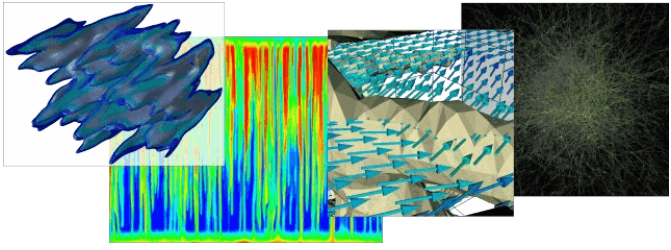
# Introducción a la herramienta cmake

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



Practique los ejemplos en Gitpod

Open in Gitpod

Disponible en

¡Únete al grupo en Telegram!

# CMake (2000 - actualidad)

Es un *generador de sistemas de compilación* de código abierto multiplataforma e independiente del compilador, es decir, genera instrucciones para otros sistemas de compilación como GNU Make, Ninja y archivos de proyecto para IDEs populares como Microsoft Visual Studio, Eclipse, Qt Creator, CLion, Code::Blocks, Sublime Text, Kate, Android Studio y Xcode.
Se mantiene centrado en soportar compiladores modernos y cadena de herramientas. Tiene soporte nativo de pruebas de software, empaquetamiento e instalación como una parte inherente del proceso de construcción.

**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` y `pytest`.
- Usar dependencias de terceros en un proyecto CMake.

**Prerrequisitos**

- Está orientado a cualquier estudiante que quiera aprender a usar efectivamente CMake en un proyecto basado en Dune sobre Arch Linux.
- No se asume que esté familiarizado con GNU/Linux o C++, pero sí que esté familiarizado con algún lenguaje de programación.
- Los ejemplos y ejercicios han sido probados en Arch Linux.

# Instalación de software

Esta tecnología imprescindible se encuentra disponible en la mayoría de repositorios de distribuciones GNU/Linux importantes. Si desea instalar una versión actual, lo puede encontrar en el repositorio `[extra]`

```
[user@host somedir]$ sudo pacman -Syu # Actualizar
[user@host somedir]$ sudo pacman -S cmake clang doxygen graphviz
```

# Un archivo `CMakeLists.txt` minimal

```
cmake_minimum_required(VERSION 3.27)
project(Awesome VERSION 1.0.0 LANGUAGES CXX)
message("Hola CMake!")
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.
- `message()`
  .
- `add_subdirectory()`
  .
- `add_executable()`
  crea un ejecutable a partir de un script en C++.

# Primeros pasos con CMake

Esperemamos automatizar la construcción escribiendo un script que vaya a través de nuestro árbol del proyecto que compile todo.
Este escript debe evitar compilaciones innecesarias y detecte cuando el script ha sido modificado desde la última vez que corrimos.
También debe conocer cómo enlazar todos los archivos compilados en un binario.
CMake no puede construir por sí mismo, este delega la tarea a otras herramientas en el sistema para desarrollar la compilación actual, enlace y otras tareas.
Puede pensar como el orquestador del proceso de construcción (sabe qué pasos necesita realizarse, cuál es el objetivo final y cómo encontrar a los trabajadores correctos y materiales para el trabajo)

**árbol de construcción (build tree)** es la ruta al directorio objetivo / salida.

**árbol de fuente (source tree)** es la ruta en el que se encuentra su código fuente.

# Las tres etapas en CMake

**Etapa de configuración**

- Lee los detalles del proyecto guardados en el árbol de construcción y preparará un árbol de construcción para la etapa de generación.
- Crea un árbol de construcción vacío y colecciona los detalles del entorno en el que se trabaja (arquitectura, compiladores disponibles, enlazadores, revisa un problema de prueba simple que puede compilar correctamente. archivos)
- El archivo de configuración del proyecto es analizado y ejecutado (en lenguaje CMake). Le dice acerca de la estructura del proyecto, objetivos, dependencias (bibliotecas y paquetes CMake).
- Guarda información recolectada en el árbol de construcción como detalles del sistema, configuración del proyecto, registros, archivos temporales; que serán usados en la siguiente etapa.
- El archivo `CMakeCache.txt` es creador para guardar variables estables (rutas del compilador y otras herramientas) y ahoora tiempo durante la siguiente configuración.

**Etapa de generación**

- Generará un sistema de construcción para el entorno exacto en el que trabaja (`Makefile` para GNU Make, `Ninja` o IDE).
- Puede aplicar toques finales de la configuración evaluando expresiones generadoras.

**Etapa de construcción**

- Corriendo las herramientas de construcción apropiadas obtendremos los artefactos.
- Estas heramientas ejecutarán pasos para producir objetivos con compiladores, enlazadores, herramientas de análisis estático, frameworks de prueba, herramientas de reporte, y cualquier otra cosa que pueda pensar.

# Aprendiendo la línea de comando

### Herramientas de línea de comando

- `/usr/bin/cmake`
  ejecutable principal que configura, genera y construye proyectos.
- `/usr/bin/ctest`
  programa de manejador de pruebas usado para correr y reportar resultados de pruebas.
- `/usr/bin/cpack`
  este es el programa de empaquetamiento usado para generar el instalador y paquete de origen.

### Herramientas interactivas

- `/usr/bin/cmake-gui`
  envoltorio gráfico alrededor de `cmake`.
- `/usr/bin/ccmake`
  envoltorio en modo texto (curses) alrededor de `cmake`.

# /usr/bin/cmake

Este binario provee algunos modos de operaciones

**Generación de un sistema de construcción del proyecto**
Este es el primer paso requerido para construir nuestro proyecto.

**Construcción de un proyecto**
.

# Tipos de construcción CMake

- Debug
- Release
- RelWithDebInfo
- MinSizeRel
- None

# Generación automática de documentación con CMake

La documentación es esencial en 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 `/usr/bin/duneproject` dentro del paquete `dune-common`.

# Referencias


- Libros
  -  Frank T. Willmore, Eric Jankowski y Coray Colina. *Introduction to Scientific and Technical Computing*. First. CRC Press, 2016. ISBN: 978-1-315-38239-5.
  -  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).