7/29/2013

Dipping your toes into the buildsystem and CMake

or how to be not completely lost

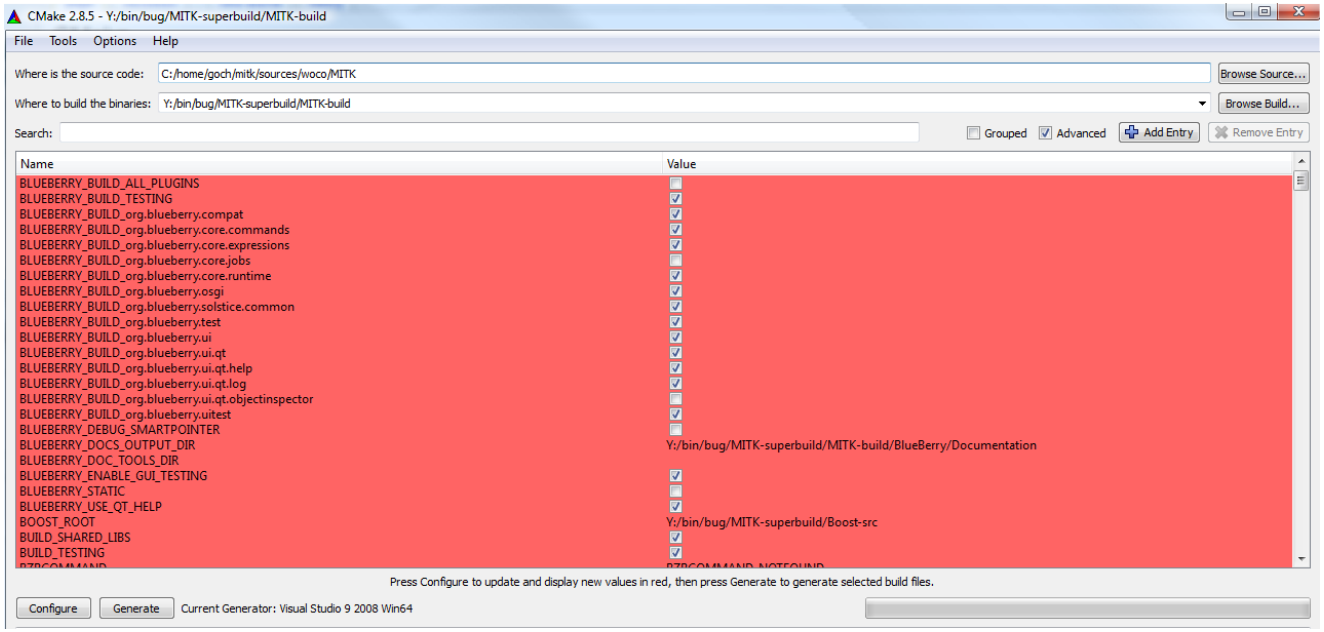
dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

Why even talk about it

- Many do not dare take on a build system bug
- Having a rough idea helps being not completely lost
- You can actually do quite a lot with it

The most basic stuff

- We use Cmake
  - Cross platform, compiler independent config files
  - Generates native config files for your compiler
  - You can easily configure by switching stuff on/off



Basic CMake I

- **SET**(VAR [VALUE])
- **MESSAGE**("Value of VAR: ${VAR}")
- Simple lists:
  SET(VAR ${VAR} "AnotherValue")

- **ADD_EXECUTABLE**(MyApplication source1.cpp source2.cpp)

- **ADD_LIBRARY**(MyLibrary libsource1.cpp …)
- **TARGET_LINK_LIBRARIES**(MyApplication MyLibrary AnotherLibrary)

- **INCLUDE_DIRECTORIES**(<path to c++ headers>)

Basic CMake II

- Defined by
  MACRO(NAME parameter1 parameter2 …)
  …
  ENDMACRO()

  FUNCTION(NAME parameters)
  ENDFUNCTION()

- In MITK there is

  MITK_CREATE_MODULE()
  MITK_CREATE_PLUGIN()

- Defined in mitk/CMake/*.cmake

Basic CMake III

- Each directory has a main file „CMakeLists.txt“
- **INCLUDE**(secondCMakeScript.cmake):
  direct parsing of the file in-place (as in C++)
- **ADD_SUBDIRECTORY**(directory):
  „directory“ has to include another CMakeLists.txt.
  Used for sub-projects.

Where is what?

- Most interesting directories:
  - /CMake/
    /BlueBerry/CMake/
    These contain functions and macros used in the rest of MITK
  - /CMakeExternals/
    If it is due to some external toolkit
- If it concerns installers
  - Grep ‚CPack‘
- It it concerns the applications
  - /Applications/<YourNameHere>/CMakeLists.txt
  - /Applications/<YourNameHere>/*.cmake

Somethin‘ is broke

Easy example:

```
foreach(mitk_app ${MITK_APPS})
  # extract target_dir and option_name
  string(REPLACE "^^" "\\;" target_info ${mitk_app})
  set(target_info_list ${target_info})
  list(GET target_info_list 0 target_dir)
  list(GET target_info_list 1 option_name)
  # check if the application is enabled
  if(${option_name} OR MITK_BUILD_ALL_APPS)
    # check whether application specific configuration files will be used
    if(use_project_cpack)
      # use files if they exist
      if(EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/Applications/${target_dir}/CPackOptions.cmake")
        include("${CMAKE_CURRENT_SOURCE_DIR}/Applications/${target_dir}/CPackOptions.cmake")
      endif()

      if(EXISTS "${PROJECT_SOURCE_DIR}/Applications/${target_dir}/CPackConfig.cmake.in")
        set(CPACK_PROJECT_CONFIG_FILE "${PROJECT_BINARY_DIR}/Applications/${target_dir}/CPackConfig.cmake")
        configure_file(${PROJECT_SOURCE_DIR}/Applications/${target_dir}/CPackConfig.cmake.in
                       ${CPACK_PROJECT_CONFIG_FILE} @ONLY)
        set(use_default_config OFF)
      endif()
    endif()
  # add link to the list
  list(APPEND CPACK_CREATE_DESKTOP_LINKS "${target_dir}")
  endif()
endforeach()
```

Somethin‘ is broke – how do I fix it

- The most important tools:
  - grep [Bash]
    - To find where the macro is called/where a variable is filled/where the error message originates
  - message() [CMake]
    - To debug the content of variables, will show during cmake run
- Common Sense:
  - Just because you can now tinker with the build system does not always mean you should