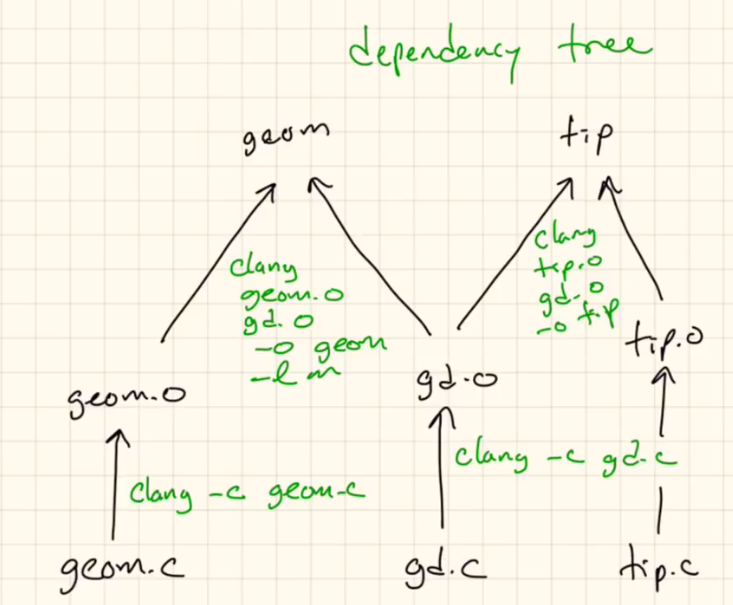
gcc –c sum1.c

All : a.out

a.out: main1.o sum1.o

gcc main1.o

gcc sum1.o



Create 3 files main.c, sum1.c and Makefile , and execute make command

File1 : main.c

File2: sum1.c

File3: Makefile

Main.c

#include<stdio.h>

Int main()

{

Int a = 100;

Int b =20;

Int c;

C = sum(a,b)

Printf(“result is %d “, c);

Return 0;

}

Sum1.c

Int sum(int xx, int yy)

{

Int zz ;

Zz = xx + yy;

Return zz;

}

Makefile

All:a.out

a.out: main.o sum1.o

gcc main.o sum1.o –o a.out

main.o:main.c

gcc –c main.c

sum1.o:sum1.c

gcc –c sum1.c

To save escape : wq ====

And to execute ====== make all or make