Introducción a Cmake

¿Qué es CMake?

Cmake es una herramienta multiplataforma para facilitarnos la etapa de building. Automatiza la generación del Makefile (en linux) para que poder compilar nuestros programas. Su nombre proviene de Cross platform Make.

¿Por qué CMake?

- Makefile es exclusivo de entornos Unix
- Makefile no escala en proyectos grandes
- Su sintaxis es poco intuitiva.
- Busca unificar la compilación de un programa en C/C++ sin importar el sistema operativo

Ventajas de CMake

- Provee una abstracción a Makefile. Su sintaxis es más fácil de seguir
- Permite modularizar proyectos de forma sencilla
- Provee herramientas para detectar *third party libraries*

Hola mundo

```
cmake_minimum_required(VERSION 2.4)
project(cmake-demo)
add_executable(ej0 main.cpp)
```

```
all: build
build:
	g++ -o ej0 main.cpp
clean:
	rm *.o ej0
.PHONY: all clean build
```

- Cmake generará un makefile más complejo que el de la derecha

¿Cómo se usa?

```
$ ls
main.cpp CMakeLists.txt
$ mkdir build
$ cd build
$ cmake ..
$ make
```

Bibliotecas

```
# Raiz del proyecto
cmake_minimum_required(VERSION 2.4)
project(cmake-demo)

# Incluyo un cmake que define una biblioteca
add_subdirectory(lib)

add_executable(ej1 main.cpp)

# Linkeo la biblioteca con nuestro programa
target_link_libraries(ej1 mylib)
```

```
# lib
add_library(mylib greeter.cpp)
```

- **add_subdirectory** nos permite anidar archivos cmake para tener módulos chicos
- **add_library:** genera una biblioteca estática. Si quisieramos crear una biblioteca dinámica, hay que pasarle el *flag* **SHARED**
- **target_link_libraries:** Define el paso de link edición. Le avisa al linker las bibliotecas que queremos se linkeen a nuestro programa.

Flags de compilación

Cmake nos permite definir flags de compilación. Para eso tenemos que usar la variable **CMAKE_CXX_FLAGS** (**CMAKE_C_FLAGS** para programas en C).

```
...
# A lo que tenga la variable, le agregamos flags que querramos
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Werror")
...
```

Wildcards

```
project(cmake-demo)
FILE(GLOB myFiles "*.cpp")
message(STATUS "Sources loaded!: ${myFiles})

add_executable(ej0 ${myFiles})
```

- Se podría usar FILE(GLOB_RECURSE) para que se haga una búsqueda recursiva de los archivos (aunque en ese caso sería preferible cmakes recursivos.
- No es una práctica recomendada. Es preferible que se defina explícitamente los archivos para que el Makefile tenga más capacidad de detectar cambios.

Directivas de instalación

```
cmake_minimum_required(VERSION 2.4)
project(cmake-install)

add_executable(ej2 main.cpp)

install(TARGETS ej2 DESTINATION ~/installed/ej2)
```

- La función install agrega un target extra en el Makefile autogenerado, para instalar los targets definidos al destino. Muy útil para generar un instalador del tp final en una sola línea!.

Bibliografía

- https://cmake.org/documentation/
- https://cmake.org/cmake/help/latest/guide/tutorial/index.html