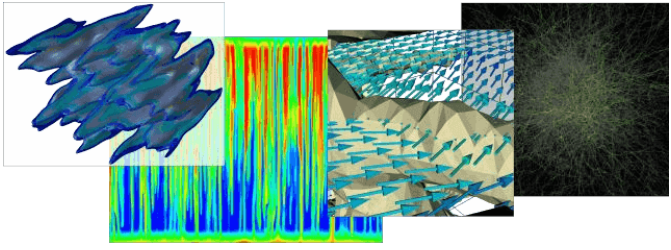
Introducción a la herramienta cmake

Parte I: `bash`, `g++`, `cmake`, `gnuplot`, `gmsh`, `paraview`



Practique los ejemplos en Gitpod

Open in Gitpod

Disponible en

¡Únete al grupo en Telegram!

CMake

Es un *generador de sistemas de compilación* de código abierto e independiente del compilador y/o plataforma, es decir, produce instrucciones para otros sistemas de compilación como Makefile, Ninja, Visual Studio, Qt Creator, Android Studio y Xcode. También incluye características que permiten la instalación, empaquetamiento y soporte nativo de pruebas de software.

Resultados del aprendizaje:

- Construir diversos ejemplos de proyectos de CMake en los lenguajes C/C++/Fortran/Python que construyan ejecutables u objetos compartidos/estáticos/interfaces.
- Correr pruebas con `ctest`, `catch2`, `gtest`, `pytest`.
- Usar dependencias de terceros en un proyecto CMake.

Está orientado a cualquier estudiante, profesor, investigador, programador que quiera aprender de Arch Linux y quiera aprender a usar efectivamente CMake en un proyecto basado en Dune.

No se se asume que tenga experiencia en GNU/Linux o C++, pero sí familiarizado con algún lenguaje de programación.

Tres herramientas de línea de comando

- `/usr/bin/cmake`
- `/usr/bin/cpack`
- `/usr/bin/ctest`

Tres herramientas interactivas

- `/usr/bin/ccmake`
- `/usr/bin/cmake-gui`

Instalación

Esta tecnología imprescindible en el lenguaje C++ se encuentra disponible en mayoría de repositorios de distribuciones GNU/Linux importantes. Si desea tener una versión actual, puede instalar desde el repositorio `[extra]`

```
[user@host somedir]$ sudo pacman -Syu cmake graphviz plantuml gcovr cppcheck python-cpplint cc
```

Un archivo `CMakeLists.txt` minimal

Supongamos que tenemos el siguiente script `CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.27)
project(Awesome VERSION 1.0.0 LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
add_executable(ejecutable main.cc)
```

Entendamos, este script en más detalle. En este script definimos los requisitos para la construcción, desde el código fuente y objetivos, pasando por las pruebas, empaquetamiento, etc. Y delegaremos la tarea de compilación al programa `make`.

- `cmake_minimum_required()` indica la versión mínima de `cmake` que requiere para ejecutar.
- `project()` define el nombre del proyecto, su número de versión y que está escrito en el lenguaje de programación C++.
- `set()` asigna la variable de entorno, en este caso establece la versión del estándar C++ 20.
- `add_executable()` crea un ejecutable a partir de un script en C++.

El proceso de construcción CMake

Tipos de construcción CMake

- Debug
- Release
- RelWithDebInfo
- MinSizeRel
- None

Generación automática de documentación con CMake

La documentación es una parte esencial de cualquier proyecto exitoso.
Veamos cómo integrar Doxygen con CMake para generar automáticamante la documentación para proyectos con CMake.
CMake generará un archivo `Doxyfile`.
Esperamos que Doxygen genere la documentación de la interfaz de programación de aplicaciones (API) para cada clase y sus diagramas de herencia con `dot` de graphviz.

Pruebas con ctest

El comando `duneproject`

Es un script asistente escrito en el lenguaje `bash` que se encuentra en `duneproject` dentro del paquete `dune-common`.

Referencias

- Libros
  -  Dominik Berner y Mustafa Kemal Gilor. *CMake Best Practices*. First. Packt, 2022. ISBN: 978-1-803-23972-9.
  -  Craig Scott. *Professional CMake: A Practical Guide*. Fifteen. 2023.
- Artículos
  -  M. Clemencic y P. Mato. "A CMake-based build and configuration framework". En: *Journal of Physics: Conference Series* (1 de ene. de 2021). DOI: 10.1088/1742-6596/396/5/052021. URL: https://dx.doi.org/10.1088/1742-6596/396/5/052021.
- Sitios web
  -  ArchWiki. *CMake package guidelines*. 16 de dic. de 2022. URL: https://wiki.archlinux.org/title/CMake_package_guidelines (visitado 16-12-2022).