Build a Geant4 application

Davide Chiappara
University of Padova (UNIPD)
Istituto Nazionale di Fisica Nucleare (INFN)

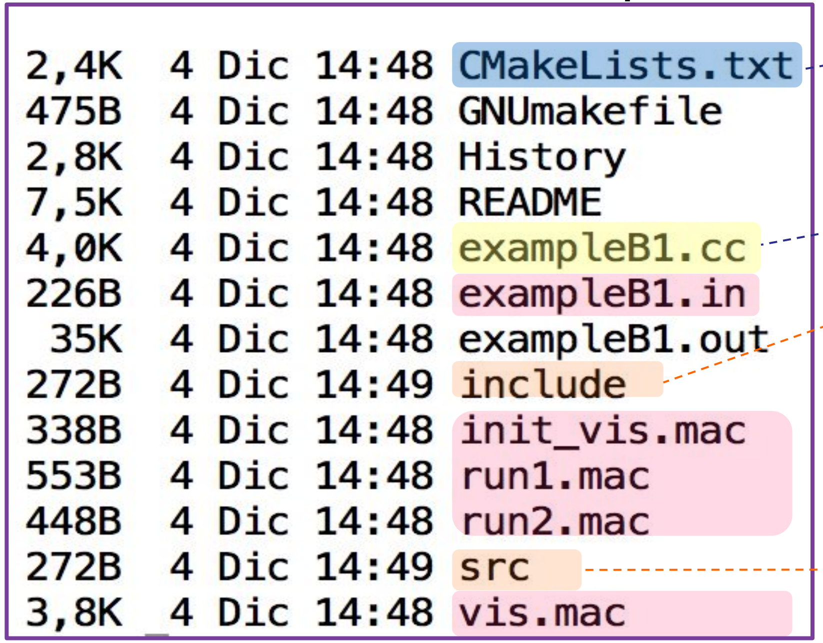
XX Seminar on Software for Nuclear, Subnuclear and Applied Physics

Application build process

1) Properly organize your code into directories
2) Prepare a CMakeLists.txt file
3) Create a build directory and run CMake
4) Compile (make) the application
5) Run the application

Application source structure in Geant4

Official basic/B1 example:



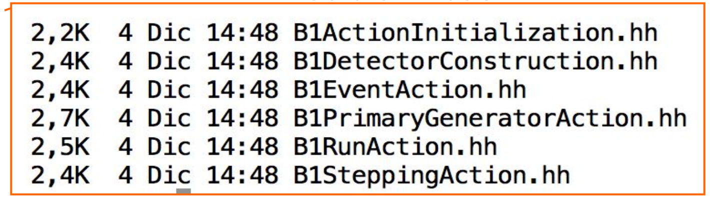
```
2,4K  4 Dic 14:48 CMakeLists.txt
475B  4 Dic 14:48 GNUmakefile
2,8K  4 Dic 14:48 History
7,5K  4 Dic 14:48 README
4,0K  4 Dic 14:48 exampleB1.cc
226B  4 Dic 14:48 exampleB1.in
 35K  4 Dic 14:48 exampleB1.out
272B  4 Dic 14:49 include
338B  4 Dic 14:48 init_vis.mac
553B  4 Dic 14:48 run1.mac
448B  4 Dic 14:48 run2.mac
272B  4 Dic 14:49 src
3,8K  4 Dic 14:48 vis.mac
```

Macro file containing the commands

The text file CMakeLists.txt is the CMake script containing commands which describe how to build the exampleB1 application

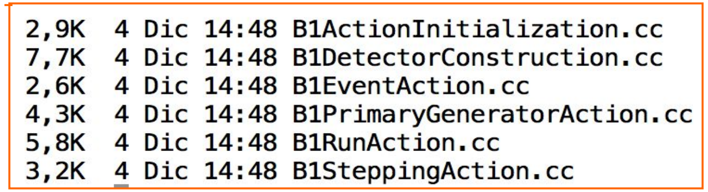
contains main() for the application

Header files



```
2,2K  4 Dic 14:48 B1ActionInitialization.hh
2,4K  4 Dic 14:48 B1DetectorConstruction.hh
2,4K  4 Dic 14:48 B1EventAction.hh
2,7K  4 Dic 14:48 B1PrimaryGeneratorAction.hh
2,5K  4 Dic 14:48 B1RunAction.hh
2,4K  4 Dic 14:48 B1SteppingAction.hh
```

Source files



```
2,9K  4 Dic 14:48 B1ActionInitialization.cc
7,7K  4 Dic 14:48 B1DetectorConstruction.cc
2,6K  4 Dic 14:48 B1EventAction.cc
4,3K  4 Dic 14:48 B1PrimaryGeneratorAction.cc
5,8K  4 Dic 14:48 B1RunAction.cc
3,2K  4 Dic 14:48 B1SteppingAction.cc
```

CMake (again)

- CMake is a build configuration tool
    - it takes configuration file (**CMakeLists.txt**)
    - it finds all dependencies (in our case, **Geant4**)
    - creates **Makefile** to run the compilation itself

- You have to write this **CMakeLists.txt** file

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)

project(B1)
option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()

include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

add_executable(exampleB1 exampleB1.cc ${sources} ${headers})
target_link_libraries(exampleB1 ${Geant4_LIBRARIES})

set(EXAMPLEB1_SCRIPTS
  exampleB1.in
  exampleB1.out
  init_vis.mac
  run1.mac
  run2.mac
  vis.mac
  )

foreach(_script ${EXAMPLEB1_SCRIPTS})
  configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
    ${PROJECT_BINARY_DIR}/${_script}
    COPYONLY
    )
```

File structure

1) Cmake minimum version and **project name**
2) Find and configure G4
3) Configure the project to use G4 and B1 headers
4) List the **sources**
5) Define and link the **executable**
6) Copy any macro files to the build directory

Build directory and CMake

1) If modifying the Geant4 examples, copy them to your **$HOME** first:

```
$ cp -r /usr/local/geant4/geant4.11.0.1/examples/basic/B1 ~
```

2) Create a **build directory***, where the compiled application will be put:

```
$ mkdir -p ~/B1-build
$ cd ~/B1-build
```

***Note:** It is possible (though not recommended) to compile **inside** source directory.

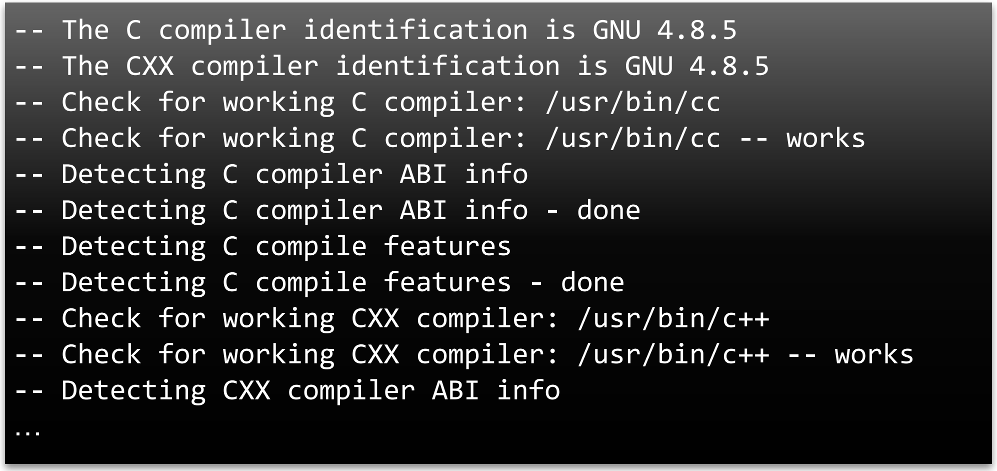
Run CMake

In the build directory you just created, run CMake:

Path to Geant4

```
$ cmake
-DGeant4_DIR=/usr/local/geant4/geant4.v11.1.1-install/lib/Geant4-1
1.1-1/ ~/B1/
```

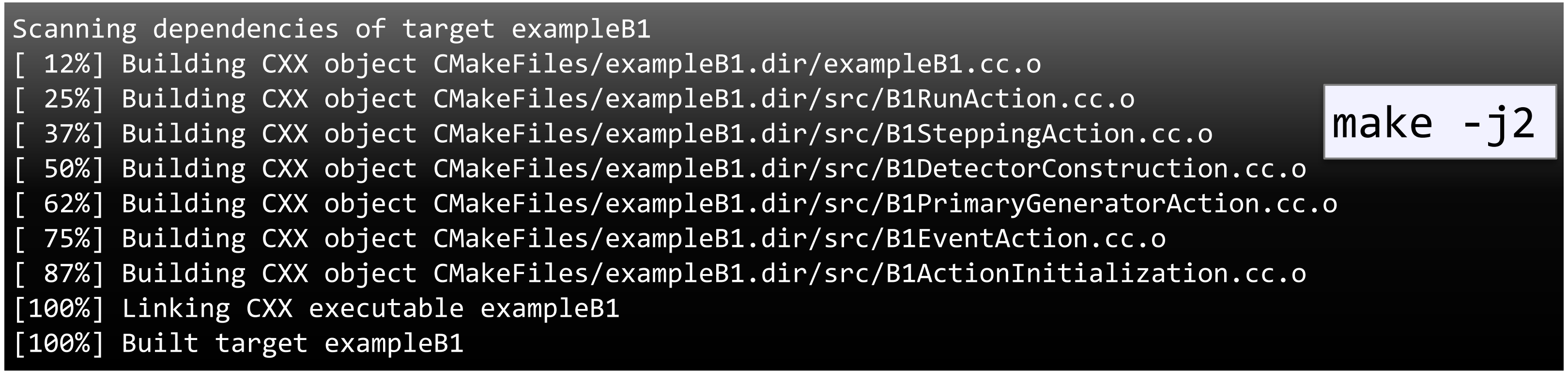
Path to source



```
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
…
```

Compilation

- In the build directory, run **make**
  - You have only a couple of files, it should be ready in a minute or two
  - An **executable** with the name of your application is created (e.g. **exampleB1**) in build directory
  - **Macros** and other auxiliary files are copied into build directory



```
Scanning dependencies of target exampleB1
[ 12%] Building CXX object CMakeFiles/exampleB1.dir/exampleB1.cc.o
[ 25%] Building CXX object CMakeFiles/exampleB1.dir/src/B1RunAction.cc.o
[ 37%] Building CXX object CMakeFiles/exampleB1.dir/src/B1SteppingAction.cc.o
[ 50%] Building CXX object CMakeFiles/exampleB1.dir/src/B1DetectorConstruction.cc.o
[ 62%] Building CXX object CMakeFiles/exampleB1.dir/src/B1PrimaryGeneratorAction.cc.o
[ 75%] Building CXX object CMakeFiles/exampleB1.dir/src/B1EventAction.cc.o
[ 87%] Building CXX object CMakeFiles/exampleB1.dir/src/B1ActionInitialization.cc.o
[100%] Linking CXX executable exampleB1
[100%] Built target exampleB1
```

```
make -j2
```

Do never modify files in the build directory

Remember that the **build directory** must be thought as a **temporary** directory.
Every time you run your **`make`**, you may **erase** or **overwrite** the files stored in build.

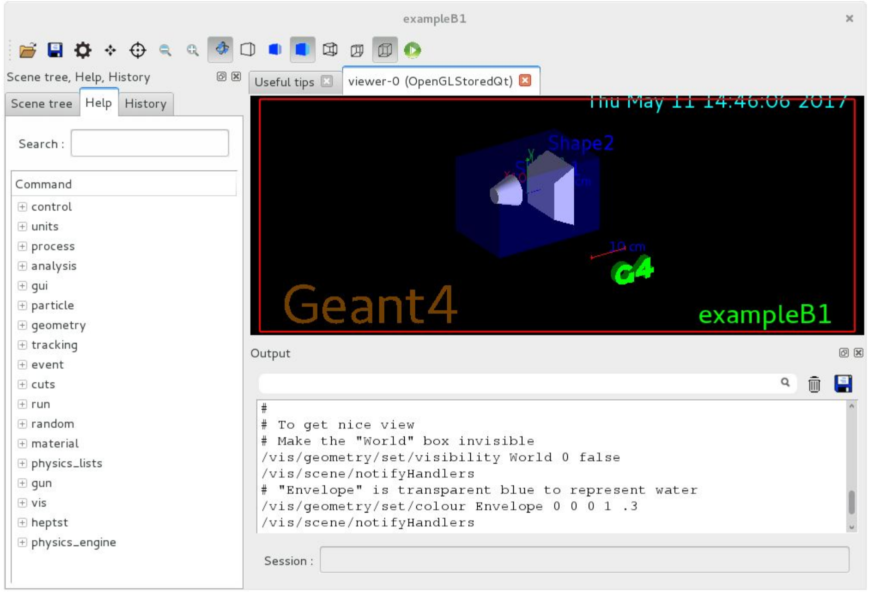
↓

If you want to **change anything** at all (also macros) change the file in the **source directory**, with a **`make`**, everything will be updated accordingly in the build directory (if everything is configured correctly).

Run the application - GUI

Just type the name of your application, including the **./** identifier of current directory (e.g. **./exampleB1**)

```
$ ./exampleB1
```



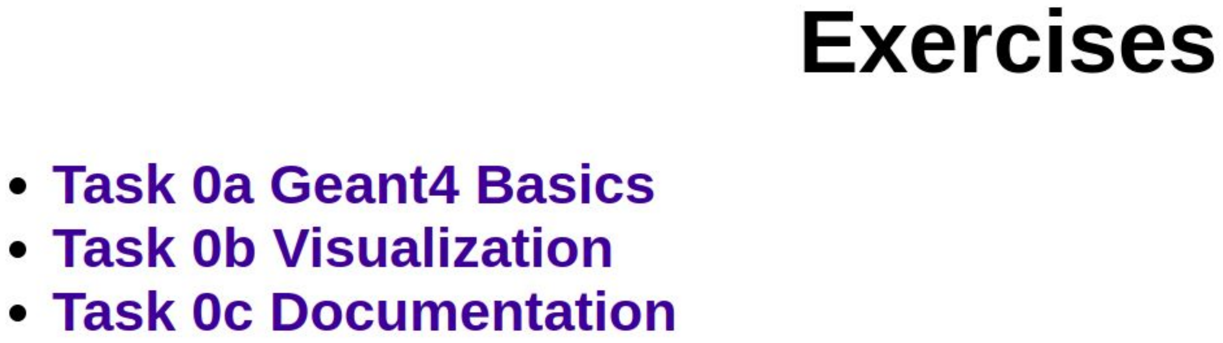
```
#
# To get nice view
# Make the "World" box invisible
/vis/geometry/set/visibility World 0 false
/vis/scene/notifyHandlers
# "Envelope" is transparent blue to represent water
/vis/geometry/set/colour Envelope 0 0 0 1 .3
/vis/scene/notifyHandlers
```

Task 0

Link to the Tasks:

http://geant4.lns.infn.it/alghero2023/task0

Task 0 - Geant4 basics


Exercises

- Task 0a Geant4 Basics
- Task 0b Visualization
- Task 0c Documentation