How to install the tools necessary for C++11/17 projects

Given we wish build real world GUI applications on Linux based operating systems using C++11/17 **when** we successfully install the `GNU Compiler Collection` (`gcc, g++, make, gdb, ssh`) as our standard set of build tools **then** we can build Unix shell style applications that will run on most of the Popular Linux distributions as well as most Android, iPhone/iOS and Windows 10/11 platforms

Note: Cross-platform GUI compliance like Android and iPhone support?

- first we build Unix shell compliant console applications (with just C++11/17 & the STL)
- then *attach* operating system specific GUI platforms to the console's shared libraries
- popular methods of sharing data such as simple JSON strings make this possible
- this is preferred method for developing software for real world applications

Prerequisites

- How to install Linux

Wish Case

Assuming you have a Linux instance and are currently logged into it (either locally or remotely):

- [ ] Login as dev (unless you are already `logged in as dev`):

  ```
  su dev
  ```

  Note: In the case of a remote Linux box type (unless you are already `logged in remotely as dev`):

  ```
  ssh dev@ip_address
  ```


- [ ] Now install `gcc, g++, make, gdb, ssh` as well as a few other necessary Unix-standard build tools:

  ```
  cd ~
  sudo apt update -y
  sudo apt upgrade -y
  sudo apt install -y build-essential libtool \
  autotools-dev automake pkg-config git clangd \
  cppcheck clang-tidy python3-pip checkinstall \
  gdb xclip openssh-server net-tools zip xterm \
  curl ncdu tidy
  ```

  Note: *CMake/CPM installed seperately* (below) ...

- [ ] Set the following environment variables:

```
cd ~
ANSI_YELLOW="\e[33;1m"
ANSI_RESET="\e[0m"
output=$(hostname -I)
output=($output)
email_address="$(basename $PWD)@${output[0]}"
echo -e
"${ANSI_YELLOW}MY_EMAIL_IS="\"$email_address\""${ANSI_RESET}"
echo -e "${ANSI_YELLOW}MY_NAME_IS="\"$(basename
$PWD)\""${ANSI_RESET}"
```

Note: Copy everything in `yellow` (or `orange`) to the clipboard

  - Setting your own email address is *optional* but not *required* here

- [ ] Now paste **the contents of the clipboard** to the terminal
- [ ] Now make sure these environment variables are **set**:

```
echo $MY_EMAIL_IS
echo $MY_NAME_IS
```

- [ ] The output would look like this (just a different IP address):

```
MY_EMAIL_IS="dev@146.190.130.95"
MY_NAME_IS="dev"
dev@ubuntu-4g2:~$ MY_EMAIL_IS="dev@146.190.130.95"
MY_NAME_IS="dev"
dev@ubuntu-4g2:~$  echo $MY_EMAIL_IS
echo $MY_NAME_IS
dev@146.190.130.95
dev
dev@ubuntu-4g2:~$
```

- [ ] Now configure the git utility:

```
git config --global user.email ${MY_EMAIL_IS}
git config --global user.name ${MY_NAME_IS}
```

- [ ] Now create an SSH key for the Linux box:

```
ssh-keygen -t ed25519 -C ${MY_EMAIL_IS}
```

  - Do **not** supply a passphrase or change the filename

- ☐ Show the SSH public key and remember how to display it when it is needed later on (for SSH authentication)

```
cat ~/.ssh/id_ed25519.pub
```

  - Feel free to copy & paste the `SSH public key` that you see displayed
  - to the `GPG & SSH Keys` section of your Github account (under `Settings`).

- ☐ Now determine the ssh command you'd need to log into this Linux box (once you add the clients SSH key):

```
cd ~
ANSI_YELLOW="\e[33;1m"
ANSI_RESET="\e[0m"
output=$(hostname -I)
output=($output)
ssh_cmd="$(basename $PWD)@${output[0]}"
echo -e "${ANSI_YELLOW}ssh $ssh_cmd${ANSI_RESET}"
echo -e "${ANSI_YELLOW}$ssh_cmd${ANSI_RESET}"
```

Copy & paste whatever displays in `yellow` (or `orange`) to your notes (for *later reference*):

```
ssh dev@146.190.130.95
dev@146.190.130.95
```

PROF

- ☐ Assuming that was successful, install **CMake** (on the new Linux box)

```
 sudo apt install python3.11-venv -y
 python3 -m venv .venv
 source .venv/bin/activate

 sudo apt-get -y install clang-format
 pip install cmake-format
 sudo apt-get update -y
 sudo apt-get install cmake -y
 sudo apt-add-repository universe -y
 sudo apt-get install cmake-extras -y
 mkdir -p ~/.local/bin
```

Note: `CMake`

- is built on top of the Unix standard `make` utility
- it's purpose is to make the `make` syntax more readible
- as well as add more features to `make` capabilities

- [ ] Now edit the startup script with the vi editor:

```
vi ~/.bashrc
```

Note: Are you new to the `vi` editor?

- [ ] `Down arrow` to the bottom of the script
- [ ] `Right arrow` till you are at the end of the line
- [ ] Press `a` to go into `append` mode and press ENTER (twice)
- [ ] Now copy the following to the clipboard:

```
export PATH=$HOME/.local/bin:${PATH}
export CPM_SOURCE_CACHE=$HOME/.cache/CPM
export LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
```

  - Now paste these environment variables into the shell
  - Now press ENTER once or twice (to give it a new line)
  - Now press ESCAPE and then `SHIFT z` button twice

- [ ] Then source it ...

PROF

```
source ~/.bashrc
```

- [ ] Now close any apps (including browsers) and remove all unnecessary files

```
sudo apt-get autoremove -y
sudo apt-get autoclean -y
sudo apt-get clean -y
journalctl --disk-usage
sudo journalctl --vacuum-time=3d
du -h /var/lib/snapd/snaps
rm -rf ~/.cache
```

- [ ] Now remove any unnecessary snap images:

```
  echo Removes old revisions of snaps
  echo CLOSE ALL SNAPS BEFORE RUNNING THIS
  echo Attention: **experimental optimization**
  echo Attention: Have your Linux instance backed up beforehand
  echo Attention: should snap behave strangely resort to the backup
  read -p "Are you sure you want to reset local SNAPS cache: (y/N)?
" name
  if [[ "$name"=="Y" || "$name"=="y" ]];
  then
     echo -e "local SNAPS cache deleted"
     set -eu
     snap list --all | awk '/disabled/{print $1, $3}' |
         while read snapname revision; do
             snap remove "$snapname" --revision="$revision"
     done
  else
     echo -e "local SNAPS cache not deleted"
  fi
```

Optional: This step is just meant to save space

See *Resume/Next Steps* below ...

Alternate Case

client_loop: send disconnect: Broken pipe

You SSH connection went down (part of the Internet experience).

- Just relog in with the same SSH string
- Typically you would just `up arrow` and hit `enter`

Alternate Case

PROF

Something didn't install correctly?

You might need to update/upgrade the system first:

- Go into *sudo* mode:

```
  sudo ls
```

- Update the instance and reboot:

```
  sudo apt update -y
  sudo apt upgrade -y
  sudo apt autoremove -y
```

```
sudo apt autoclean -y
sudo reboot
```

- When rebooted relogin as dev:

```
su dev
```

Alternate Case

Are you new to the vi editor?

- see How to use the VI editor (basic instructions)

Alternate Case

Visual Studio Code (VSC)

In the case that you the Ubuntu instance has x11/GUI access feel free to install VSC at this point time:

```
sudo snap install --classic code # or code-insiders
```

Alternate Case

Visual Studio could not be installed or started (directly)

In certain cases VSC cannot be installed (in the case of no gui support or processor incompatibilies). In this situation a SSH client could be setup.

- see How to remote connect to your Linux box via SSH

PROF

Alternate Case

/home/perry/.local/bin/xsnap.sh: 8: [[: not found

Make sure the first line of the bash script copies over as `#!/bin/bash` and not `#!/bin/bash`

Alternate Case

Dark Theme

In the case where you start up VSC and the title bar portion of the editor is Light coloured and you desire to have it Dark themed merely do this:

1. Open the Settings app, (click on the bottom left menu and type 'Settings')
2. Select Appearance and then click on the Dark theme

Alternate Case

Semi-transparent Terminal boxes

In the case where you would like your Terminal box to have a certain level of transparency:

1. Open a Terminal box, , (click on the bottom left menu and type 'Terminal')
2. On the top left of the screen click on 'Terminal'
3. Select Preferences -> Unnamed -> Colours
4. Deselect 'Use transparency from system theme'
5. Select 'Use transparent background'
6. Adjust the scroll bar to your preferred level of transparency

> In this way you'll be able to see things behind your Terminal box, (comes in handy)

Alternate Case

error: snap "code" is not available on stable for this architecture (arm64) but exists on other architectures (amd64).

In the case of the Apple M1 (and you are running Linux under a VM like Parallels or VirtualBox) what you want to do is connect to the Linux box via it's IP address over SSH.

```
sudo apt install net-tools
ifconfig
```

Get the IP address and after adding the ~/.ssh/id_ed25519.pub key to the ~/.ssh/authorized_keys of the Linux box log into it via ssh protocol

Alternate Case

PROF

Visual Studio Code Extensions

Visual Studio Code will detect whatever language you are using and offer to install extentions automatically. Feel free to allow all recommendations as they appear to the bottom right of the Visual Studio Code environment.

Alternate Case

Install cmake using snap:

```
sudo snap install cmake --classic
```

Alternate Case

Bad CMake executable "/snap/bin/cmake"

cmake has been going through alot of improvemetns and the latest method of installation from the command line provides the 3.21 requirement (see bad cmake executable vscode):

```
snap remove cmake -y
sudo apt-get update -y
sudo apt-get install cmake -y
sudo apt-add-repository universe -y
sudo apt-get install cmake-extras -y
```

Alternate Case

Ubuntu 18.04

Slightly different parameters required

```
sudo apt install -y build-essential libtool autotools-dev automake pkg-config git clang-9 cppcheck clang-tidy python3-pip checkinstall gdb gcc-multilib g++-multilib
```

> In the case you've installed a version of gcc / g++ that doesn't ship by default (such as g++-4.8 on lucid) you'll want to match the version as well:

```
sudo apt-get install gcc-4.8-multilib g++-4.8-multilib
```

Summary

PROF

Now you have installed the development environment and editor for a C++17 project (complete with cmake 3.21 support).

Resume/Next Steps

- How to install injections.io(C++17)

Disclaimer

> That `word` is not to appear anywhere on this page, (except here)