Installing compilers using apt command

Open the terminal app and type the following apt/apt-get command:

$ sudo apt update
$ sudo apt upgrade
$ sudo apt install build-essential

OR

$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential

Verify installation

Type the following commands:

$ whereis gcc
$ gcc --version
$ make -v

Related Websites

nixCraft
Ubuntu
C Programs

Console Apps

Tmux:

Sudo apt install tmux

CTRL B + %   To split the current screen vertically
CTRL B + "   To split the current screen horizontally CTRL B + arrow keys   Moves cursor to another screen

tmux
tmux start

Midnight Commander:
Sudo apt install mc

Spreadsheet Calculator:
Sudo apt install sc

ANSI Testing

Colortest:

sudo apt install colortest
Start test: colortest-256   colortest-16

Time to do an update:
Sudo apt update
Also upgrade if need:
Sudo apt upgrade

Install SQLite:
Sudo apt install sqlite3

Openssh Setup

Uninstalled ssh-server, reinstalled it:
sudo apt purge openssh-server
sudo apt install openssh-server

Configure ssh-server: sudo nano /etc/ssh/sshd_config
disallow root login by setting: PermitRootLogin no

Then add a line beneath it that says: AllowUsers yourusername

make sure is set to yes if you want to login using a password: PasswordAuthentication yes

Disable privilege separation by adding/modifying:

UsePrivilegeSeparation no

made sure it's started with: sudo service ssh --full-restart

Connect to your Linux subsystem from Windows using a ssh client like PuTTY

C console programs

Ok, let's get started with some coding. Start in your home directory and make some new directories. This is where your code will go.

$ mkdir code

$ cd code

$ mkdir cdev

$ mkdir cppdev

$ cd cdev

Open up editor to create new C code:

$ vi hi.c

Insert this code by pressing the I key and typing in the following C program.

When done hit the esc key and type in :wq

```
#include<stdio.h>
Int main()
{
   printf("Hi\n");
}
```

Compile your C program by using this command:

$ cc hi.c

Run your program with this command:

$ ./a.out

C++ console programs

You made the directory in the C sections. So if you are in the cdev directory just do a "cd .." to get back to the code directory.

From the code directory:

$ cd cppdev

Or

From the home directory

$ cd code/cppdev

Open editor to create new C++ code:

$ vi hi.cpp

Insert this code by pressing the I key and typing in the following C++ program.
When done hit the esc key and type in :wq

```
#include<iostream>
using namespace std;
// Simple C++
// no class no ooping
// using cout
int main()
{
 cout<<"Hi there\n";
 return 0;
}
```

Compile your C++ program by using this command:

$ g++ hi.cpp

Run your program with this command:

$ ./a.out

There are two more simple C++ programs in the ccpdev. The first one hicolorv1.cpp is just like the hi.cpp but has the Esc code sequence added to it. The second, hicolorv2.cpp has enum type added to it. None of these C++ have any Object Orientated Programming stuff in them. I will make another directory called oop start with simple oop code.

Python

Install python and startup:

```
$ sudo apt-get install python2.7
$ python
```

Or and both

```
$sudo apt-get install python3
$ python
```

Python test:

```
$ python
```

The python2.7 interactive script IDE starts up

```
>>> 5+6
Returns 11
>>> a=3*5
>>> print a
Returns 15
```

cd to code and mkdir pycon and cd pycon
vi assii.py
insert the code below:

```
 c='\033[1;31m'
for a in range(ord('A'), ord('Z')):
    c = c + chr(a);
print c;
```

python assii.py

Do the same for this one:

```
# Hi app in color
print u'\u001b[2J'
print u'\u001b[31mHi there \u001b[0mAppa'
print u'\u001b[1;31mHi there \u001b[0mAppa'
print u'\u001b[32mHi there \u001b[0mAppa'
print u'\u001b[1;32mHi there \u001b[0mAppa'
print u'\u001b[33mHi there \u001b[0mAppa'
print u'\u001b[1;33mHi there \u001b[0mAppa'
print u'\u001b[34mHi there \u001b[0mAppa'
print u'\u001b[1;34mHi there \u001b[0mAppa'
```

ANSI Terminal esc sequence

I will try not to get into too much history at this time. In the late 70s, early 80s I was writing Fortran programs for Automated Test Equipment (ATE). I was having problems getting the results I wanted on some of the plotters and test equipment with the given software libraries. So, I asked Hewlett- Packard about it. They gave me a tape with all the subroutine source code for all of test equipment, plotters, and terminals. The first thing I saw that they all had in common were esc codes being sent to them. I had seen this before with a programable Techtronic oscilloscope. I thought that it was something new. It turns out that almost all manufactures computer peripherals have some esc code sequence. This is how I got started in making my plotters plot better, test equipment measured more accurate and terminal readouts flasher. Even printouts were cooler.

Some of these esc sequence become standard. The ones that I am using in this project is called the ANSI Terminal esc sequence. These esc codes are used on vt100. I will show some simple examples in C/C++ and Python and we will see where we go from there.

==== General text attributes ====

esc code m	Description
"\033[0 m"	Reset all
'\033'[1 m"	bright
"\033[2 m"	dim" attribute
'\033'[3 m	standout
"\033[4 m"	underscore
"\033[5 m"	blink
"\033[7 m"	reverse
'\033'[8 m"	hidden

==== Foreground coloring ====

esc code m	Description
"\033[30 m"	black
"\033[31 m"	red
"\033[32 m"	green
"\033[33 m"	yellow
"\033[34 m"	blue
"\033[35 m"	magenta
"\033[36 m"	cyan
"\033[37 m"	white
"\033[39 m"	default

Some esc sequence code ANSI terminal table:

Text attributes
    0   All attributes off
    1   Bold on
    2   Dim
    3   Standout
    4   Underscore
    5   Blink on
    7   Reverse video on
    8   Concealed on

Foreground colors
    30   Black
    31   Red
    32   Green
    33   Yellow
    34   Blue
    35   Magenta
    36   Cyan
    37   White

Background colors
    40   Black
    41   Red
    42   Green
    43   Yellow
    44   Blue
    45   Magenta
    46   Cyan
    47   White

ANSI escape code
VT100 Programming

Object-oriented programming (OOP)

Ok so there is lots of ways to go and I wanted to try a couple of things C++ 11 that I hand not tried before.

Like, default member data in privet section of a class, and enum class, which is not a class at all. But does

Resolve some problems with enum type. I started off my making several classes for cursor position, text attributes

and screen clearing. I did not want to get into massive C++ oop programming. That not what this project is about.

It is just to show you how to use ANSI Terminal escape sequence. What I really want to do is to have one simple

class that could do all those things. The first thing I found is that I did not have a C++ 11 compliant g++ compiler.

That's ok there a procedure to install one. We will not worry about that at this time.

Games

Space Invaders:

sudo apt install ninvaders

Bastet:
sudo apt install bastet

Pacman4console:
sudo apt install pacman4console

BSDGames:
sudo apt install bsdgames

Just for fun stuff:

That is if you think Artificial Intelligent,
Expert Systems, and Artificial Neural Networks
are cool.

sudo apt install gprolog
mkdir AI
cd AI
mkdir Prolog
mkdir Lisp
mkdir ANN
cd Prolog

vi friends.pl
insert code:

likes(joe, jan).
likes(jan, joe).
likes(don, jan).

friends(X, Y):-likes(X, Y), likes(Y, X).

start prolog:
gprolog
[friends].
Friends(X, Y).

sudo apt install brandy bwbasic