COLLEGE OF ENGINEERING CHENGANNUR

CS232

FREE AND OPEN SOURCE SOFTWARE LAB RECORD

Submitted by

Name *Joel Joseph*

Class *S4 D*

Roll No. *29*

Certificate

Name :　　　　　　　　　　　　　　　　Class :

Roll No :　　　　　　　　　　　　　　　Exam No :

This is certified to be the bonafide record of practical work done in Free and Open Source Software as per Syllabus of class .... in the Lab during the academic year 20　/20

....　　　　　　　　　　　　　　................

Teacher In-charge　　　　　　　　　　　*Head of Dept.*

....　　　　　　　　　　　　　　................

Examiner's Signature　　　　　　　　　*Principal*

Date : ....　　　　　　　　　*Institution Rubber Stamp*

TABLE OF CONTENTS:-

1) LINUX COMMANDS

This experiment consists of basic linux commands. It gave a brief introduction to what the different linux commands are.

Some of the commands I tried out in this experiment were:-

1) ls:- this command is used to just list out the different files and directories present in the working directory.
2) ls-l :- this command long lists a file or gives details such as the file permissions, date of modification, directory name etc.
3) cd:- cd command is used to change the current directory.
4) rm:- it's to rename a file.
5) mkdir:- used to make a new directory.
6) cat>filename:- it's used to make a new file and add contents to it.
7) cat -n filename:- this numbers the contents in the mentioned file.
8) cat:- displays contents in a file.
9) cat f1>>f2:- this concatenates contents of f1 to f2.
10) man:- this lists out all information regarding a command.
11) free:- displays the amount of free and used memory in the system.
12) grep:- used to search something in a file.
13) sort:- sorts the file we want.

This experiment gave me an idea of what all basic commands are to be used for the experiments that followed.

2) SHELL SCRIPTING

The aim of this experiment was to calculate the sgpa of cs students in S1 & S2 and use it to finally calculate their cgpa using shell script.

First, the results of s1 and s2 were downloaded.

1) pdfoftext:- the results were in pdf format. Using this command, it was converted into text file.

2) Using grep command, the results of cs students were taken.

3) sed:- using sed command, the grades were converted into its corresponding grade point.

4) awk:- using this command, the sgpa was calculated by multiplying the grade point with the credits corresponding to each subject.

5) Then the results of d batch were sorted out from the complete cs results by comparing the reg no of d batch students with the reg no of entire cs results.

6) The same steps were done for S2.

7) Then both the sgpa for s1 and s2 were used to finally calculate the cgpa.

8) All the details including reg no, name, sgpa's of s1, s2 and cgpa were added to a single file.

Commands used:-

1) pdftotext:- to convert pdf to text file.

2) grep:- to separate out cs results.

3) awk:- to calculate the sgpa and cgpa.

4) sed:- to convert the grades to its corresponding grade points.

5) paste:- to paste the different files to a new file.

6) cat:- to display the final text file.

3) IF CONFIGURATION

In this experiment, IF/ Interface Configuration was done. The basic need of IF configuration in linux is to assign an ip address and netmask to an interface or to disable or enable a given interface.

This experiment was done using a hub to create a interface between 2 to 3 system networks which were connected to the hub at the same time.

The command used for configuration was ifconfig.

By using ifconfig command, all information about all network interfaces currently in operation were displayed.

Then, ifconfig eth0 was used to display configuration of the first ethernet interface with ip 192.168.0.90.

Then, the network mask for this interface was configured using netmask.

Using ping command, the network connectivity between the connected system network.

Finally, using ifconfig down command, the interface was brought down/closed.

Commands Used:-

1) ifconfig:- to configure the interface.

2) ifconfig eth0:- to display first interface configuration.

3) netmask:- to configure the network mask for the interface.

4) lo:- loopback interface, ie., a special interface system uses to communicate with itself.

5) ping:- to check the network connectivity.

6) ifconfig down:- to close the interface.

4) SSH, SCP AND RSYNC

In this experiment, the usage of ssh, scp and rsync commands were focused.

Using the ssh command, a secure connection to a ssh server was enabled. After the ssh server was created, files could be transferred securely to the server. Ssh command helps in connecting to a remote server/system. First, it was checked whether the connection was actually made. This was done by logging into this server which happens automatically as we apply the command for ssh server.

Then, for copying files into the server the scp command was used. Scp command enables the user to copy files from the host system to the server as well as back to the host system to the server.

Rsync was used to efficiently sync the files and directories between the host and the remote server or between 2 hosts.

Rsync also enables fast transfer of files locally over a network as it used a special delta transfer algorithm.

But scp is more secure as compared to rsync.

This experiment taught me how to create a client server and securely transfer as well sync the files over the network.

Commands used:-

1) ssh:- to create the ssh server.

2) scp:- to copy the files between the system and the server and vice versa.

3) rsync:- to efficiently sync the files and directories between the host and server or b/w 2 hosts.

4) ls:- to check whether the copied files are actually present in the server.

5) ifconfig:- to know the ip address of the host system.

5) FTP USAGE

The aim of this experiment was to get an idea of how to use an ftp server. FTP means file transfer protocol. It's used to transfer files between a client and a server on a computer network. Due to security issues, we use sftp instead of ftp which secure file transfer protocol.

The steps involved in this experiment were:

1) A FTP connection had to be established by logging onto the server:- To connect to a ftp server, first we had to log in to the server using the ip address and domain name. sftp ip address@domain name was used for this. Then we had to enter our password. Thus logged in.
2) After we were logged in to the server, we can work with directories, change the current directory, list out the files in the server etc.
3) Files were transferred from the client to the server using put command and using get command, we retrieved the files in the server in the client.
4) After we were done with transferring files and retrieving files, we exited the server.

Commands Used:-

1) sftp:- to securely log on the server.
2) ls:- to list out the files in the server.
3) cd:- to change the directory.
4) pwd:- to display the current working directory.
5) lpwd:- to display the local working directory.
6) put:- to transfer the files to the server.
7) get:- to retrieve the files from the server.
8) goodbye:- to exit from the server.

6) OS INSTALLATION

The first thing that users do when they want to try out a new operating system is to install that particular os they want. In this experiment, we learned how to install linux in our system.

I tried out the installation of openSUSE.

Steps involved were:-

1) First, the installation DVD drive was inserted into the drive. Then system was rebooted and entered into the bios.
2) On the boot screen the installation option was entered which loaded the openSUSE installer.
3) Keyboard and Language layouts were selected as our desire and license agreement was signed.
4) Next, the partitioning was done. By default, openSUSE had a partition proposal. We could select that if we want or we could select manually.
5) The time zones were set.
6) Next, the GUI was set. There were different patterns of GUI and we could select anyone of that or we could use the default GUI of openSUSE.
7) A user had to be created at this stage. By default root user passwd was the same as that of the username. So to add more security, we changed the password to our liking. Thus, a new user was created.
8) Next, the installation settings were configured and no changes could be done after we commit the changes in this stage.
9) Finally, the actual installation was done. Then, the system was rebooted and thus the os was successfully installed.

7) SETTING UP HTTP AND FTP SERVERS

This experiment consists of setting up a http and ftp server of our own.

I made a simple http server which displays my name and shows a scrolling text welcome to my http server.

First a http server was created using nginx. The server was then made to run. Then the html file of the server was created where the html code for displaying my name and the scrolling text was added. Thus a simple http server was complete. The server was run in localhost.

A FTP server was already tried in a previous experiment, but in this experiment a FTP server of our own was created. ie., we ourselves make a server with a username and password. We use vsftpd which is a FTP server for Unix like systems such as linux. It's the default ftp server for Ubuntu, Fedora and some other Linux distros.

Using vsftpd first, we connected to a localhost and then logged on to a server using the username which is the local system name.

After login, we could do whatever we did in the previous ftp experiment.

Some of the Commands used :-

a) HTTP:-
   1) apt-get install nginx:- to install nginx
   2) systemctl start nginx:- to start the server.
   3) gedit:- to edit the html file.
   4) Systemctl restart nginx:- to restart the server.
b) FTP:-
   1) apt-get install vsftpd:- to install vsftpd.
   2) vsftpd:- to start the ftp server.
   3) sftp localhost:- to connect to a localhost.

8) FURTHER TASKS

This experiment consisted of 4 different tasks. They were package management, perl, LAMP stack and Kernel Compilation.

a) Package Management:- This task was to actually get an idea of what all different packages were present in linux and how it could be installed, updated, deleted etc.

   Packages are type of archives which contains computer programs and additional files that come under a particular application.

   Commands used:-

   1) apt-get install :- to install the packages we want.
   2) apt-get update:- to update the available packages and their versions.
   3) apt-get remove:- to remove the packages.
   4) apt-get upgrade:- to upgrade/ install newer versions of the packages we have.

b) Perl:- perl is a open source software programming language used for text manipulation and now used widely for GUI development, network programming etc.
   perl file.pl is used to run the program written in perl language.

c) LAMP stack:- LAMP is set of open source software that can be used to create websites and web applications. LAMP consists of Linux as its operating system, Apache HTTP server as its server, MySQL as the relational database management system and PHP as its programming language. I made a lamp server that will calculate the sum of 2 numbers provided.

   Some of the Commands used:-
   1) apt install :- to install apache http server, mysql, php, mariadb server.
   2) systemctl enable apache2:- to enable apache.
   3) systemctl start apache2:- to start the server.
   4) gedit:- to edit the php file.

d) Kernel Compilation:- Kernel is the essential center of an OS. It acts as an interface between os and hardware. Kernel compilation is required to actually update the kernel version that’s currently there. This experiment takes nearly 2 hrs to complete. ie., it takes nearly 2 hrs for the kernel to be completely finish its compilation.

Some of the commands used:-

1) wget:- to download files from the internet.
2) tar:- to extract archived files.
3) make:- it’s a command to compile those pieces of a program that’s needed to be recompiled.

9) OWN Web Server

In this experiment I created my own web server using html where I could upload contents which included the data about all the experiments done. The name of task, upload date, code link, the description regarding the experiment and finally the result/link showing the output.

My web server was created using html, the code of the server was in a html file index.html.

The html file contained the html code for the attributes and the different contents that goes with each attribute. It's a table which holds all this info. Links to the different tasks were linked from github where I had already uploaded every experiment I completed.

My server could be accessed as http://192.168.0.30/cs17d/cs17d29 in a given area. But, for public, they have to access it as http://14.139.189.217/cs17d/cs17d29.

The commands that are used in this experiment are the html codes such as <body>, <head>, <a href, <td, <table width etc.

MAIN EXPERIMENT: SYLLABUS EXPERIMENT 16

Task name: GUI Programming: Create scientific calculator – using any one of Gambas, GTK, QT.

I made a scientific calculator using qt.

What is QT:-

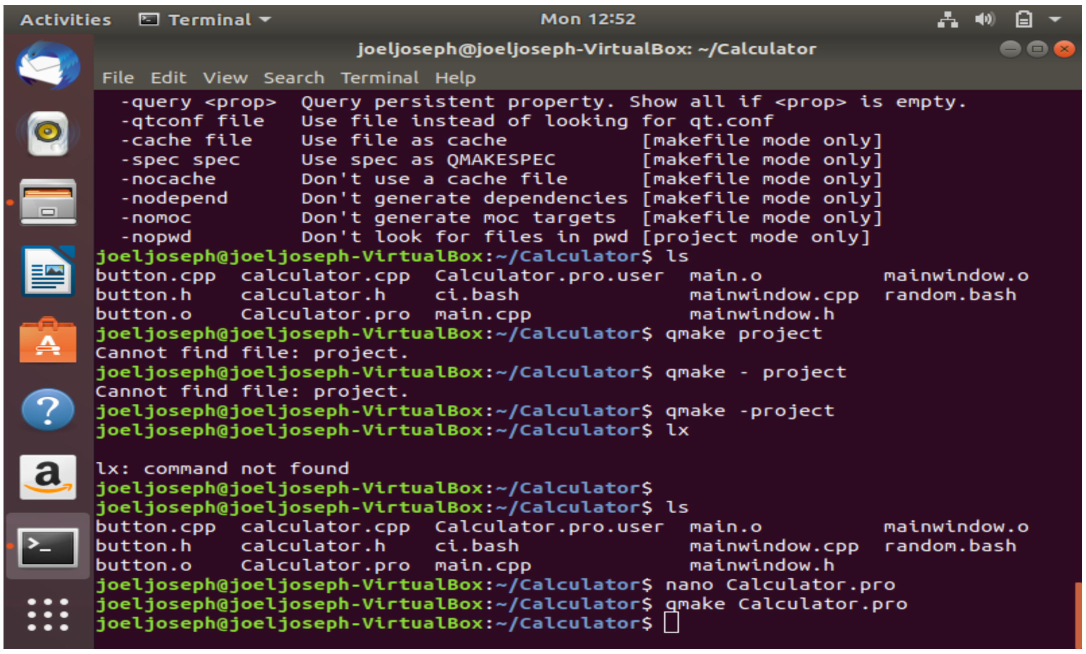
QT is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while having the power and speed of native applications.

Qt is used mainly for developing application software with graphical user interfaces (GUIs); however, programs without a GUI can be developed, such as command-line tools and consoles for servers. An example of a non-GUI program using Qt is the Catalyst web framework. GUI programs created with Qt can have a native-looking interface, in which cases Qt is classified as a widget toolkit.

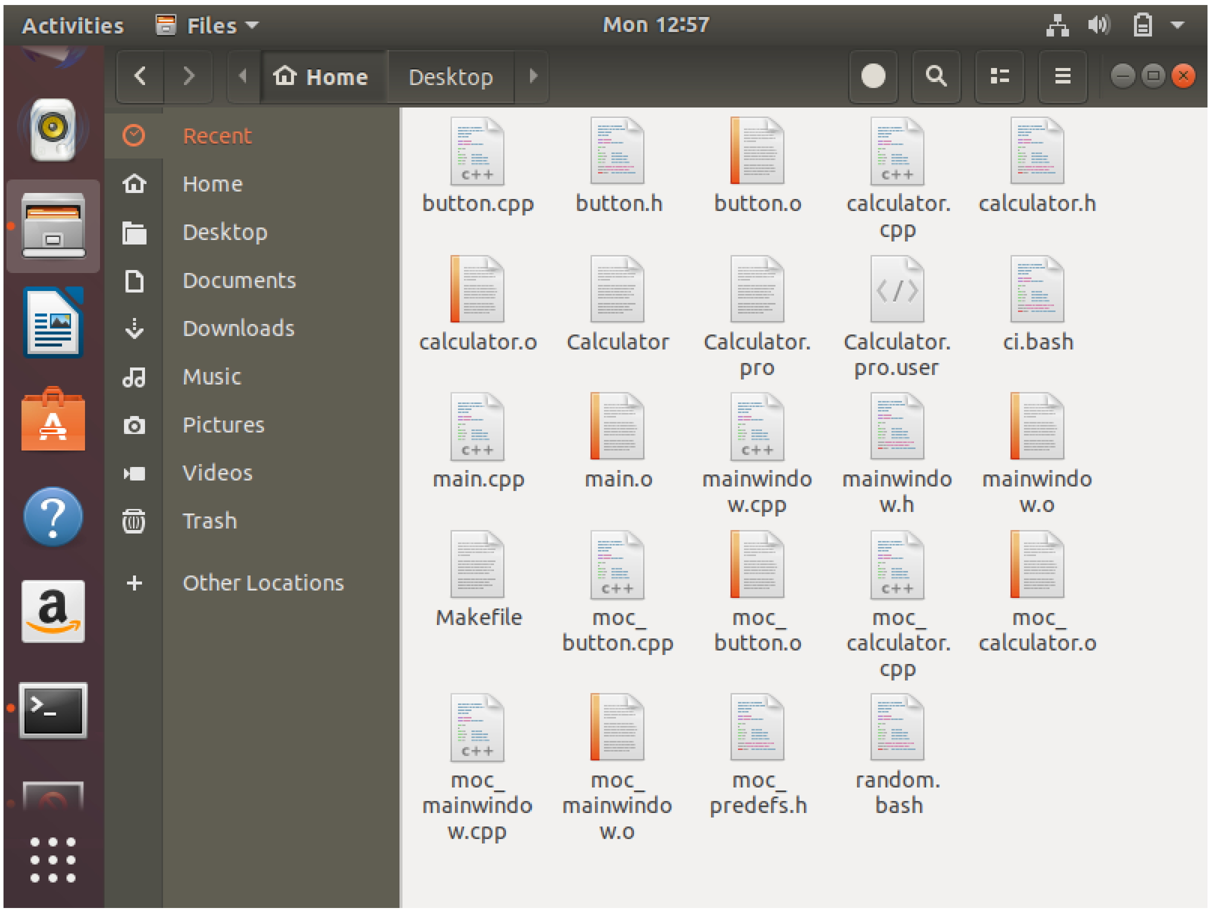
Qt uses standard C++ with extensions including signals and slots that simplifies handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly. Qt supports many compilers, including the GCC C++ compiler and the Visual Studio suite. Qt also provides Qt Quick, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, although logic can be written with native code as well to achieve the best possible performance. Qt can be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms.

STEPS INVOLVED:-

1) First qt was installed in ubuntu using commands typed in the terminal.
2) It took some while to complete the installation, later the essential files and packages needed for qt was installed.
3) Then, the gcc compiler and g++ was installed.
4) For first compilation, a directory was created for storing all the files related to this experiment.
5) Then, the main cpp file was created where the main function of the task was written after going to the newly made directory.
6) After the main cpp file was created, the cpp files for the calculator which contained the code for the different operations to be performed in the calculator was made along with its header files calculator.h consisting of important functions. Button.cpp, Button.h and the other needed files were created to add the necessary codes. The code for converting number to text was also added using bash.
7) After all necessary codes were added, it was time to make the project using qmake command which made the project calculator.pro. Then, it had to be compiled using make command.
8) After compilation, the calculator was run using ./project name.
9) Thus, the task was completed.


Activities
Terminal
Mon 12:52
joeljoseph@joeljoseph-VirtualBox: ~/Calculator
File Edit View Search Terminal Help

```
  -query <prop>  Query persistent property. Show all if <prop> is empty.
  -qtconf file   Use file instead of looking for qt.conf
  -cache file    Use file as cache           [makefile mode only]
  -spec spec     Use spec as QMAKESPEC       [makefile mode only]
  -nocache       Don't use a cache file      [makefile mode only]
  -nodepend      Don't generate dependencies [makefile mode only]
  -nomoc         Don't generate moc targets  [makefile mode only]
  -nopwd         Don't look for files in pwd [project mode only]
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ ls
button.cpp  calculator.cpp  Calculator.pro.user  main.o          mainwindow.o
button.h    calculator.h    ci.bash              mainwindow.cpp  random.bash
button.o    Calculator.pro  main.cpp             mainwindow.h
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ qmake project
Cannot find file: project.
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ qmake - project
Cannot find file: project.
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ qmake -project
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ lx

lx: command not found
joeljoseph@joeljoseph-VirtualBox:~/Calculator$
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ ls
button.cpp  calculator.cpp  Calculator.pro.user  main.o          mainwindow.o
button.h    calculator.h    ci.bash              mainwindow.cpp  random.bash
button.o    Calculator.pro  main.cpp             mainwindow.h
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ nano Calculator.pro
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ qmake Calculator.pro
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ 
```


Activities
Files
Mon 12:57
Home
Desktop
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
button.cpp
button.h
button.o
calculator.cpp
calculator.h
calculator.o
Calculator
Calculator.pro
Calculator.pro.user
ci.bash
main.cpp
main.o
mainwindow.cpp
mainwindow.h
mainwindow.o
Makefile
moc_button.cpp
moc_button.o
moc_calculator.cpp
moc_calculator.o
moc_mainwindow.cpp
moc_mainwindow.o
moc_predefs.h
random.bash

CODES:-

The necessary codes were:-

1) main.cpp:- contains the main function.

```
#include <QApplication>

#include "calculator.h"

int main(int argc, char *argv[])
{
QApplication app(argc, argv);
Calculator calc;
#if defined(Q_OS_SYMBIAN)
calc.showMaximized();
#else
calc.show();
#endif
return app.exec();
        }
```

2) button.cpp :- contained the code for the button and its operations. button.h contained the functions needed for it.

```
#include <QtGui>

#include "button.h"

Button::Button(const QString &text, QWidget *parent)
: QToolButton(parent)
{
setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
setText(text);
}
```

```
QSize Button::sizeHint() const
{
QSize size = QToolButton::sizeHint();
size.rheight() += 20;
size.rwidth() = qMax(size.width(), size.height());
return size;
        }
```

3) random.bash:- contains the code for converting the final output in number to text.

```
        #!/bin/bash
say() { local IFS=+;/usr/bin/mplayer -ao alsa -really-quiet -noconsolecontrols
"http://translate.google.com/translate_tts?tl=en&q=$*"; }
        say $*
```

4) calculator.cpp:- contained the code for all the operations and actions to be performed. As it is very big in content, it’s not being added here.
calculator.h:- the functions needed for calculator.

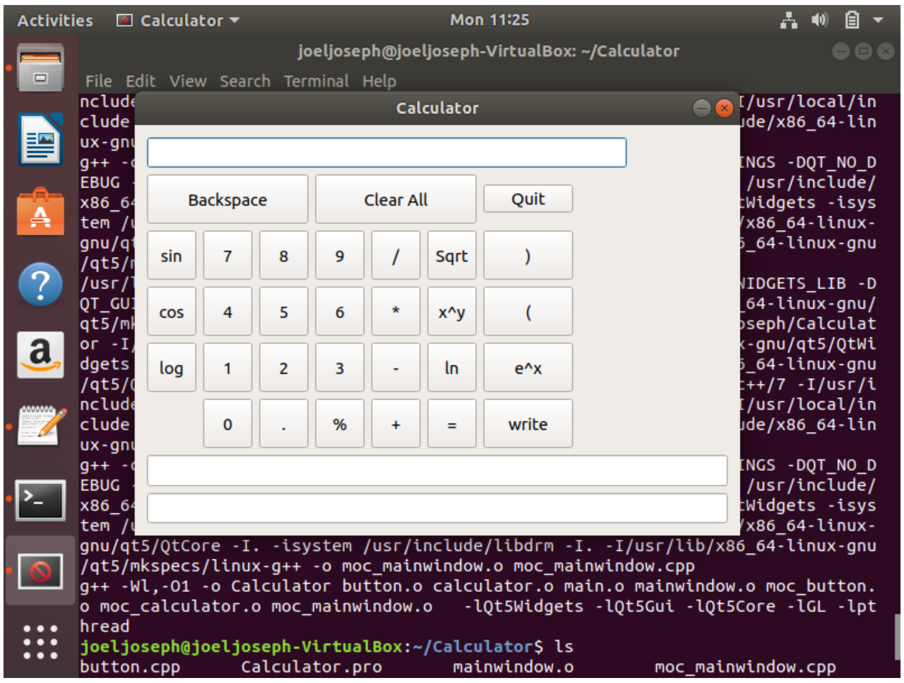
```
#ifndef CALCULATOR_H
#define CALCULATOR_H
#include <string>
#include <QWidget>
using namespace std;

class QLineEdit;

class Button;

class Calculator : public QWidget
{
Q_OBJECT

public:
Calculator(QWidget *parent = 0);
private slots:
void digitClicked();
void quit();
void unaryOperatorClicked();
void equalClicked();
void writeClicked();
void pointClicked();
void backspaceClicked();
void clearAll();

private:
Button *createButton(const QString &text, const char *member);
void abortOperation();
bool calculate(double rightOperand, const QString &pendingOperator);
bool HelperConvertNumberToText(int num, char *buf, int len);
bool ConvertNumberToText(int num, char *buf, int len);
string tobedisplayed(int x);
double sumInMemory;
double sumSoFar;
double factorSoFar;
QString pendingAdditiveOperator;
QString pendingMultiplicativeOperator;
bool waitingForOperand;

QLineEdit *display;
QLineEdit *display1;
QLineEdit *display2;

enum { NumDigitButtons = 10 };
Button *digitButtons[NumDigitButtons];
};

        #endif
```

Some of the commands used in terminal:-

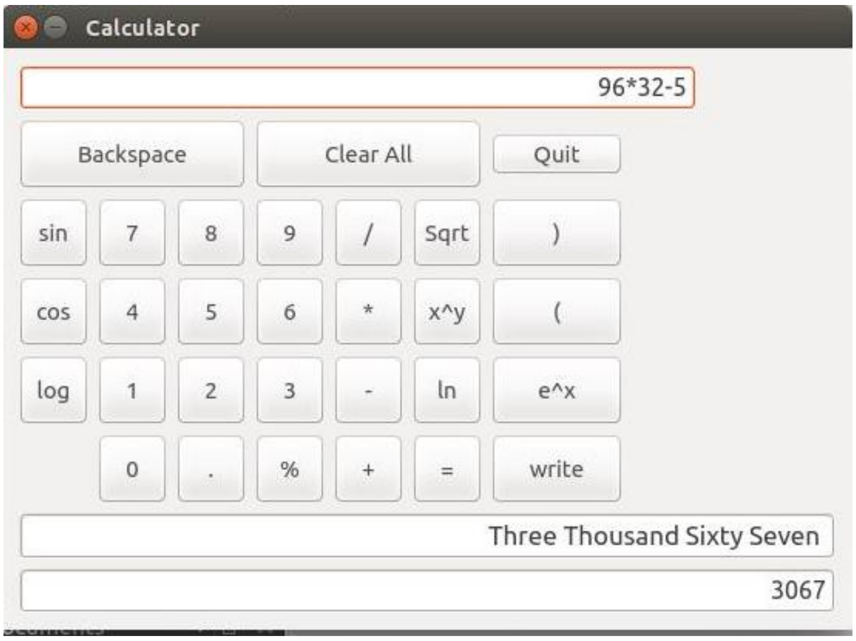
1) sudo apt-get update:- to show the updates available.
2) sudo apt-get install:- to install gcc, g++.
3) Wget:- to download the qt installer from net.
4) Mkdir:- to create directory.
5) qmake- to make the project file.
6) make:- to compile the project
7) ./:- to run the project.
8) cd:- to change directory.

Result:-

The result was obtained by using ./project name. This command runs the project file. Output of this task was a GUI scientific calculator.



```
gnu/qt5/QtCore -I. -isystem /usr/include/libdrm -I. -I/usr/lib/x86_64-linux-gnu
/qt5/mkspecs/linux-g++ -o moc_mainwindow.o moc_mainwindow.cpp
g++ -Wl,-O1 -o Calculator button.o calculator.o main.o mainwindow.o moc_button.
o moc_calculator.o moc_mainwindow.o    -lQt5Widgets -lQt5Gui -lQt5Core -lGL -lpt
hread
joeljoseph@joeljoseph-VirtualBox:~/Calculator$ ls
button.cpp       Calculator.pro       mainwindow.o        moc_mainwindow.cpp
```



```
96*32-5
Three Thousand Sixty Seven
3067
```