Subject: Building TheIDE with using CMake
Posted by Sender Ghost on Sat, 07 May 2011 18:32:13 GMT
View Forum Message <> Reply to Message

I see more people interested using CMake to build U++ applications.

This topic was mentioned a long time ago, discussed on U++ meetings, etc. Therefore, now I want to show what was done (in cross-platform way).

For the example, I took TheIDE application, which can be used to build other U++ applications, after its build.
Tested on Windows (for MSC9 and TDM-GCC 4.6.1 compilers) for 4179 svn revision (release version).

How to build:
1. Get uppsrc sources, either:
a) Download src archive from SourceForge.
b) Using svn checkout:

svn checkout -r 4179 http://upp-mirror.googlecode.com/svn/trunk/uppsrc upp
svn export upp uppsrc

c) Export ide packages using existing TheIDE, by clicking on "Output mode" drop list and then clicking on "All" button for selected export project directory.

2. If you want to build with rainbow support, get rainbow sources using svn checkout:

svn checkout -r 4179 http://upp-mirror.googlecode.com/svn/trunk/rainbow rainbow
svn export rainbow uppsrc/rainbow

In case of 1.c) you also need to copy Painter package directory to exported uppsrc directory.

3. Rename all *.icpp files to *.cpp files inside following directories (or use CMakeLists.txt files for reference from ISRC_LIST list):
CtrlCore, CtrlLib, ide/Browser, ide/Builders, PdfDraw, plugin/bmp, plugin/gif, plugin/jpg, plugin/png, RichEdit, RichText, Web.
In case of 2.: Painter.

Also, you can use following RenameFiles.cmake script:
Toggle Spoiler

```
# Recursive renaming of *.icpp to *.cpp files for specified directory
# Usage:
# cmake -P RenameFiles.cmake
# or
# cmake -DRENAME_PATH="SomePathWithFilesToRename" -P RenameFiles.cmake
# Note: -D options need to be specified before -P option, to apply.
```

```
cmake_minimum_required(VERSION 2.8)

if(NOT RENAME_PATH)
 # uppsrc directory by default
 set(RENAME_PATH uppsrc CACHE STRING "Directory with *.icpp files to rename" FORCE)
endif()

if(NOT EXISTS ${RENAME_PATH})
 message(FATAL_ERROR "Selected directory doesn't exists: ${RENAME_PATH}")
else()
 message("Selected directory: ${RENAME_PATH}")
endif()

file(GLOB_RECURSE FILE_LIST "${RENAME_PATH}/*.icpp")

list(LENGTH FILE_LIST files_count)

foreach(icppfile ${FILE_LIST})
 get_filename_component(filename ${icppfile} NAME_WE)
 get_filename_component(filepath ${icppfile} PATH)

 set(cppfile ${filepath}/${filename}.cpp)

 file(RENAME ${icppfile} ${cppfile})
endforeach()

if(${files_count} GREATER 0)
 message("Files renamed: " ${files_count})
else()
 message("No files to rename")
endif()
```

4. Copy CMakeLists.txt, *.cmake and build scripts files from attachments to source directory relative to uppsrc.

5. On Windows:

```
_start_here.bat
rem For GCC compiler
build.bat
rem For Microsoft Visual C++ 9.0 compiler
build_MSC9.bat
rem For rainbow support
build_MSC9_RAINBOW.bat
```

6. On FreeBSD (Linux, Unix, etc.):

./build.sh
For rainbow support
./build_RAINBOW.sh

If all runs ok, ide(.exe) executable would be found inside install directories, relative to running commands.

CMakeLists.txt files and build scripts could be downloaded from attachments.
Full archive could be found here.

To note:
There were linking problems for MinGW version of build script with rainbow support, which I didn't include to attachment. The MSC9 version as is.

Edit:
Updated links to 4179 release version. But also applicable to 4193 release version.

File Attachments
1) TheIDE_CMakeLists_r4179.zip, downloaded 711 times

Subject: Re: Building TheIDE with using CMake
Posted by Mindtraveller on Sat, 07 May 2011 18:50:56 GMT
View Forum Message <> Reply to Message

So you managed to build TheIDE with Digital Mars compiler?

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Sat, 07 May 2011 19:09:34 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Sat, 07 May 2011 20:50So you managed to build TheIDE with Digital Mars compiler?

Not tested for Digital Mars compiler (and didn't see it on CMake generators list).
Tested for GCC and MSC9 compilers only. Which was result from directly rewriting *.upp package files to CMakeLists.txt files.

May be, you can use it as reference.

Subject: Re: Building TheIDE with using CMake

Posted by chickenk on Sun, 08 May 2011 06:46:55 GMT
View Forum Message <> Reply to Message

Nice to see a new effort in that direction

When I have more time I'll have a look at your CMakefiles, I may find interesting ideas in it for my upp-waf build system.

You may also find some stuff in it, we never know...

Unfortunately I didn't have the opportunity to test it under windows (and I'm 100% sure it doesn't work there) so congrats for the cross-platform portability!

Lionel

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Sun, 08 May 2011 13:03:31 GMT
View Forum Message <> Reply to Message

Hello, Lionel.
Thanks for kind words.

I looked at your Waf build script. It's kind of "universal" build system for U++ packages.

The purposes of this thread is a bit different. It shows the possibility to create and build (using various native generators) complex U++ application(s), with using CMake.

Currently, one of the problems with *.icpp files solved directly, by renaming them to regular *.cpp files and building them alongside with "main" sources of application. It have some drawback, - additional step before building - and advantage, - correctly builds inside other IDEs, supported by CMake, without additional custom build steps (which also may be the possibility, otherwise).

Also, it possible to "globbing" C/C++ files to create list of sources to build, using aux_source_directory CMake command. But it is not used, - full sources provided in the list.

The "general" solution, for using with different U++ applications, can be some kind of CMake module(s), which resolves possible U++ build flags (e.g. flagGCC, flagMSC, - for compilers; flagBSD, flagFREEBSD, flagLINUX, flagOSX11, flagPOSIX - for platforms/OS, etc.) and checks include/linker directories. Now, it just have some CMake equivalents.

Honestly to say, I created CMakeLists.txt files for all current uppsrc packages, which gives me some kind of global review of used build flags/libraries, etc. But much of them, is kind of

```
set(SRC_LIST
 # C/C++ files
)
```

```
add_library(PackageName ${SRC_LIST})

set(USES_LIST) # dependent packages to link
set(LINK_LIST) # libraries to link

if(DEFINED USES_LIST)
 add_dependencies(PackageName ${USES_LIST})
endif(DEFINED USES_LIST)

if(DEFINED USES_LIST OR DEFINED LINK_LIST)
 target_link_libraries(PackageName ${USES_LIST} ${LINK_LIST})
endif(DEFINED USES_LIST OR DEFINED LINK_LIST)
```

Most of the "Core" packages is already there (Core, CtrlCore, CtrlLib, Draw, etc. packages).

Edit (08.05.2011): Added UppToCMakeLists.cmake script to attachments for partial conversion of U++ package to CMakeLists.txt file.
Edit: Fixed flag name for FreeBSD (flagFREEBSD).

File Attachments

1) UppToCMakeLists.zip, downloaded 855 times

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Tue, 22 Nov 2011 01:38:00 GMT
View Forum Message <> Reply to Message

The first topic updated to 4179 release version.

The main changes are:
- Various definition checks for build flags and helper functions inside cmake/modules directory.
- Global initialization list, which fills with renamed *.icpp files inside concrete packages, instead of adding them to main package explicitly. The main package just get such global initialization list for sources list.
- Rainbow support for CMakeLists.txt files.
- The RenameFiles.cmake script added to archive(s).

Subject: Building umk with using CMake
Posted by Sender Ghost on Tue, 22 Nov 2011 03:05:47 GMT
View Forum Message <> Reply to Message

The umk is another useful command-line application to build U++ packages.

In the attachment you could find full umk source files with CMakeLists.txt files to build.

Edit: Updated to 4353 revision.

File Attachments
1) umk_r4353_full.7z, downloaded 542 times

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Wed, 23 Nov 2011 14:23:28 GMT
View Forum Message <> Reply to Message

In this message I will explain possible reason why renaming of *.icpp files was needed.

According to ""init" files in packages" topic, it is possible to use init file inside particular package to resolve *.icpp files dependencies. But practically, they are useful for command-line builds only. Mainly, because of lack of *.cpp file extension (e.g. init.cpp), which prevents using it inside another IDEs.

I made CMake scripts for three cases to demonstrate this problem:
1. Using init file of package directly by setting source properties as for CXX compiler and linker.
2. Using init.cpp file inside current package as an copy of init file. Including init.cpp file to source files to compile.
3. Using init.cpp file inside package binary directory with #include path to package init file. Including init.cpp file to source files to compile.

The CMakeLists.txt and build script files for mentioned cases you could find in the attachment.
To use, just create a backup of uppsrc directory (in case of testing) and copy files from particular case directory. Renaming of *.icpp files not needed.

To note:
The 2 and 3 cases are workable solutions for IDEs and command-line builds.

Edit: Fixed URL link.

File Attachments
1) TheIDE_CMakeLists_r4179_cases.zip, downloaded 558 times

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Fri, 13 Feb 2015 12:16:46 GMT
View Forum Message <> Reply to Message

Hi!

I'm trying to use CMake to have QtCreator as my main UPP IDE.

Compilation & link is ok, but at runtime the GUI theme doesn't seems to apply.
I attached what I get for SQLApp: the window background is darker and popup menu background

is totally black..

When I strace the application with processmonitor, I see that the one compiled with theIDE seems to look for the windows themes (right part of the image): there are a bunch of lines like:

CreateFile;C:\Windows\Resources\Themes\aero\aero.msstyles etc.)
QueryBasicInformationFile;C:\Windows\Resources\Themes\aero\aero.msstyles

that are not present in the version I compiled with CMake.

I also try forcing the theme with baazaar/Theme: it changes some colors but its not working.

I'm using the lastest nightly build of UPP.

Any hints of what I need to do to fix that?

Thx!
Damien.

File Attachments

1) cmake_theide_procmon.jpg, downloaded 620 times

Subject: Re: Building TheIDE with using CMake
Posted by dolik.rce on Fri, 13 Feb 2015 13:37:45 GMT
View Forum Message <> Reply to Message

cyrion wrote on Fri, 13 February 2015 13:16Hi!

I'm trying to use CMake to have QtCreator as my main UPP IDE.

Compilation & link is ok, but at runtime the GUI theme doesn't seems to apply.
I attached what I get for SQLApp: the window background is darker and popup menu background is totally black..
Hi Cyrion,

I can't tell you how to fix it, because I don't know. What I can tell you, is the reason: It is because of the *.icpp files (or those with this extension before renaming, as described above) not being linked correctly. They must be linked directly into the final executable, otherwise some of the registration code is removed as unused, while it is actually needed. You'll need to check what exactly happens in the build process and how those files are linked.

Best regards,
Honza

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Fri, 13 Feb 2015 16:43:07 GMT
View Forum Message <> Reply to Message

Hello, Damien.

Your screenshot is quite big (wide, actually). I think, better to create small public image with URL to bigger private
image, but Bazaar attachments restricted to 1 attachment at the moment, therefore this might require to use external
source for this.

This topic was created in 2011 year (which is about 4 years ago). The proposed examples and CMakeLists.txt files wasn't
created automatically at the time of creation. Currently, they adapted to 4179 revision only (and some others with the same
file lists and dependencies).

What you could use to fix the issue in your screenshot is properly using *.icpp/init files, e.g. what explained in the above message.
You could find the related functions inside of uppsrc/cmake/modules directories of attachment, e.g. add_init_file function.

The automatic creation of CMakeLists.txt files might require to create correspondence between U++ and CMake defines,
methods to resolve library dependencies (or by using them directly as U++ TheIDE does). Also, there might be a binary files
used inside of U++ packages, which could require to write special parser for them.

Currently, this topic just show, that this is possible (manually).

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Fri, 13 Feb 2015 17:26:21 GMT
View Forum Message <> Reply to Message

Thank you Honza, I miss that in the previous post.

About the icpp files, what I have done is simply create a new file like that:
For example SqlCtrl_init.icpp, I create a new file named SqlCtrl_init.icpp.cpp with the content:

#include "SqlCtrl_init.icpp"

That way, I did not mess UPP sources.

But I didn't notice I have to "force" linking to the objs. I'll try that!

Btw, I have a lot of warnings like the following:

warning LNK4221: This object file does not define any previously undefined public symbols, so it will not be used by any link operation that consumes this library

Not sure if its related..

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Fri, 13 Feb 2015 17:34:52 GMT
View Forum Message <> Reply to Message

Hello Sender Ghost :)
Sorry for the big image! Maybe I could have shrinked it, or yes, put it on some external server. I edited my post to put in on hostingpics.net.
In fact I merged my three screenshots into one big image because of the "one attachement" restriction.

About the CMakeFiles, I created them manually, I didn't used the one provided in this thread.
I had a quick look - way too quick actually - at the cmake modules: I'll check that too, I think that is the solution!

Thank you very much!

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Fri, 13 Feb 2015 17:55:16 GMT
View Forum Message <> Reply to Message

Yeah, it's working :)

Quite ugly, but I manually added the icpp files to the application target to make it sure they are linked with it, and everything ok!

```
include_directories( . )

set( sources
 SQLApp.h
 SQLApp.sch
 query.cpp
 book.cpp
 borrow.cpp
 main.cpp
      Theme.cpp
      Theme.h
```

```
	C:/upp/uppsrc/CodeEditor/CRegister.icpp.cpp
	C:/upp/uppsrc/Core/Core_init.icpp.cpp
	C:/upp/uppsrc/CtrlCore/CtrlCore.icpp.cpp
	C:/upp/uppsrc/CtrlLib/CtrlLib.icpp.cpp
	C:/upp/uppsrc/plugin/bmp/BmpReg.icpp.cpp
	C:/upp/uppsrc/plugin/png/pngreg.icpp.cpp
	C:/upp/uppsrc/Report/ReportI.icpp.cpp
	C:/upp/uppsrc/RichText/RichImage.icpp.cpp
	C:/upp/uppsrc/SqlCtrl/SqlCtrl_init.icpp.cpp
)

add_executable( SQLApp WIN32 ${sources} )
```

Now I just have to make something automatic & clean :) I'm checking the previous posts...

Thank you very much all!

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Fri, 13 Feb 2015 18:23:03 GMT
View Forum Message <> Reply to Message

cyrion wrote on Fri, 13 February 2015 17:55Yeah, it's working :)
Quite ugly, but I manually added the icpp files to the application target to make it sure they are linked with it, and everything ok!

I updated the link to "init" files in packages topic for above message.

What you did with *.icpp files are created automatically by current TheIDE inside of init U++ package files.

For example, I wrote following add_init_function, from case 3:

```
function(add_init_file init_file copied_file)
 # Creates ${init_file}.cpp inside binary directory with #include path to ${init_file}
 # and returns the path to copied_file
 set(input_file ${CMAKE_CURRENT_SOURCE_DIR}/${init_file})
 if(EXISTS "${input_file}")
  set(output_file "${CMAKE_CURRENT_BINARY_DIR}/${init_file}.cpp")
  file(WRITE "${output_file}" "#include \"${input_file}\"\n")
  set(${copied_file} "${output_file}" PARENT_SCOPE)
 endif()
endfunction()
```

For example, inside of CtrlCore CMakeLists.txt file:

```
# Header files
set(H_LIST
)
# Compilable files
set(SRC_LIST
)
# The *.icpp files. Just to edit/view them inside of IDEs
set(ISRC_LIST
 CtrlCore.icpp
)

set_source_files_properties(${H_LIST} ${ISRC_LIST} PROPERTIES HEADER_FILE_ONLY ON)

add_init_file(init INIT_FILE)
add_library(CtrlCore ${INIT_FILE} ${SRC_LIST} ${H_LIST} ${ISRC_LIST})

# And so on
```

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Fri, 13 Feb 2015 18:30:35 GMT
View Forum Message <> Reply to Message

Yep, I saw that, I'm on it!

Your script is working, and creates the ini files in the bin dir, but they are not linked with the main app exe, so it's not working directly for me here.
Also, since it's constantly updating the init.cpp files, CMake recompile them all the time. Annoying... :)

So what I'm trying to do currently is creating a new target with all the init.cpp, or maybe a global variable holding all the files, in order to add them (or the target) to the dependencies of the executable..

[EDIT] Btw I don't receive notifications of replies in the thread.. I unsubscribed/subscribed but nothing..

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Fri, 13 Feb 2015 18:46:15 GMT
View Forum Message <> Reply to Message

cyrion wrote on Fri, 13 February 2015 18:30Your script is working, and creates the ini files in the bin dir, but they are not linked with the main app exe

This is wrong assumption. What I wrote in message 44261 is excerpt from actual attachment.

cyrion wrote on Fri, 13 February 2015 18:30Also, since it's constantly updating the init.cpp files, CMake recompile them all the time.
The method was created for IDEs, mainly to configure CMake files once, which allows to view/edit and build U++ package files, when needed. I didn't have issue, which you mentioned.

No need to create competition here. This topic was created for different purpose (e.g. to show the possibility in 2011 year). The semi-automatic method already exists. There are other solutions also, which not use CMake.

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Fri, 13 Feb 2015 20:21:27 GMT
View Forum Message <> Reply to Message

What I said might look like as rudeness, but it is not.

There were many CMake releases (and possible compatibility issues for previous versions) from 2011 year.
Constantly updating the attachments for new versions of U++ and CMake is not productive, in my opinion, especially when done manually.

This topic is open for questions, related to the topic messages. But they might be quite outdated, when used with modern U++ and CMake.

If you have a (new) method or a way how to use CMake and U++, then better to create a new topic about this, in my opinion.

Overall, I glad that somebody is interested in what said and done in this topic (in broad sense of this word).

Subject: Re: Building TheIDE with using CMake
Posted by cyrion on Sat, 14 Feb 2015 00:05:10 GMT
View Forum Message <> Reply to Message

Quote:
What I said might look like as rudeness, but it is not.

No problem, I was not there for a few hours, I just saw your answer now.
Thank you for the updated links!

Quote:
There were many CMake releases (and possible compatibility issues for previous versions) from 2011 year.
Constantly updating the attachments for new versions of U++ and CMake is not productive, in my

opinion, especially when done manually.

This topic is open for questions, related to the topic messages. But they might be quite outdated, when used with modern U++ and CMake.

If you have a (new) method or a way how to use CMake and U++, then better to create a new topic about this, in my opinion.

From my side, I'm just happy I found a solution to my problem with your help!
In fact I started creating my CMakeFiles without knowing you already explored that problem, then I found this forum entry when I looked for a solution to my colors bug.
Thought it was the good place to ask - and in fact it was :) Thanks again!

I really love the UPP way to create nice and small GUI apps without external dependencies, and I just wanted to make this working with QtCreator.
TheIDE is great, fast & smooth ; I like the nested code blocks with the different colors, the << completion and the indispensable statistics and elapsed time :) but IMHO it lacks some recent functionalities I'm really used to: easy global/local search & replace, simultaneous multiple lines editing (aka "column mode" or "block selection mode"), refactoring, ctrk/k+key stuff, VIM integration, etc. Also JOM for parallel builds is probably as fast as BLITZ at first sight, maybe faster. I'll check that but I first have to add the PCH to be fair :)

So my goal here was not to automatically create CMakeFiles from upp projects, I'm good with the idea of maintaining my CMakeFiles manually, but of course that's only a workaround.

About my final solution - no competition here ;) - I updated your function add_init_file for this one:

```
function(create_cpps_from_icpps )
    file( GLOB icpp_files RELATIVE "${CMAKE_CURRENT_SOURCE_DIR}"
"${CMAKE_CURRENT_SOURCE_DIR}/*.icpp"   )
    foreach( icppFile ${icpp_files} )
        set(output_file "${CMAKE_CURRENT_BINARY_DIR}/${icppFile}.cpp")
        file(WRITE "${output_file}" "#include
\"${CMAKE_CURRENT_SOURCE_DIR}/${icppFile}\"\n")
    endforeach()
endfunction()
```

It creates the icpp.cpp files in the binary dir. Each one of them having the name of the original icpp it makes easier to catch compile / link errors compared to the multiple init.cpp that had the same name.

So as you may have noticed, I do not rely on Mirek ini files. That's probably not a good idea, but when I tried using them it added a lot of unnecessary (in my case) dependencies, missing symbols and also multiple symbols definitions that broke everything.
I didn't really digged very far to check why, but I think ini files can potentially add the same cpp multiple times.
The way I do makes possible to only link with the required modules, and not the others.

To finish, my main app looks like that :

```
set( sources
 SQLApp.h
 SQLApp.sch
 query.cpp
 book.cpp
 borrow.cpp
 main.cpp
)

file( GLOB_RECURSE ini_files "${CMAKE_CURRENT_BINARY_DIR}/../*.icpp.cpp"   )

add_executable( SQLApp WIN32 ${sources} ${ini_files} )
```

The GLOB_RECURSE stuff is not very pretty, but it does the job: it links the icpp to the app the exact same way that fixed my problem at first.

For now its working, but I only use a subset of UPP, namely :

```
add_subdirectory( "plugin/z"  )
add_subdirectory( Core )
add_subdirectory( Draw )
add_subdirectory( "plugin/bmp"  )
add_subdirectory( "plugin/png"  )
add_subdirectory( RichText )
add_subdirectory( CtrlCore )
add_subdirectory( CtrlLib  )
add_subdirectory( "plugin/sqlite3"  )
add_subdirectory( Sql  )
add_subdirectory( SqlCtrl  )
add_subdirectory( Report  )
add_subdirectory( CodeEditor  )
add_subdirectory( "plugin/pcre" )
add_subdirectory( "plugin/zip" )
```

So maybe it will break in the future, I don't know!

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Thu, 26 May 2016 06:16:22 GMT
View Forum Message <> Reply to Message

Hi,

Based on this discussion I created bash script to generate CMakeList.txt from the Ultimate++ project - see https://github.com/CoolmanCZ/upp_cmake

Script was tested on Ubuntu 14.04LTS with GCC 4.9+. Any comments are welcome.

Radek

Subject: Re: Building TheIDE with using CMake
Posted by Sender Ghost on Thu, 26 May 2016 13:03:56 GMT
View Forum Message <> Reply to Message

Hello Radek.

coolman wrote on Thu, 26 May 2016 06:16
Based on this discussion I created bash script to generate CMakeList.txt from the Ultimate++ project - see https://github.com/CoolmanCZ/upp_cmake

Looks like, you did a lot of work to automate CMakeLists.txt files generation. Very nice.

I managed to build TheIDE on FreeBSD 10.2 with using "Ninja" CMake generator (just for example), with some fixes. I added "link_directories(/usr/local/lib)" (because this is where local libraries reside on FreeBSD by default) and some checks for Clang compiler:
Toggle patch

```
diff -ruN upp_cmake.orig/GenerateCMakeFiles-lib.sh upp_cmake/GenerateCMakeFiles-lib.sh
--- upp_cmake.orig/GenerateCMakeFiles-lib.sh 2016-05-26 13:09:40
+++ upp_cmake/GenerateCMakeFiles-lib.sh
@@ -983,11 +983,12 @@
     echo >> ${OFN}
     echo '# Set compiler flags' >> ${OFN}
     echo 'if ( "${CMAKE_CXX_COMPILER_ID}" STREQUAL "Clang" )' >> ${OFN}
+    echo '  set ( CMAKE_COMPILER_IS_CLANG TRUE )' >> ${OFN}
     echo '  set ( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wno-logical-op-parentheses" )'
>> ${OFN}
     echo 'endif()' >> ${OFN}

     echo >> ${OFN}
-    echo 'if ( CMAKE_COMPILER_IS_GNUCC )' >> ${OFN}
+    echo 'if ( CMAKE_COMPILER_IS_GNUCC OR CMAKE_COMPILER_IS_CLANG)' >> ${OFN}
     echo '  set ( CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE}
-std=c++11 -O3 -ffunction-sections -fdata-sections" )' >> ${OFN}
     echo '  set ( CMAKE_CXX_FLAGS_DEBUG "${CMAKE_CXX_FLAGS_DEBUG} -std=c++11
-O0" )' >> ${OFN}
     echo 'elseif ( MSVC )' >> ${OFN}
@@ -1045,6 +1046,10 @@
     echo '      include_directories( ${PNG_INCLUDE_DIR} )' >> ${OFN}
     echo "      list ( APPEND main_${LINK_LIST} \${PNG_LIBRARIES} )" >> ${OFN}
```

```
    echo '  endif()' >> ${OFN}
+   echo >> ${OFN}
+   echo '  if ( ${CMAKE_SYSTEM_NAME} MATCHES BSD )' >> ${OFN}
+   echo '      link_directories( /usr/local/lib )' >> ${OFN}
+   echo '  endif()' >> ${OFN}
    echo 'endif()' >> ${OFN}

    echo >> ${OFN}
```

Then I configured upp_cmake/GenerateCMakeFiles.sh file for FreeBSD, used it, generated CMake building files and make a build. Just example for current U++ nightly source files:

```
% cd upp_cmake
% sed -i '.bak' -e '/^UPP_SRC_DIR/s/=/?&/' -e 's/LINUX/BSD -DflagFREEBSD/'
GenerateCMakeFiles.sh
% wget http://www.ultimatepp.org/downloads/upp-x11-src-9886.tar.gz
% tar -xf upp-x11-src-9886.tar.gz
% env UPP_SRC_DIR=upp-x11-src-9886/uppsrc bash GenerateCMakeFiles.sh
% mkdir build && cd build && cmake -G Ninja ../
% ninja
```

Edit:
Submitted pull request with proposed patch in this message. And it was committed later.

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Fri, 15 Jul 2016 18:55:07 GMT
View Forum Message <> Reply to Message

Hi,

Bash script to generate CMakeLists.txt for the Ultimate++ project now support MINGW and MSVC compilation and link flags.

BR, Radek

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Thu, 25 Aug 2016 10:44:54 GMT
View Forum Message <> Reply to Message

Hi,

Some information about the progress with the script:

Support

* New Core with C++11 build (require GCC 4.9+)
* Release or debug build
* Binary resource support (BINARY, BINARY_MASK, BINARY_ARRAY)
* Cross compile support (require MINGW GCC 4.9+)
* (MSYS2) MINGW support
* Generated CMakeLists.txt files can be used to create a MS Visual C++ project
* Generated CMakeLists.txt files are generated only for dependent modules of the processed Ultimate++ project
* Create a distribution package
* Build shared libraries as the target (DLL, SO)

Limitation
* Ultimate++ source tree and directory of the project should be in the same directory as the generator scripts during build (you can use symlinks)
* Some options are not taken into account when generating CMakeLists

TODO
* Support of the pre-compiled headers (PCH)

Best regards, Radek

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Mon, 29 Aug 2016 10:31:58 GMT
View Forum Message <> Reply to Message

Hi,

The script now supports precompiled headers for GCC and Clang. Here are some measurements when building the ide:
Results using new PCH refactored code are red marked.

Upp source code:
git-svn-id: svn://ultimatepp.org/upp/trunk@10191 f0d560ea-af0d-0410-9eb7-867de7ffcac7 (last commit: .doc .ctrlib Change several image in GUI tutorial)

Notebook Dell Latitude E5440:
(cat /proc/cpuinfo | grep 'model name' | uniq)
model name : Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
(cat /proc/cpuinfo | grep processor | wc -l)
4

Ubuntu 14.04 LTS Linux:
(uname -a)
Linux ul001172 4.3.0-040300-generic #201511020949 SMP Mon Nov 2 15:04:56 UTC 2015 i686 i686 i686 GNU/Linux
(cmake --version)
cmake version 2.8.12.2

Using original Makefile
linux, gcc version 4.9.3 (Ubuntu 4.9.3-8ubuntu2~14.04)
00:24:24 - orig Makefile - 1 core (size: 11 872 144)

Using generated CMakeLists.txt
linux, gcc version 4.9.3 (Ubuntu 4.9.3-8ubuntu2~14.04)
00:27:12 - pch disable - 1 core (size: 15 339 076)
00:24:36 - pch enable - 1 core (size: 15 334 980)

00:10:39 - pch disable - 4 core (size: 15 339 076)
00:13:40 - pch disable - 4 core (size: 15 339 076)
00:12:40 - pch disable - 4 core (size: 15 339 076) (refactored code)
00:08:07 - pch enable - 4 core (size: 15 334 980)
00:09:28 - pch enable - 4 core (size: 15 334 980)
00:09:43 - pch enable - 4 core (size: 15 334 980)
00:06:14 - pch enable - 4 core (size: 15 334 980) (refactored code)
00:05:57 - pch enable - 4 core (size: 15 334 980) (refactored code)

linux, clang version 3.5.0-4ubuntu2~trusty2 (tags/RELEASE_350/final) (based on LLVM 3.5.0)
00:19:30 - pch disable - 1 core (size: 12 939 896)
00:15:48 - pch disable - 1 core (size: 12 939 896)
00:08:08 - pch enable - 1 core (size: 12 939 896)

00:09:06 - pch disable - 4 core (size: 12 939 896)
00:09:20 - pch disable - 4 core (size: 12 939 896) (refactored code)
00:04:36 - pch enable - 4 core (size: 12 939 896)
00:03:56 - pch enable - 4 core (size: 12 939 896)
00:03:54 - pch enable - 4 core (size: 12 939 896)
00:04:10 - pch enable - 4 core (size: 12 939 896) (refactored code)
00:04:40 - pch enable - 4 core (size: 12 939 896) (refactored code)

linux, cross-compilation, mingw32 gcc version 4.9.2 (GCC)
00:20:13 - pch disable - 1 core (size: 12 535 296)
00:13:02 - pch enable - 1 core (size: 12 513 280)

00:10:05 - pch disable - 4 core (size: 12 535 296)
00:10:11 - pch disable - 4 core (size: 12 535 296) (refactored code)
00:08:22 - pch enable - 4 core (size: 12 513 280)
00:08:19 - pch enable - 4 core (size: 12 513 280)
00:08:14 - pch enable - 4 core (size: 12 513 280)
00:04:30 - pch enable - 4 core (size: 12 513 280) (refactored code)
00:05:10 - pch enable - 4 core (size: 12 513 280) (refactored code)

linux, cross-compilation, mingw64 gcc version 4.9.2 (GCC)
00:20:28 - pch disable - 1 core (size: 13 800 448)
00:09:12 - pch enable - 1 core (size: 13 778 672)

00:11:13 - pch disable - 4 core (size: 13 800 448)
00:10:46 - pch disable - 4 core (size: 13 800 448)
00:10:35 - pch disable - 4 core (size: 13 800 448) (refactored code)
00:04:57 - pch enable - 4 core (size: 13 778 672)
00:04:40 - pch enable - 4 core (size: 13 778 672)
00:04:45 - pch enable - 4 core (size: 13 778 672)
00:05:05 - pch enable - 4 core (size: 13 778 672) (refactored code)
00:04:47 - pch enable - 4 core (size: 13 778 672) (refactored code)

Best regards, Radek

Subject: Re: Building TheIDE with using CMake
Posted by Klugier on Thu, 01 Feb 2018 22:03:17 GMT
View Forum Message <> Reply to Message

Hello Coolman,

I tried to use your script with upp git mirror. But it deosn't work. I called following command:

cd uppmirror
uppbox/Cmake/GenerateCMakeFiles-lib.sh uppsrc/ide/ide.upp "-DflagGUI -DflagGCC -DflagLINUX -DflagPOSIX -DflagSHARED"

I also copy your CMake generator to uppbox/lpbuild directory
uppbox/Cmake/

It returns not output and cmake list is not generated either. I believe the minimal error output is needed.

Please provide support, if you can.

Sincerely,
Klugier

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Fri, 02 Feb 2018 06:09:09 GMT
View Forum Message <> Reply to Message

Klugier wrote on Thu, 01 February 2018 23:03Hello Coolman,

I tried to use your script with upp git mirror. But it deosn't work. I called following command:

cd uppmirror
uppbox/Cmake/GenerateCMakeFiles-lib.sh uppsrc/ide/ide.upp "-DflagGUI -DflagGCC -DflagLINUX -DflagPOSIX -DflagSHARED"

I also copy your CMake generator to uppbox/lpbuild directory
uppbox/Cmake/

It returns not output and cmake list is not generated either. I believe the minimal error output is needed.

Please provide support, if you can.

Sincerely,
Klugier

Hi Klugier,

GenerateCMakeFiles-lib.sh is the source script. Take a look at the example.sh, how to set up the build.

BR, Radek

Subject: Re: Building TheIDE with using CMake
Posted by coolman on Thu, 30 Jul 2020 10:05:09 GMT
View Forum Message <> Reply to Message

Hi all,

I would like to point out that the script is still being developed and expanded. See the upp_cmake repository. I would like to highlight the faster processing of UPP packaging sections and the shorter running of the script itself.

BR, Radek
