

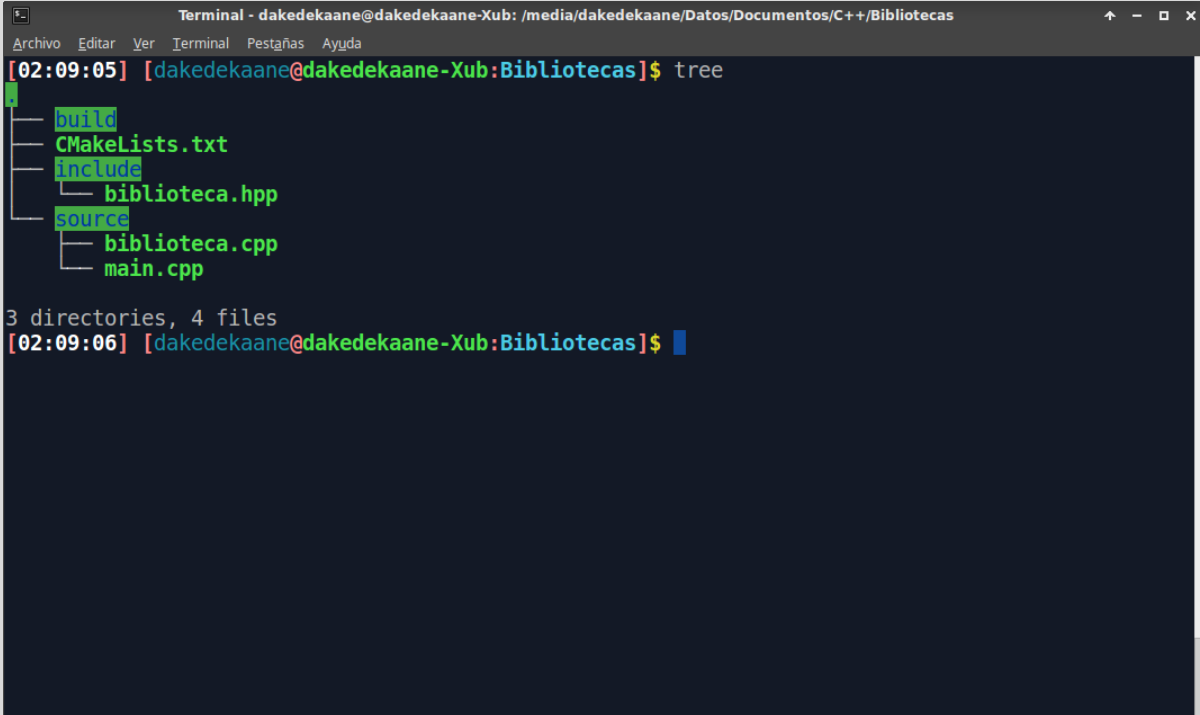
## Uso de bibliotecas y archivos de cabecera en Linux (Debian. Ubuntu)

Para un manejo fácil de múltiples archivos de código fuente en Linux, usaremos una herramienta llamada Cmake, que tendrá un comportamiento muy similar a lo que hace un IDE al momento de compilar.

Primero necesitaremos instalar Cmake, ésto se logra ingresando en la terminal el siguiente comando:

```
sudo apt-get install cmake
```


Una vez terminada la instalación, procederemos a cambiar al directorio en el que se encuentran nuestros códigos a compilar. Se recomienda tener una estructura similar a la mostrada en la siguiente imagen para facilitar la búsqueda y edición de archivos.



```
Terminal - dakedekaane@dakedekaane-Xub: /media/dakedekaane/Datos/Documentos/C++/Bibliotecas
Archivo Editar Ver Terminal Pestañas Ayuda
[02:09:05] [dakedekaane@dakedekaane-Xub:Bibliotecas]$ tree
.
├── build
├── CMakeLists.txt
├── include
│   └── biblioteca.hpp
└── source
    ├── biblioteca.cpp
    └── main.cpp

3 directories, 4 files
[02:09:06] [dakedekaane@dakedekaane-Xub:Bibliotecas]$
```

Se puede observar un archivo llamado “CMakeLists.txt. Dicho archivo servirá para configurar cómo se llevará a cabo la compilación de nuestro proyecto. El archivo debe tener la siguiente estructura:



The screenshot shows the Sublime Text editor with the file `CMakeLists.txt` open. The editor has a menu bar with `File`, `Edit`, `Selection`, `Find`, `View`, `Goto`, `Tools`, `Project`, `Preferences`, and `Help`. The tabs at the top show `main.cpp`, `biblioteca.hpp`, `biblioteca.cpp`, and `CMakeLists.txt`. The content of `CMakeLists.txt` is as follows:

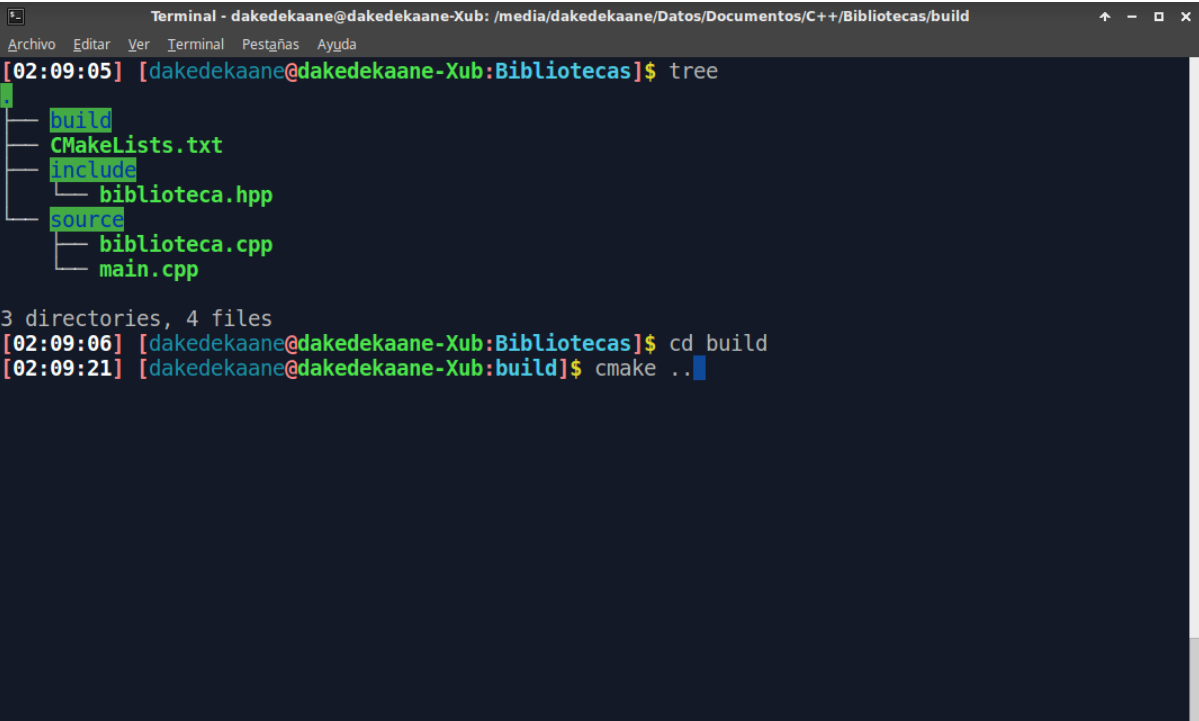
```
1 #cmake_minimum_required(version)
2 cmake_minimum_required(VERSION 2.8.9)
3
4 #project(nombreProyecto)
5 project(Bibliotecas)
6
7 #include_directories(directorio con los .hpp)
8 include_directories(include)
9
10 #set(SOURCES archivos .cpp)
11 set(SOURCES source/main.cpp source/biblioteca.cpp)
12
13 #add_executable(nombreEjecutable ${SOURCES})
14 add_executable(main ${SOURCES})
```

The status bar at the bottom indicates `Line 1, Column 33`, `Tab Size: 4`, and `Makefile`.

Una vez estando listo este archivo, regresamos a la terminal, nos dirigimos al subdirectorio build y escribimos el comando

```
cmake ..
```

para generar los archivos necesarios para el proyecto



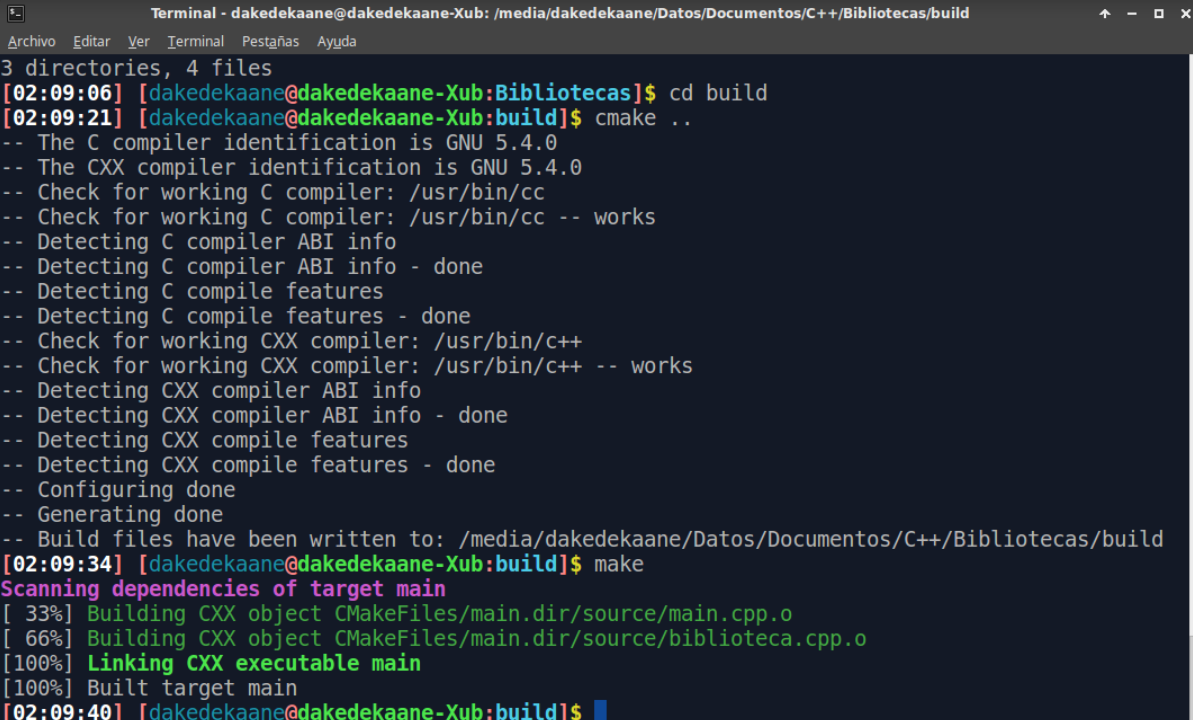
The screenshot shows a terminal window with the title `Terminal - dakedekaane@dakedekaane-Xub: /media/dakedekaane/Datos/Documentos/C++/Bibliotecas/build`. The terminal output is as follows:

```
[02:09:05] [dakedekaane@dakedekaane-Xub:Bibliotecas]$ tree
.
├── build
├── CMakeLists.txt
├── include
│   └── biblioteca.hpp
├── source
│   ├── biblioteca.cpp
│   └── main.cpp
3 directories, 4 files
[02:09:06] [dakedekaane@dakedekaane-Xub:Bibliotecas]$ cd build
[02:09:21] [dakedekaane@dakedekaane-Xub:build]$ cmake ..
```

Después, bastará con ingresar el comando

```
make
```

estando en el subdirectorio build para compilar nuestro programa, sin importar que tantos archivos hayamos modificado.



```
Terminal - dakedekaane@dakedekaane-Xub: /media/dakedekaane/Datos/Documentos/C++/Bibliotecas/build
3 directories, 4 files
[02:09:06] [dakedekaane@dakedekaane-Xub:Bibliotecas]$ cd build
[02:09:21] [dakedekaane@dakedekaane-Xub:build]$ cmake ..
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /media/dakedekaane/Datos/Documentos/C++/Bibliotecas/build
[02:09:34] [dakedekaane@dakedekaane-Xub:build]$ make
Scanning dependencies of target main
[ 33%] Building CXX object CMakeFiles/main.dir/source/main.cpp.o
[ 66%] Building CXX object CMakeFiles/main.dir/source/biblioteca.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
[02:09:40] [dakedekaane@dakedekaane-Xub:build]$
```

En caso de agregar algún archivo .cpp, no hay que olvidar actualizar el archivo CMakeLists.txt en la línea de set.