# Preliminares de Geant4

* Es un toolkit de simulación de Monte Carlo.
* Utilizado para HEP (High Energy Physics), ciencias espaciales, física de radiación, física medica y otros.
* Geant4 es un toolkit que realiza simulaciones basándose en el concepto de simulación Monte Carlo.
* Geant4 emplea código de Monte Carlo para el transporte de radiación.

## Simulación Monte Carlo

1. La simulación Monte Carlo consta de definir un conjunto de entradas y restricciones para un sistema.
2. Generar entradas de forma aleatoria en el sistema y realizar los cálculos.
3. Agregar los resultados de los cálculos individuales al resultado final.

### Puntos de entrada y condicionamiento

* Definición del sistema a modelar.
* Definición de los códigos de transporte de radiación.
  + Arreglo / configuración a simular.
    - Dentro de la configuración definición de geometrías, materiales, modelos en 3D, detectores.
* Definición de campo de radiación.
  + Fuente, energía, dirección, partículas.
* Definición de procesos físicos.
* Resultado de cálculos (dosis depositadas, sección transversal, dirección de energía y momento).

### Generación aleatoria de entradas

* Creación de entradas aleatorias al sistema simulado.
* Definición de partículas primarias a simular.
* Para la creación aleatoria de la partícula primaria con condiciones iniciales se simula ene el medio inicial, determinando las posibles interacciones con el medio.
* Para elegir un proceso físico, se utiliza un generador de números aleatorios(pseudoaleatorio).
* Se establecen registros, ejemplo: perdida de energía debido a los procesos físico de interacción.
* Se mantiene un seguimiento de las partículas secundarios creadas por procesos físicos.
* Se mantiene un rastreo de las partículas primarias y las particular secundarias, terciarias … hasta una energía (energía de corte).
* De manera que el proceso se repite para cada partícula primaria, hasta llegar al valor establecido de partículas primarias a simular.

### Anexar resultados

* Adjuntar los resultados de los cálculos individuales.
* Realizar el análisis de los datos de la simulación de Monte Carlo.
* Datos obtenidos, pueden ser el calculo de cantidades dosimétricas, dosis absorbidas, dosis depositada por partículas primarias, incluyendo formas de dosis como, dosis puntuales, individuales, mapa de dosis de 1, 2 y 3 dimensiones, espectro de energía, etc.

### Esquema de simulación

Esquema general de un detector de simulación

Un programa de simulación de detectores debe brindar la posibilidad de describir con precisión una configuración experimental (tanto en términos de materiales como de geometría)

● El programa debe brindar la posibilidad de generar eventos físicos (cinemática) y rastrear partículas de manera eficiente a través del detector simulado

● Las interacciones entre partículas y materia deben simularse teniendo en cuenta todos los procesos físicos posibles, para todo el rango de energía.

● Se debe proporcionar la posibilidad de registrar en tiempo de ejecución todas las cantidades necesarias para reproducir el comportamiento del experimento.

● Debe haber algún soporte gráfico.

## ¿Por qué necesitamos simulación?

* Modelar procesos físicos
* Diseñar un conjunto de "cortes" de análisis que encuentren de manera óptima lo que estamos buscando.
* Simular un nuevo modelo de teórico para ver si podemos implementarlo.

## Introducción Geant4

* Geant4 es un toolkit para desarrollo de una aplicación para el transporte de radiación de partículas.
* El cual es utilizado como una forma de estimar los efectos de la radiación, para una región dada.

Dentro del transporte de radiación las partes importantes son:

1. Una fuente de radiación.
2. El modelo 3D, de la geometría, es decir la configuración de un detector.
3. Un volumen de donde medir la radiación.

Dentro del mismo esquema, la forma mas simple de estimación es definiendo:

* Deposición de energía en un volumen y su varianza.
* Dosis en el volumen.
* Flujos, partículas (e-, neutrones ...) las interacciones en una particular región.
* Es posible estimar observables complicados, es decir distribuciones de deposición de energía, dosis, etc.

## Diseño de Geant4

* Geant4 esta diseñado para fácil mantenimiento y desempeño.
* Es un toolkit, Open source.
* Tiene un diseño modular.
* Diseñado de forma que los componentes son relativamente independientes de cada uno.
* Diseño implementado en 17 categorías.
* El marco esta diseñado para permitir al usuario implementar su propio diseño de aplicaciones. Con el fin de que el usuario pueda definir y diseñar las bibliotecas a utilizar de forma granular, y así reducir los tiempos de compilación.

### Paradigmas de Programación

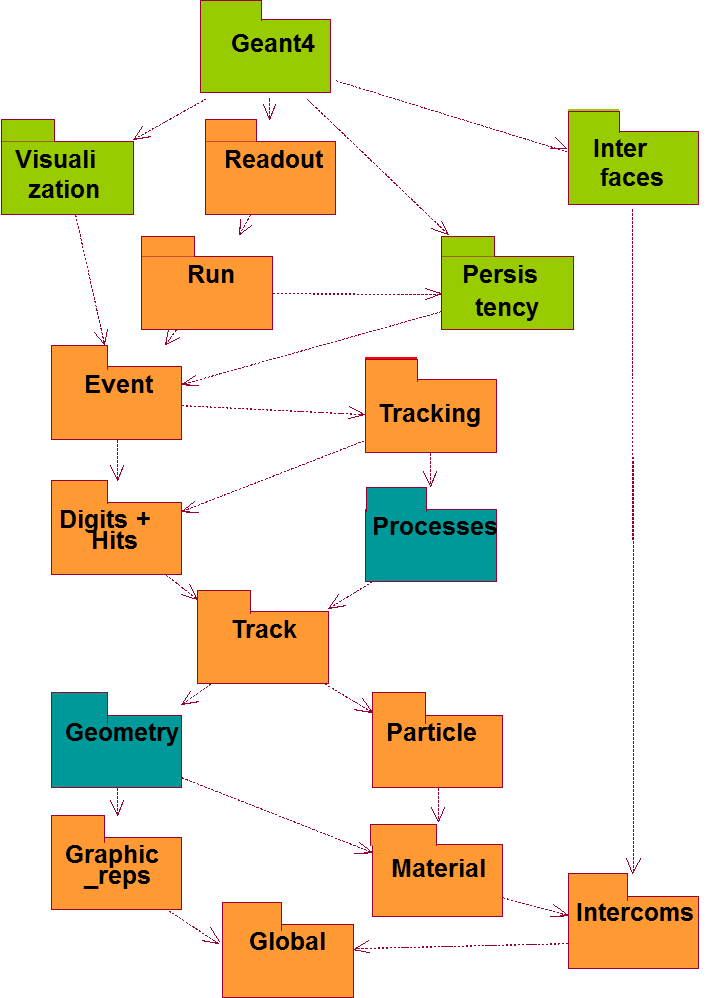
* Geant4 utiliza el paradigma de programación orientada a objetos (OOP).
* Específicamente el lenguaje C++, dado a su gran desempeño, código fuente libre, gran uso de la comunidad y gran soporte.

### Kernel de Geant4

* El kernel, es la parte central ya que maneja las partes esenciales de una aplicación de Geant4. El cual permite la física y la grabación de los eventos ocurridos, incluyendo:
* Geometría y materiales.
* Tracking.
* Eventos (colisiones o interacciones primarias).
* Runs (las simulaciones hechas).
* En la figura se muestra el diagrama del kernel de GEant4. REF: Agostinelli, S., Allison, J., Amako, K. A., Apostolakis, J., Araujo, H., Arce, P., ... & Geant4 Collaboration. (2003). GEANT4—a simulation toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *506*(3), 250-303.

El kernel de Geant4 esta constituido de forma modular, con una estructura jerárquica, donde cada categoría maneja una parte específica del toolkit. La figura siguiente muestra el diagrama unidireccional de cada dependencia, empezando de abajo hacia arriba.

La diferentes categorías son usadas por categorías son usadas por categorías con jerarquías más altas, por ejemplo, la categoría de “Geometry” utiliza la menor categoría de “Material”, que establece primero el tipo de material antes de definir la geometría.



Los módulos de mayor importancia incluyen:

1. Global. - cubre el sistema de unidades, constantes, números y la aleatoriedad.
2. Material .- definiciones de los materiales.
3. Particles .- el tipo de partícula, así como las características a utilizar (energía, momento, etc.).
4. Geomtry .- la geometría, volúmenes, detectores, rastreo de partículas sobre un objeto.

Seguido de los módulos que describen el seguimiento de las partículas y los procesos físico a los que se somete la simulación.

### Nomenclatura de Geant4

* Los archivos de programación editables de Geant4 tienen la extensión .cpp o .cc.
* Las bibliotecas (header files) tienen extensión .hh.
* Clases:
  + Geant4 tiene definidas sus propias clases, a pesar de que existen clases ya prescritas en C++.
  + Al menos las clases relacionadas con Geant4, tienen el prefijo G4; ejemplo, G4int, G4double, G4RunManager, G4Box, etc.

### Convenciones utilizadas

● Todas las clases de Geant4 tienen su nombre con un prefijo, G4.

- G4RunManager, G4Step, G4LogicalVolume.

● Las clases abstractas agregan una V al prefijo.

- G4VHit, G4VPhysicalVolume

● Cada palabra de un nombre compuesto se escribe en mayúscula.

- G4UserAction, G4VPV Parametrización

● Los métodos y las funciones siguen las mismas convenciones de nomenclatura que los nombres de las clases.

- G4RunManager :: SetUserAction (), G4LogicalVolume :: GetName ().

En el caso de los tipos de clases dentro de la programación de Geant4. Geant4 redefine los conceptos básicos tipos para que tengan siempre el mismo rendimiento de bits.

* Para tipos numéricos básicos, diferentes compiladores en diferentes plataformas proporcionan diferentes rangos de valores.
* ﻿int 🡪 G4int
* long 🡪 G4long
* float 🡪 G4float
* double 🡪 G4double
* bool 🡪 G4bool
* string 🡪 G4String

● Las definiciones de estos tipos se colocan en una archivo de encabezado único (globals.hh), que también proporciona inclusión de todos los encabezados del sistema, así como global funciones que necesita el kernel Geant4.

### Componentes requeridos

* Una ‘Physics List’, para decirle a Geant4 que procesos debe simular.
* Una construcción de un detector, la geometría a simular.
* Un generador de partículas, seleccionar y establecer que partículas debe simular Geant4.

Los componentes anteriores son la base de mínimos componentes que debe tener una aplicación de Geant4.

### El tracking en Geant4

* El track es un instante de una partícula.
  + Un track, en la simulación es borrado si:
    - Deja el volumen mas eterno (el volumen del mundo definido).
    - Desaparece en una interacción (decaimiento o choque inelástico). Es posible, además, establecer una energía de corte, a la cual a partir de interacciones de menor energía ya no se realiza un seguimiento de la partícula.
    - Si su energía se convierte en 0 y no tiene proceso en reposo.
    - Si el usuario decide detener la simulación o el proceso.
* Sus cantidades físicas representan el instante actual en la simulación.
* Antes del track, del inicio de una simulación no hay registro de cantidades previas.
* Un step es el delta, el cambio de un track. El track no es una colección de pares, sino que un track se actualiza en una serie de pasos.
* Para grabar los tracks, se debe utilizar objetos de una clase de trayectoria. Esta clase se llama G4TrackingManager, la cual gestiona el procesamiento de un track, cada track es representado por un objeto de la clase G4Track.
* Un usuario puede crear sus propias opciones como G4UserTrackingManager.

### Esquema General

Típicamente en una aplicación de Geant4, durante una simulación tendremos los siguiente:

* Unas partículas iniciales.
* Cada track (trayectoria), tendrá su estado actual, donde cada track tendrá su punto inicial y su punto final. Marcados por los steps que denotan que tan lejos se llega a la partícula.
* Los procesos físicos suceden en tres puntos: 1) en reposo cuando las partículas no tienen energía cinética. 2) Durante un step, en particular la deposición de energía es proporcional al tamaño del step. 3) Después del step, suceden interacciones discretas fuera del step.
* Dentro de este esquema existe también un cut, que es un umbral de corte, dado que un procesos físico puede generar un número infinito de procesos secundarios y para evitar esto es posible introducir una energía de corte.

## Configuración de un experimento y simulación de física de altas energías

Diagram

Description automatically generated

### Simulación de procesos físicos

Los procesos físicos describen cómo las partículas interactúan con el material.

Geant4 proporciona siete categorías principales:

* electromagnético
* hadrónico
* decaer
* fotoleptón-hadrón
* óptico
* parametrización
* transporte

A picture containing bar chart

Description automatically generated

Procesos de física electromagnética

* Procesos de fotones: dispersión de Compton, conversión gamma, efecto fotoeléctrico, e + e y producción de pares de muones
* Partícula cargada Procesada (Electrón / posítron, muones, iones…):
* ionización y emisión de rayos delta
* Bremsstrahlung
* aniquilación de positrones
* Dispersión múltiple

Interacciones hadrónicas

* interacciones leptón-hadrón
* reacciones fotonucleares y electronucleares
* reacciones núcleo-núcleo
* dispersión elástica
* cascadas nucleares
* modelos de fisión, evaporación y ruptura
* interacciones de neutrones de baja energía
* desintegración radioactiva

Fotones ópticos:

* Radiación de Cerenkov
* Centelleo
* cambio de longitud de onda
* Absorción
* Dispersión de Rayleigh y Mie.

# Instalación de Geant4

## Especificaciones de Ordenador

Sistema operativo: macOS Big sur ver. 11.4.

Macbook Air (M1, 2020).

Chip Apple sillicon M1.

Memoria Ram 8 gb.

La siguiente imagen muestra las especificaciones del sistema.

Graphical user interface, website

Description automatically generated

Antes de comenzar esta guía se realizo basándose en la información de como instalar Geant4 en la página de la documentación.

* https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/index.html

## Prerrequisitos

Para conocer todos los prerrequisitos consulte el link [link de la pagina referenciada de arriba] página de la documentación de la instalación.

Los prerrequisitos de a continuación son los más básico para poder utilizar Geant4.

Para esto se requiere primero el código fuente el cual se descarga del link, el cual es el archivo geant4.xxx.tar.gz. Donde xxx denota para la versión que este disponible. Para este caso se va a utilizar la versión más reciente, la 10.07.p01.

 Geant4 Toolkit [Source Code](http://cern.ch/geant4/support/download.shtml). (Archivo con extensión xxx.tar.gz).

 C++ Compiler and Standard Library supporting the C++11 Standard:

* macOS: Apple Clang ([Xcode](https://developer.apple.com/xcode/)) 11 or higher.

En el caso de macOS, es necesario la versión más reciente de Xcode. Para esto descargamos Xcode de la Apple store, el cual contiene las librerías para el compilador de C++. Además de esto es necesario tener el Xcode comando line tolos para poder ejecutar comandos con Xcode desde la terminal, abriendo una terminal y ejecutando el código xcode-select –install.

Una vez instalado podemos checar la versión con /usr/bin/xcodebuild -version.

La versión usada en esta guía es 12.5. Como se ve en la figura de abajo.

A picture containing text, watch, screenshot

Description automatically generated

\*Nota: en caso de presentar problemas con clang(c language) instalar clang-format, con brew install clang-formag.

* [CMake](https://cmake.org) 3.8 o más reciente.
  + Para instalar cmake, nos dirigimos a la página web y descargamos el archivo .dmg y procedemos a instalar la aplicación.
  + En este caso es la versión 3.20.3
  + Para verificar la versión en terminal, cmake --version.
  + Si se instala desde el archivo .dmg, solo se tiene acceso a la interfaz gráfica de usuario (GUI), para usar cmake desde la terminal ejecutamos la aplicación y vamos a la barra de tareas en Tool -> How to install for command line use, y se siguen las instrucciones que aparecen de agregar Cmake a PATH , que es la variable de entorno Unix que le dice al shell en donde buscar los ejecutables según el comando escrito y ejecutado.

## Prerrequisitos Opcionales

Dependiendo de los sistemas que se planear usar, es necesario tener algunas librerías instaladas para el correcto funcionamiento de Geant4. En esta guía solo se van a instalar los prerrequisitos los cuales son los que se van a utilizar.

En referencia a macOS, la guía recomienda instalar las librerías necesarias a través de los archivos binarios y de los instaladores de la página oficial. Sin embargo, para este caso ya que estos métodos no fueron exitosos procedemos a utilizar la aplicación [Homebrew](https://brew.sh/) o [MacPorts](https://www.macports.org/), el cual permite instalar dependencias y paquetería desde la terminal, pero en ocasiones puede fallar ya que la instalación por este medio no ha sido testeada por los desarrolladores de Geant4.

En este caso vamos a utilizar el administrado de paquetes Homebrew.

* En la página de Hombrew se encuentran las instrucciones de como instalarlo.
* [Instalación](https://docs.brew.sh/Installation).
* En el caso de chips Apple, el folder raíz es /opt/homebrew.
* Terminado de instalar, pide que agregar el PATH de homebrew, lo cual sigue las instrucciones.

Una vez instalado, los comandos son desde la terminal

* brew install <pkg\_name>.
* brew uninstall <pkg\_name>. \*\* Para remover copias antiguas de archivos descargados usar brew cleanup.
* brew list (muestra los paquetes instalados).
* brew list <pkg\_name> (muestra si el paquete xx está instalado.
* brew list <pkg\_name> (muestra si el paquete xx está instalado.
* brew install –-cask <app\_name> En el caso de una aplicación es, con homebrew cask. \*Nota: por ejemplo es posible instalar la aplicación firefox desde homebrew, brew install –-cask firefox

Nota: homebrew instala por default las dependencias de otros paquetes por lo que puede instalar algunos paquetes sin necesariamente ejecutar el comando del paquete que se quiere tal cual.

## Librerías CLHEP, Libraries

* En este caso no se va a utilizar esta librería ya que no se ha podido instalarla sin error.

## Soporte de geometría GDML XML Parser

Para habilitar el uso de lectura / escritura de geometría desde archivos GDML XML, los encabezados y biblioteca Xerces-C ++> = 3 deben estar instalados.

* brew install xerces-c. Apache xerces-c++ XML parser, la versión se encuentra dentro de ../xerces-c/3.2.3/lib/libxerces-c-3.2.dylib.
* Versión 3.2.3.

## Controladores de interfaz de usuario y visualización

Para manejar la visualización de Geant4 es necesario instalar las dependencias de paquetería. Geant4 utiliza la visualización Qt.

Interfaz de usuario y visualización de Qt5 (todas las plataformas Unix,Linux,Windows)

Encabezados y bibliotecas de Qt5

* Geant4 utiliza la visualización de Qt5.
* Para esto es posible, registrarse para una cuenta gratis y descargar el código fuente, e instalar lo en una carpeta principal, pero en este caso se va a descargar Qt5 por medio de homebrew.
* brew install qt@5
* Los paquetes individuales que utiliza Geant4 son los módulos de Qt5 individuales, el
* Qt5Core.
* Qt5Gui.
* Qt5Widgets.
* Qt5OpenGL.
* Qt5PrintSupport.
* El uso del controlador Qt3D opcional y experimental requiere además Qt5 5.15 y los módulos Qt53DCore, Qt53DExtras y Qt53DRender.
* La versión instalada de homebrew es 5.15.2.
* En caso de presentar error también es posible instalar qt, con brew install qt.

### Bibliotecas y encabezados OpenGL o MesaGL.

* Es sistema operativo ya tiene por default istalado OpenGL.

### Visualización X11 OpenGL (Linux y macOS)

* Para esto es necesario tener instalado X11, que es XQuartz para Unix(macOs).
* Para esto ir a la página de [XQuartz](https://www.xquartz.org/), descargar el .dmg e instalar la aplicación.
* Versión instalada 2.8.1, como se ve en la figura de abajo

A picture containing text

Description automatically generated

* Para esto es también XQuartz, X11 para las bibliotecas y encabezados X11 (XQuartz en macOS).
* Así como también para la visualización de X11 RayTracer (Linux y macOS).

## Python Bindings

\*\* Nota: esta parte de Python no se instalo porque presento errores a la hora de realiza el make install ya que para la construcción de los archivos no presento error, sino hasta la instalación marcando un error de no encontrar una la carpeta G4Python más o menos ese nombre.

The Python bindings to Geant4, Geant4Py, may now be built as part of the main Geant4 build and require:

* [Python](https://www.python.org) 3.X intérprete, librería y encabezados. Python3 es posible instalarlo desde la página oficial o con homebrew, con brew install python.
* [Boost Python](https://www.boost.org) 1.69 biblioteca y encabezados, compilados / vinculados con la versión requerida de Python.
* En el caso de Python instalar la versión 3.
* En el caso de Boost Python con, brew install boost-python3.

Una vez finalizado la instalación de los prerrequisitos procedemos a instalar Geant4.

# Instalación Geant4

Para esto se abre la terminal y se escoge en donde se van a construir los archivos necesarios y donde se van a instalar.

\*\*Nota: En caso de requerir más información a fondo y conocer la instalación completa ir a [link](https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/installguide.html).

\*\*Nota: La Shell utilizada en la terminal es zsh.

En este caso se va a utilizar como directorio de trabajo el directorio -$HOME. O Para acceder ejecutar cd ~.

Pasos que seguir.

1. Para crear los archivos del código fuente Geant4 es necesario descomprimir el archivo geantxxx.tar.gz en el directorio de trabajo.
2. Para esto ya que estamos en el directorio $HOME, para saber en el directorio en donde nos encontramos usamos pwd.

A continuación, se muestra la sesión de la terminal.

macosx@z37 ~ % cd $HOME

macosx@z37 ~ % pwd

/Users/macosx

macosx@z37 ~ %

Creamos una carpeta que contendrá el código fuente, la carpeta de construcción y la de instalación. En esta carpeta principal se descomprime el código fuente de Geant4.

macosx@z37 ~ % mkdir Geant4\_Root\_Folder

macosx@z37 ~ % cd Geant4\_Root\_Folder

macosx@z37 Geant4\_Root\_Folder % mkdir geant4.10.07-build geant4.10.07-install

macosx@z37 Geant4\_Root\_Folder % ls -l

total 0

drwxr-xr-x 2 macosx staff 64 Jun 1 22:58 geant4.10.07-build

drwxr-xr-x 2 macosx staff 64 Jun 1 22:58 geant4.10.07-install

drwxr-xr-x 2 macosx staff 64 Jun 1 22:58 geant4.10.07.p01

macosx@z37 Geant4\_Root\_Folder % tree

.

├── geant4.10.07-build

├── geant4.10.07-install

└── geant4.10.07.p01

3 directories, 0 files

El comando tree muestra la estructura del directorio de trabajo. Donde la carpeta \*\*\*-build va a contener los archivos binarios de geant4 y \*\*\*-install los archivos instalados.

Para cargar los archivos binarios se realizan con cmake, en la carpeta \*\*\*-build. Pero antes de cargar los archivos binarios debemos crear dentro de la carpeta build una carpeta con el nombre de data, la cual tiene los folders descomprimidos de las librerías de geant4 que utiliza; los cuales se descargan de la página de geant4, donde se encuentra el código fuente, como se ve en la figura de abajo.

Text

Description automatically generated

Estos archivos se descomprimen y se copian a la carpeta geant4xxx-build 🡪 data.

Una vez finalizado procedemos a crear los archivos binarios. Las opciones son las siguientes

* -DCMAKE\_INSTALL\_PREFIX=/Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install, es en donde se va a instalar el toolkit de geant4.
* -DGEANT4\_INSTALL\_DATADIR=/Users/macosx/Geant4\_Root\_Folder/geant4.10.07build/data, es la dirección del directorio del folder data de los archivos descomprimidos. Es también posible descargar la data con la opción -DGEANT4\_INSTALL\_DATA=ON pero requiere de una conexión estable de internet ya que si presenta error marca error. Es por eso que es recomendable descargar la data y dar la dirección de directorio en donde se encuentra.
* -DGEANT4\_USE\_QT=ON -DGEANT4\_USE\_GDML=ON -DGEANT4\_USE\_OPENGL\_X11=ON DGEANT4\_USE\_RAYTRACER\_X11=ON, marcan las opciones de Qt, GDML, X11 y Raytracer\_X11 que se van a utilizar e instalar.
* ../geant4.10.07.p01 marca la dirección de donde se encuentra la carpeta del código fuente.

macosx@z37 geant4.10.07-build %

cmake -DCMAKE\_INSTALL\_PREFIX=/Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install

-DGEANT4\_INSTALL\_DATADIR=/Users/macosx/Geant4\_Root\_Folder/geant4.10.07-build/data

-DGEANT4\_USE\_QT=ON -DGEANT4\_USE\_GDML=ON -DGEANT4\_USE\_OPENGL\_X11=ON DGEANT4\_USE\_RAYTRACER\_X11=ON ../geant4.10.07.p01

-- The C compiler identification is AppleClang 12.0.5.12050022

-- The CXX compiler identification is AppleClang 12.0.5.12050022

-- Detecting C compiler ABI info

-- Detecting C compiler ABI info - done

-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc - skipped

-- Detecting C compile features

-- Detecting C compile features - done

-- Detecting CXX compiler ABI info

-- Detecting CXX compiler ABI info - done

-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ - skipped

-- Detecting CXX compile features

-- Detecting CXX compile features - done

-- Found EXPAT: /Library/Developer/CommandLineTools/SDKs/MacOSX11.3.sdk/usr/lib/libexpat.tbd (found version "2.2.8")

-- Found XercesC: /opt/homebrew/lib/libxerces-c.dylib (found version "3.2.3")

-- Found OpenGL: /Library/Developer/CommandLineTools/SDKs/MacOSX11.3.sdk/System/Library/Frameworks/OpenGL.framework

-- Found XQuartzGL: /usr/X11R6/include

-- Found X11: /opt/homebrew/include

-- Looking for XOpenDisplay in /opt/homebrew/lib/libX11.dylib;/opt/homebrew/lib/libXext.dylib

-- Looking for XOpenDisplay in /opt/homebrew/lib/libX11.dylib;/opt/homebrew/lib/libXext.dylib - found

-- Looking for gethostbyname

-- Looking for gethostbyname - found

-- Looking for connect

-- Looking for connect - found

-- Looking for remove

-- Looking for remove - found

-- Looking for shmat

-- Looking for shmat - found

-- Looking for IceConnectionNumber in ICE

-- Looking for IceConnectionNumber in ICE - not found

-- Reusing dataset G4NDL (4.6)

-- Reusing dataset G4EMLOW (7.13)

-- Reusing dataset PhotonEvaporation (5.7)

-- Reusing dataset RadioactiveDecay (5.6)

-- Reusing dataset G4PARTICLEXS (3.1.1)

-- Reusing dataset G4PII (1.3)

-- Reusing dataset RealSurface (2.2)

-- Reusing dataset G4SAIDDATA (2.0)

-- Reusing dataset G4ABLA (3.1)

-- Reusing dataset G4INCL (1.0)

-- Reusing dataset G4ENSDFSTATE (2.3)

-- Looking for sys/types.h

-- Looking for sys/types.h - found

-- Looking for stdint.h

-- Looking for stdint.h - found

-- Looking for stddef.h

-- Looking for stddef.h - found

-- Check size of off64\_t

-- Check size of off64\_t - failed

-- Looking for fseeko

-- Looking for fseeko - found

-- Looking for unistd.h

-- Looking for unistd.h - found

-- Looking for pthread.h

-- Looking for pthread.h - found

-- Performing Test CMAKE\_HAVE\_LIBC\_PTHREAD

-- Performing Test CMAKE\_HAVE\_LIBC\_PTHREAD - Success

-- Found Threads: TRUE

-- The following Geant4 features are enabled:

GEANT4\_BUILD\_CXXSTD: Compiling against C++ Standard '11'

GEANT4\_USE\_SYSTEM\_EXPAT: Using system EXPAT library

GEANT4\_USE\_GDML: Building Geant4 with GDML support

GEANT4\_USE\_QT: Build Geant4 with Qt support

GEANT4\_USE\_OPENGL\_X11: Build Geant4 OpenGL driver with X11 support

GEANT4\_USE\_QT3D: Build Geant4 Qt3D driver

-- Configuring done

-- Generating done

-- Build files have been written to: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-build

macosx@z37 geant4.10.07-build %

instalar Una vez finalizado debe decir “-- Build files have been written to:” marcando el folder -build. Lo siguiente es marcar la construcción de los archivos con make -jn donde n quiere decir el número de proceso paralelos a llevar.

macosx@z37 geant4.10.07-build % make -j8

[ 0%] Building C object source/CMakeFiles/G4zlib.dir/externals/zlib/src/adler32.c.o

[ 0%] Building C object source/CMakeFiles/G4zlib.dir/externals/zlib/src/crc32.c.o

[ 0%] Building CXX object source/externals/ptl/CMakeFiles/ptl-shared.dir/src/TaskRunManager.cc.o

[ 0%] Building C object source/CMakeFiles/G4zlib.dir/externals/zlib/src/compress.c.o

[ 0%] Building CXX object source/externals/ptl/CMakeFiles/ptl-shared.dir/src/TaskAllocator.cc.o

[ 0%] Building CXX object source/externals/ptl/CMakeFiles/ptl-shared.dir/src/TaskAllocatorPool.cc.o

.

.

.

[100%] Building CXX object source/CMakeFiles/G4physicslists.dir/physics\_lists/util/src/G4WarnPLStatus.cc.o

[100%] **Linking CXX shared library ../BuildProducts/lib/libG4physicslists.dylib**

[100%] Built target G4physicslists

Y así empezando la instalación, dado que es muy largo no se muestra todo completo. Y así finalizando la instalación con make install

macosx@z37 geant4.10.07-build % make install

**Consolidate compiler generated dependencies of target ptl-shared**

[ 0%] Built target ptl-shared

**Consolidate compiler generated dependencies of target G4clhep**

[ 2%] Built target G4clhep

**Consolidate compiler generated dependencies of target G4global**

[ 4%] Built target G4global

**Consolidate compiler generated dependencies of target G4intercoms**

[ 5%] Built target G4intercoms

**Consolidate compiler generated dependencies of target G4graphics\_reps**

[ 5%] Built target G4graphics\_reps

**Consolidate compiler generated dependencies of target G4zlib**

[ 5%] Built target G4zlib

.

.

.

**Consolidate compiler generated dependencies of target G4error\_propagation**

[100%] Built target G4error\_propagation

**Consolidate compiler generated dependencies of target G4VRML**

[100%] Built target G4VRML

Install the project...

-- Install configuration: "Release"

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/include/Geant4

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/include/Geant4/tools

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/include/Geant4/tools/rroot

.

.

.

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/share/Geant4-10.7.1/examples/.doxygen/header.html

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/share/Geant4-10.7.1/examples/.doxygen/.README.novice.txt

-- Installing: /Users/macosx/Geant4\_Root\_Folder/geant4.10.07-install/share/Geant4-10.7.1/examples/.README.HowToRun

macosx@z37 geant4.10.07-build %

Dado que el proceso es largo se muestra un resumen de como es. Ahora si verificamos el directorio de -install debería de estar instalado Geant4, dentro de share, con el nombre de Geant4-10.7.1, el cual también contiene los ejemplos da las diferentes aplicaciones y la ejecutable fuente geantxxx.sh.

macosx@z37 geant4.10.07-build % cd ..

macosx@z37 Geant4\_Root\_Folder % cd geant4.10.07-install

macosx@z37 geant4.10.07-install % ls -l

total 0

drwxr-xr-x 5 macosx staff 160 Jun 1 23:48 bin

drwxr-xr-x 3 macosx staff 96 Jun 1 23:47 include

drwxr-xr-x 40 macosx staff 1280 Jun 1 23:47 lib

drwxr-xr-x 3 macosx staff 96 Jun 1 23:47 share

macosx@z37 geant4.10.07-install % cd share

macosx@z37 share % ls

Geant4-10.7.1

macosx@z37 share % ls -l

total 0

drwxr-xr-x 5 macosx staff 160 Jun 1 23:48 Geant4-10.7.1

macosx@z37 share % cd Geant4-10.7.1

macosx@z37 Geant4-10.7.1 % tree -L 2

.

├── examples

│   ├── CMakeLists.txt

│   ├── GNUmakefile

│   ├── History

│   ├── README

│   ├── README.HowToRun

│   ├── README.HowToRunMT

│   ├── advanced

│   ├── basic

│   ├── extended

│   └── novice

├── geant4make

│   ├── config

│   ├── geant4make.csh

│   └── geant4make.sh

└── tools.license

Listo Geant4 ha sido instalado. Para verificar esto corre el ejemplo B1 básico.

## Instalación de ROOT

La instalación de ROOT de Cern, se llevo a cabo en el mismo sistema con la siguientes especificaciones.

## Especificaciones de Ordenador

Sistema operativo: macOS Big sur ver. 11.4.

Macbook Air (M1, 2020).

Chip Apple sillicon M1.

Memoria Ram 8 gb.

1. Para instalar ROOT se instalo desde homebrew con brew install root y se instala la versión root6.
2. Una vez finalizado homebrew debe de agregar el thisroot.sh al home path, y poder ejecutarlo desde la terminal escribiendo root o root -l para suprimir el texto de bienvenida.

Text

Description automatically generated

1. Para acceder al PyROOT, solo debemos agregar . ~/local/root/pro/bin/thisroot.sh al archive de ~/.bash\_profile/ .
2. De modo que para esto usamo TextEdit open -a TextEdit ~/.bash\_profile
3. Finalmente, ya es posible importar módulos de ROOT a python3.

Instalacion de ROOT ambiente Linux

/---//---/

/---//---/

En la siguiente lista se muestran los programas requisitos antes de poder instalar ROOT de CEARN. En un ambiente con las siguientes especificaciones.

# Raspberry pi4

* Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
* 8GB LPDDR4-3200 SDRAM
* 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
* Gigabit Ethernet
* 2 USB 3.0 ports; 2 USB 2.0 ports.
* Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
* 2 × micro-HDMI ports (up to 4kp60 supported)
* 2-lane MIPI DSI display port
* 2-lane MIPI CSI camera port
* 4-pole stereo audio and composite video port
* H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
* OpenGL ES 3.0 graphics
* Micro-SD card slot for loading operating system and data storage
* 5V DC via USB-C connector (minimum 3A\*)
* 5V DC via GPIO header (minimum 3A\*)
* Power over Ethernet (PoE) enabled (requires separate PoE HAT)
* Sistema Operativo: Ubuntu-MATE 20.04 LTS
* sd card 64gb Samsung endurable

## Prerrequisitos

* sudo apt-get install git dpkg-dev cmake g++ gcc binutils libx11-dev libxpm-dev libxft-dev libxext-dev
* sudo apt-get install gfortran libssl-dev libpcre3-dev xlibmesa-glu-dev libglew1.5-dev libftgl-dev libmysqlclient-dev libfftw3-dev libcfitsio-dev graphviz-dev libavahi-compat-libdnssd-dev libldap2-dev python-dev libxml2-dev libkrb5-dev libgsl0-dev libqt4-dev

## Opcionales

* sudo apt-get install gfortran libssl-dev libpcre3-dev \
* xlibmesa-glu-dev libglew1.5-dev libftgl-dev \
* libmysqlclient-dev libfftw3-dev libcfitsio-dev \
* graphviz-dev libavahi-compat-libdnssd-dev \
* libldap2-dev python-dev libxml2-dev libkrb5-dev \
* libgsl0-dev libqt4-dev

# Instalación

Se descarga la versión ultima de ROOT el cual es un archivo .tar.gz , para este caso se va a instalar la versión “root\_v6.24.00.source.tar.gz”.

El siguiente paso es crear un directorio en una localización cualquiera en este caso es el Dekstop. La carpeta a crear se llama CERN\_Root

* mkdir CERN\_Root

el archivo descargado .tar.gz se copia a este nuevo directorio creado,.

* cp <file dir>Downloads/root\_v6.20.04.source.tar.gz <new dir>CERN\_Root/
* cd CERN\_Root

descomprimimos el archiv .tar.gz

* tar -zxvf root\_v6.20.04.source.tar.gz

Se descomprime y debe quedar una carpeta root-6.20.04, ahora creamos un dir para el build y uno para install.

* mkdir root-build
* mkdir root-install

accedemos al folder de root-build y cargamos con el cmake los archivos necesarios.

* cmake ../root-6.24.00/

ahora cargamos todas la librerias marcando de que directorio debe extraerlas, el cual es del archivo .tar.gz descomprimido , con la opcion de -Dall=ON

* sudo cmake ~/Desktop/CERN\_Root/root-6.24.00/ -Dall=ON

Luego ejecutmaos el make con

* make -jN (donde N es el numero de núcleos que deben usarse para la instalación)
* sudo make install
* root