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

Midnight Commander:

Sudo apt install mc

ANSI Testing

Colortest:

sudo apt install colortest

Start test: colortest-256 colortest-16

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;
int main()
{
cout<<"Hi\n";
return 0;
}
```

Compile your C++ program by using this command:

$ g++ hi.cpp

Run your program with this command:

$ ./a.out

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

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 ====

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

==== Foreground coloring ====

ANSI esc code	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

Python color text to terminal methods

Using built-in modules:

Colorama module: shorthand ANSI escape sequence sudo apt install colorama

Python program to print # red text with green background

from colorama import Fore, Back, Style print(Fore.RED + 'some red text') print(Back.GREEN + 'and with a green background')
print(Style.DIM + 'and in dim text') print(Style.RESET_ALL) print('back to normal now')

Python program to print # green text with red background

from colorama import init from termcolor import colored

init()print(colored('Hello,World!','green','on_'))

termcolor module:

sudo apt install termcolor

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-restar

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

Games

Space Invaders:

sudo apt install ninvaders

Bastet:
sudo apt install bastet

Pacman4console:
sudo apt install pacman4console

BSDGames:
sudo apt install bsdgames