Tutorial Básico Cmake en Linux

Esta guía está pensada para gente que no tiene ningún conocimiento previo del proceso de compilación de C++ y Cmake en Linux. Existen varios manuales en internet y documentación oficial muy detallada, aunque puede ser un poco densa para alguien que está empezando. Es por eso que el objetivo de esta guía es entregarles de forma rápida y precisa la información necesaria para poder desarrollar las tareas de este curso sin problemas.

Antes de empezar

Esta guía está enfocada para linux. En caso de tener Ubuntu, sólo necesitan instalar Cmake. El compilador gcc lo trae por defecto. Los archivos .cpp y .txt pueden ser editados con cualquier procesador de texto (aunque para editar los archivos .cpp es bastante más cómodo usar un editor más especializado, como “Sublime text”).

¿ Qué es Cmake *?*

Cmake es un software multiplataforma para asistir el proceso para generar un proyecto (No es un compilador ,ni un entorno de desarrollo).

El siguiente ejemplo corresponde al programa muy simple, que solo muestra en pantalla la frase “Hola profesor Javier”. Esta separado en tres pasos.

Paso uno: Generación del proyecto (Solo es necesario una vez).

El mínimo de archivos necesario para compilar un código con Cmake son dos. Un archivo *.cpp (que trae el código), Fig. 1, y un archivo llamado CmakeLists.txt (que trae la configuración de para la compilación), Fig. 2.

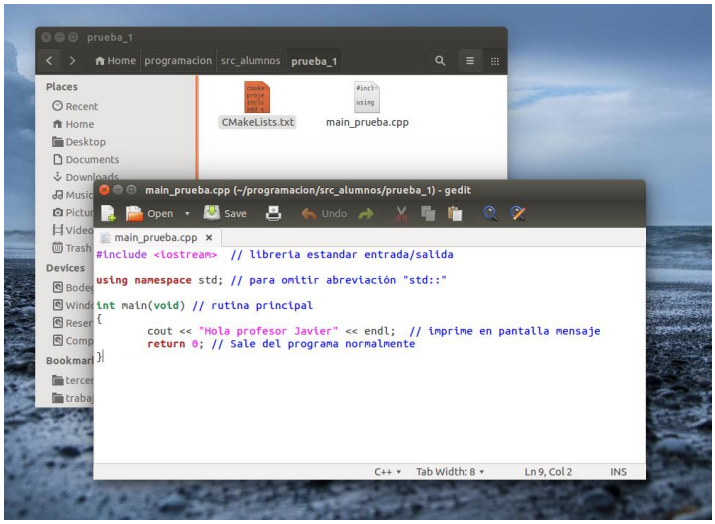

Figura 1: Código del programa en archivo main_prueba.cpp

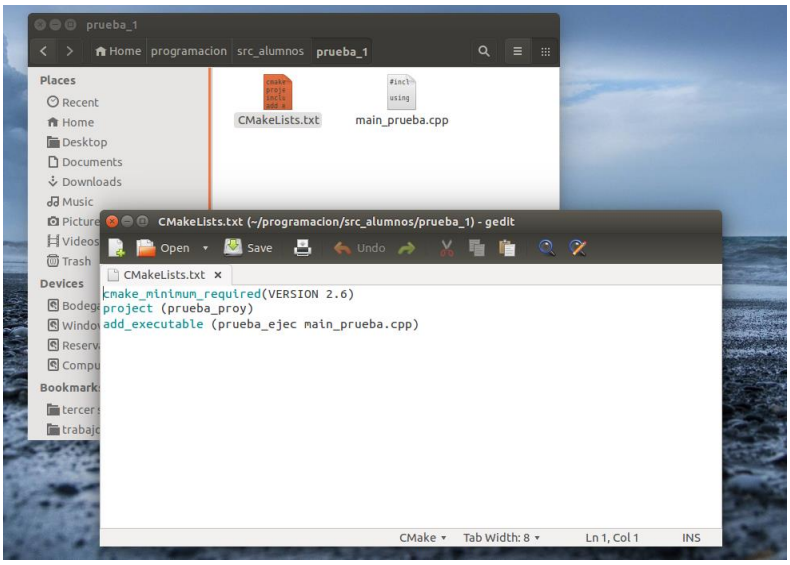

Figura 2: Configuración para Cmake en CmakeLists.txt.

En la fig. 2 se ve la configuración mínima del archivo Cmake. Se especifica dónde está el código y que cree un ejecutable bajo el nombre de “prueba_ejec”. También se especifica el nombre proyecto y la versión mínima necesario de Cmake.

Durante el curso la única librería necesaria es OpenCV. Se les entregará un archivo CmakeList.txt linkeado con la librería OpenCV en la primera tarea, el cual les servirá para todas las tareas. En caso de ocupar otras librerías o funciones, se les facilitara un archivo CmakeList.txt configurado.

El archivo que se ve en la fig.2 busca por defecto los archivos fuentes (*.cpp) y crea los ejecutables en la misma carpeta. Dado que en las tareas los códigos son cortos, lo anterior no es problema. Sin embargo, en proyectos más extensos es recomendado crear un directorio para compilar los ejecutables y otros para los archivos fuentes. Esto se puede hacer editando directamente el archivo CmakeList.txt o con alguna asistente automatizado o gui para la creación de archivos CmakeLists.txt .

Para crear la build , solo se debe escribir , estando en el directorio de los archivos, el siguiente comando desde la consola(Fig. 3):

```
cmake .
```

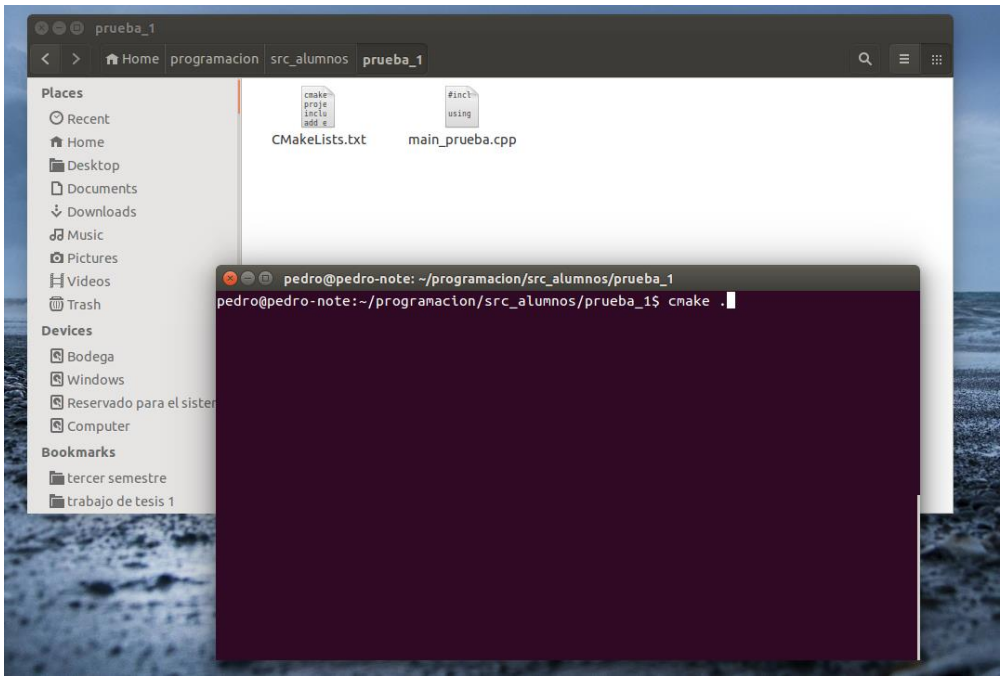

Figura 3: Generación del proyecto

Este paso muestra el siguiente mensaje y genera ciertos archivos (ver Fig. 4) necesarios para poder compilar en código. Este paso solo es necesario una vez en cada proyecto (a menos que se cambie de directorio). Si luego se modifica el código, solo es necesario compilar de nuevo.

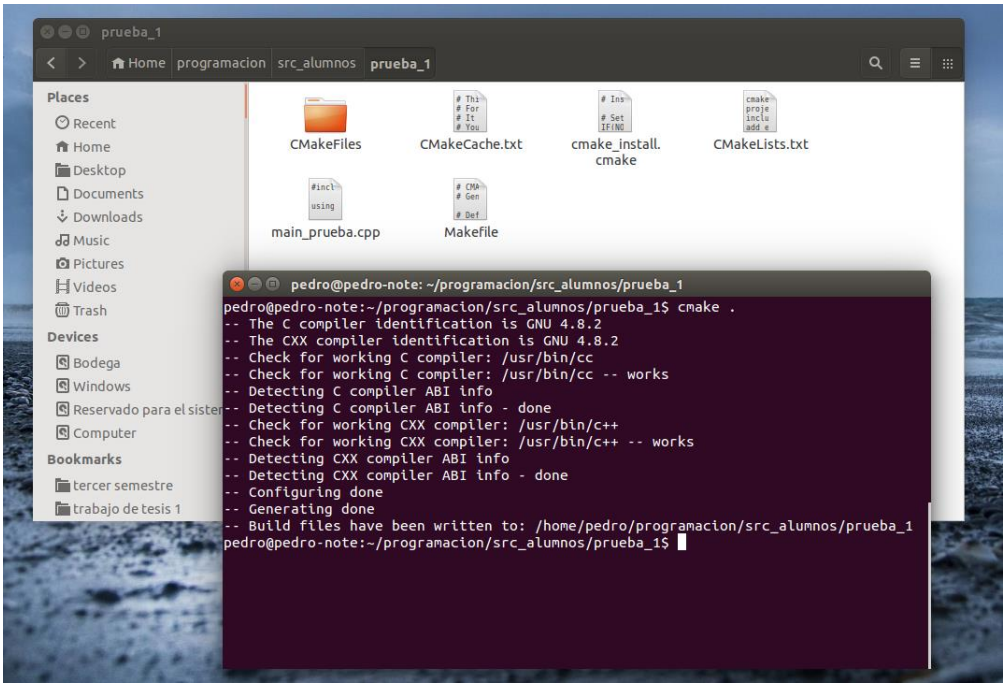

Figura 4: Generación del proyecto finalizada

Paso dos: Compilar

Para compilar lo único necesario es escribir , estando en el directorio, el siguiente comando:

```
make
```

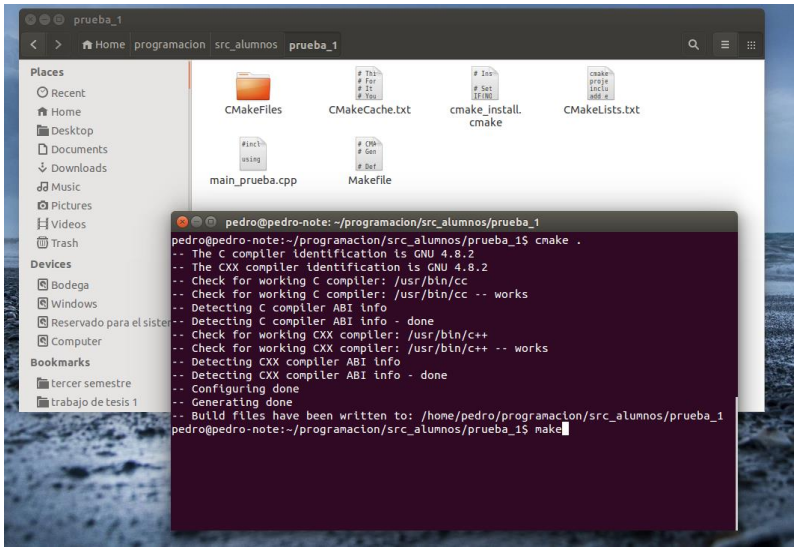

Figura 4: comando para compilar

Al finalizar de forma exitosa la compilación aparece el mensaje que se puede ver en la fig.5 , y se crea el fichero ejecutable . Si existe un error en el código y no se puede compilar, aparece un mensaje con el problema que existe y la línea de código donde ocurre.

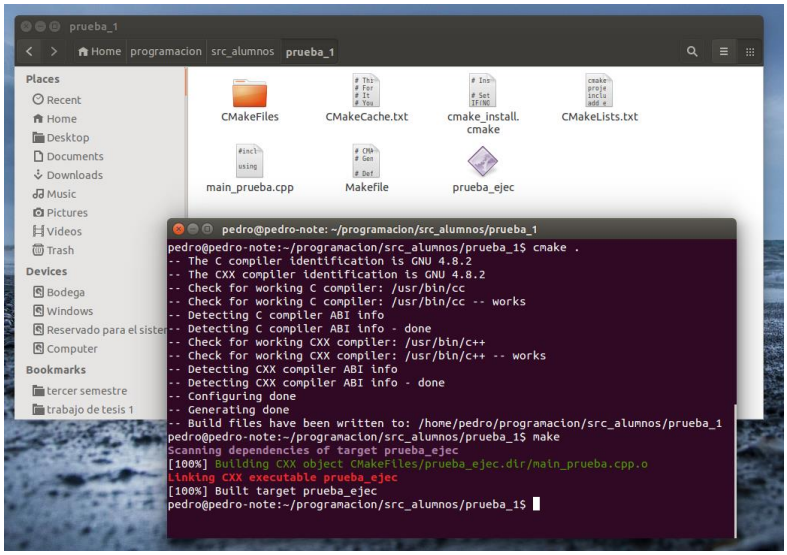

Figura 5: compilación terminada

Paso tres: Ejecutar

Finalmente para ejecutar el código, se debe escribir *"./nombre_archivo_ejecutable"*, en este caso el comando es el siguiente:

```
./prueba_ejec
```

El código se ejecuta y envía el mensaje esperado (fig.6)

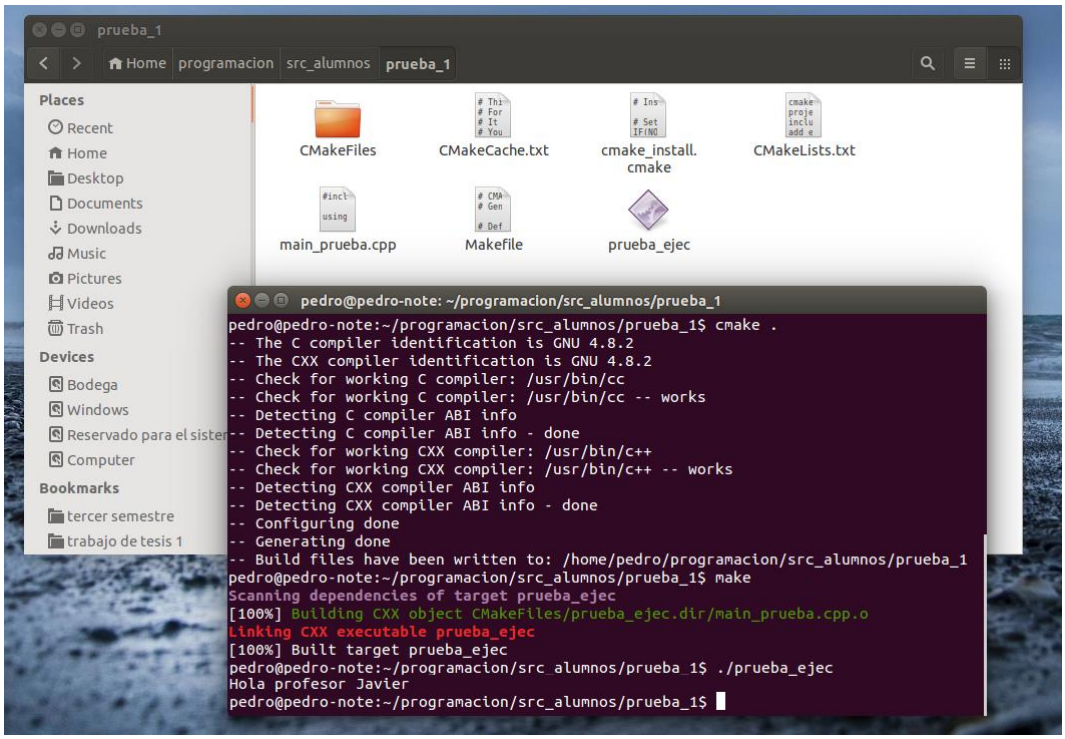

Figura 5: Ejecución del código

Finalmente, a modo de recomendación, para el nivel de complejidad de las tareas del curso un editor simple (recomendado "Sublime text") más compilar con Cmake a través de línea de comandos es más que suficiente.

Si quieren algún entorno de desarrollo integrado (IDE) más sofisticado, pueden usar Eclipse si usan Linux o Visual Studio si usan Windows. Para instalar y configurar estos editores con OpenCV existen muchos tutoriales muy sencillos paso a paso, incluso en formato de video.