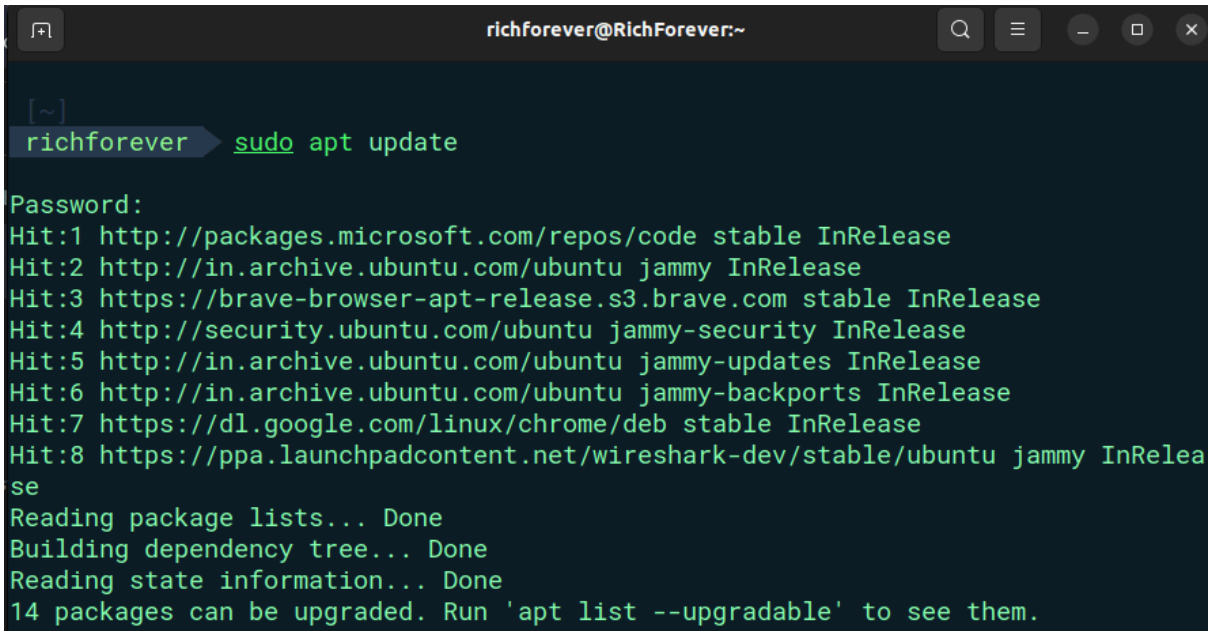
How to install cMake?

Step1) Installation is done considering the **Ubuntu/Debian system**, however, if you have any other destro just check the installation command of your destro.

```
sudo apt update
```

This command refreshes the package information from software repositories on a Debian-based Linux system.



Refreshing Package Information

Step2) Enter the following command to install '**CMake**' :

```
sudo apt install cmake -y
```

We can use this command regardless of -y , but if we specify -y ,then it doesnt asks for "yes/no" in between , it just proceeds considering all permissions as "yes"

```
richforever@RichForever:~

richforever  sudo apt install cmake -y

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cmake-data dh-elpa-helper librhash0
Suggested packages:
  cmake-doc ninja-build cmake-format
The following NEW packages will be installed:
  cmake cmake-data dh-elpa-helper librhash0
0 upgraded, 4 newly installed, 0 to remove and 14 not upgraded.
Need to get 7,058 kB of archives.
After this operation, 31.6 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 librhash0 amd64 1.4.2-1ubuntu1 [125 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 dh-elpa-helper all 2.0.9ubuntu1 [7,610 B]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cmake-data all 3.22.1-1ubuntu1.22.04.1 [1,913 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cmake amd64 3.22.1-1ubuntu1.22.04.1 [5,013 kB]
Fetched 7,058 kB in 13s (527 kB/s)
Selecting previously unselected package librhash0:amd64.
(Reading database ... 276763 files and directories currently installed.)
Preparing to unpack .../librhash0_1.4.2-1ubuntu1_amd64.deb ...
Unpacking librhash0:amd64 (1.4.2-1ubuntu1) ...
Selecting previously unselected package dh-elpa-helper.
Preparing to unpack .../dh-elpa-helper_2.0.9ubuntu1_all.deb ...
Unpacking dh-elpa-helper (2.0.9ubuntu1) ...
Selecting previously unselected package cmake-data.
Preparing to unpack .../cmake-data_3.22.1-1ubuntu1.22.04.1_all.deb ...
Unpacking cmake-data (3.22.1-1ubuntu1.22.04.1) ...
Selecting previously unselected package cmake.
Preparing to unpack .../cmake_3.22.1-1ubuntu1.22.04.1_amd64.deb ...
Unpacking cmake (3.22.1-1ubuntu1.22.04.1) ...
Setting up dh-elpa-helper (2.0.9ubuntu1) ...
Setting up librhash0:amd64 (1.4.2-1ubuntu1) ...
Setting up cmake-data (3.22.1-1ubuntu1.22.04.1) ...
Setting up cmake (3.22.1-1ubuntu1.22.04.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
```

Step3) For checking the installation of the command , we can simply type "**cmake --version**" , if it says "command not found" then its not installed , else if we get some version number then it's correctly installed.

```
cmake  --version
```

Cmake version

How to make a CMake file and executable file of our project ?

Example : We would be understanding cMake with help of C Project

Step 1) In the root directory of your project, create a file named **CMakeLists.txt.(*Important*** : **Dont change this filename nor the case of letters,it is a convention used to identify the main build configuration file for a CMake project)** This file will specify how to build your project.

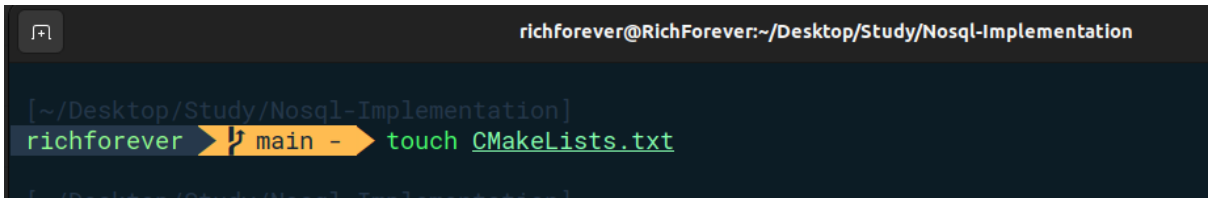
Basic Syntax :

```
cmake_minimum_required(VERSION 3.00)
project(project_name XX) # Put C,C++ or anything in place of
XX
set(STANDARD )     # Specify standard of language in place of
STANDARD
set(FLAGSANDLIBRARIES) # Specify flags and libraries of
corresponding language in place of FLAGSANDLIBRARIES
include_directories(HEADER_FILES)
file(GLOB MAIN
      sourcefile.extension# Specify files in the project with
extension , eg : abc.c , xyz.cpp
    )
    add_executable(executable ${MAIN})
```

Add the following code in "**CMakeLists.txt**" for this example

```
cmake_minimum_required(VERSION 3.00)
project(project_name C)
set(CMAKE_C_CSTANDARD 99)
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -pthread -lm")
```

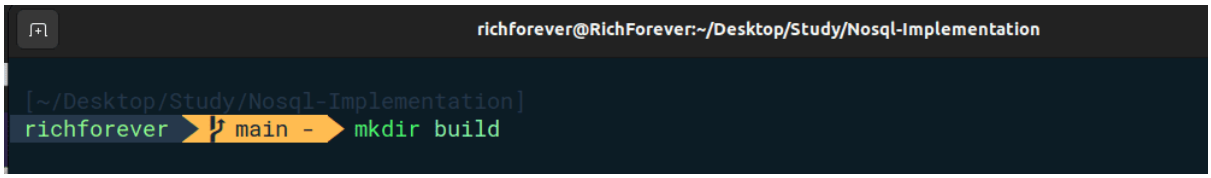
```
include_directories(HEADER_FILES)
file(GLOB MAIN
        "main.c"
        "globals/globals.c"  #All dependencies
        "Encryption/cipher.c"
        "Backend/document.c"
        "nosql.c"
        "DocumentHashmap/DocumentHashMap.c"
    )
    add_executable(executable ${MAIN})
```



Make a CMakeLists.txt file

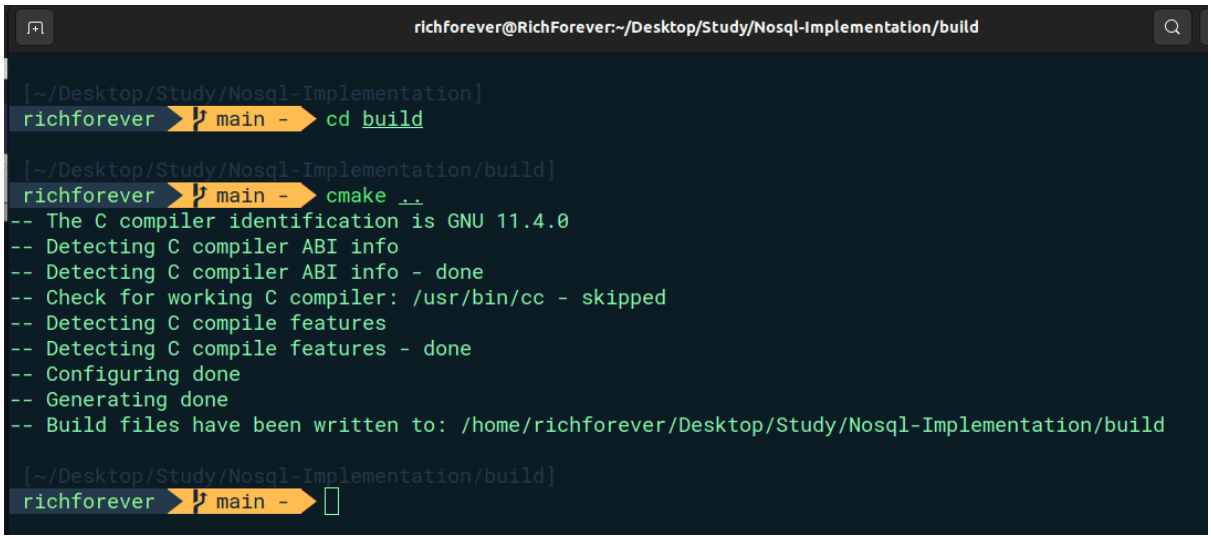
"**Touch**" command creates a new file.

Step 2) In your project's root directory, create a "**build**" directory, it's a good practice to create a separate build directory to keep your source directory clean.



"**Mkdir**" command creates a new directory.

Step 3) Go to "**build**" folder and type "**cmake ..**" there , Run **CMake** to configure the project, specifying the path to your project's root directory (where the **CMakeLists.txt** is located) which is just a folder behind .



Step 4) The **build directory** contents would now look as follows :

```
richforever@RichForever:~/Desktop/Study/Nosql-Implementation/build

[~/Desktop/Study/Nosql-Implementation/build]
richforever  main -  ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile

[~/Desktop/Study/Nosql-Implementation/build]
```

Now type '**make**' command in the terminal , the build tool to compile and build your project:

```
richforever@RichForever:~/Desktop/Study/Nosql-Implementation/build

[~/Desktop/Study/Nosql-Implementation/build]
richforever  main -  make
[ 14%] Building C object CMakeFiles/executable.dir/Backend/document.c.o
[ 28%] Building C object CMakeFiles/executable.dir/DocumentHashmap/DocumentHashMap.c.o
[ 42%] Building C object CMakeFiles/executable.dir/Encryption/cipher.c.o
[ 57%] Building C object CMakeFiles/executable.dir/globals/globals.c.o
[ 71%] Building C object CMakeFiles/executable.dir/main.c.o
[ 85%] Building C object CMakeFiles/executable.dir/nosql.c.o
[100%] Linking C executable executable
[100%] Built target executable

[~/Desktop/Study/Nosql-Implementation/build]
richforever  main -
```

After this the directory contents would look as follows :

```
richforever@RichForever:~/Desktop/Study/Nosql-Implementation/build

[~/Desktop/Study/Nosql-Implementation/build]
richforever  main -  ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  executable  Makefile
```

Step 5) Now execute the executable present in the directory that file would give you the desired output

```
./executable

[~/Desktop/Study/Nosql-Implementation/build]
richforever  main -  ./executable
                    Welcome to NoSQL implementation using C.
                    ----------------------------------------

  This project is developed under the guidance of Prof. Ashwini Matange ma'am by the students Pratyay Dhond,
  arvesh Kulkarni and Sohel Bargir of Computer Engineering Div 2.
  Available commands are ls,man,login,create. Use command man for more details

>>
```