Subject: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Thu, 23 Dec 2010 19:35:39 GMT
View Forum Message <> Reply to Message

Hello.

After watching slides from 2010 LLVM Developer's Meeting for libclang, I thought to create Upp package which can be used as binding to Clang library. When talked with dolik.rce, I understood that this could be useful for other people too. This topic is result of such efforts.

The archive contains following packages:
ClangBase - as base for Clang library, which binds to it.
SyntaxCheck - console application to syntax check a file for errors/warnings, which prints LLVM/Clang diagnostics. This is fixed and changed version of syntax-check example from slides to run with Ultimate++ Framework.

Requirements:
- Installed/compiled version of LLVM and Clang 3.4 version (or either version from 2.8 to 3.3 with LLVM_2_8 to LLVM_3_3 build flag).
- Build method points to include and library directories for LLVM/Clang (e.g. INCLUDE directories: /usr/local/include; LIB directories: usr/local/lib).

The 2.8 and 2.9 versions tested on FreeBSD 8.1 (for Clang 2.8 and Clang 2.9 port versions) and Windows XP by me and Arch Linux by dolik.rce.
The versions from 3.0 to 3.4 tested on Windows XP.

These packages considered experimental.

Changelog:
2014-01-25 Added support for LLVM 3.4 release version, which is now by default. Introduced LLVM_3_3 build flag for LLVM 3.3 version.
2013-06-24 Added support for LLVM 3.3 release version, which is now by default. Introduced LLVM_3_2 build flag for LLVM 3.2 version.
2013-01-03 Added support for LLVM 3.2 release version, which is now by default. Introduced LLVM_3_1 build flag for LLVM 3.1 version.
2012-06-04 Added support for LLVM 3.1 release version, which is now by default. Introduced LLVM_3_0 build flag for LLVM 3.0 version.
2011-12-10 Added support for LLVM 3.0 release version, which is now by default. Introduced LLVM_2_9 build flag for LLVM 2.9 version.
2011-04-07 Added support for LLVM 2.9 release version, which is now by default. Introduced LLVM_2_8 build flag for LLVM 2.8 version and LLVM build flag for current (development) version.
2011-01-08 Added object oriented interface for libclang.
2010-12-26 Added MSC library option. Added GCC library option for Windows to link statically.

File Attachments

1) Upp_Clang_v3.4.zip, downloaded 559 times

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Fri, 24 Dec 2010 16:05:12 GMT
View Forum Message <> Reply to Message

How to build and install LLVM/Clang on Windows:
I will describe how to install it using GCC compiler.

You need to download and install following software first:
- CMake build system from http://www.cmake.org.
- TDM GCC compiler from http://www.tdm-gcc.tdragon.net, Nuwen GCC compiler from http://nuwen.net or MinGW GCC compiler from http://www.mingw.org.
- Python from http://www.python.org.
- SVN command line client from http://www.sliksvn.com or http://subversion.tigris.org.

Then download "BuildSystem_v3.4.zip" archive from attachments and extract it to "C:\BuildSystem" directory, for example.
Next, you need to edit "_start_here.bat" file for "System configurable settings" section for current paths of installed software. After running it, you will see command prompt.

First of you need to checkout sources from svn repository for particular version:
Toggle versions- Run "checkout.bat" to check out current (development) version.
- Run "checkout_2_8.bat" to check out 2.8 release version.
- Run "checkout_2_9.bat" to check out 2.9 release version.
- Run "checkout_3_0.bat" to check out 3.0 release version.
- Run "checkout_3_1.bat" to check out 3.1 release version.
- Run "checkout_3_2.bat" to check out 3.2 release version.
- Run "checkout_3_3.bat" to check out 3.3 release version.
- Run "checkout_3_4.bat" to check out 3.4 release version.
For example, you run "checkout_3_4.bat" first.

Then you can build and install LLVM/Clang for selected version:
Toggle versions- Run "build.bat" to build current (development) version.
- Run "build_2_8.bat" to build 2.8 release version.
- Run "build_2_9.bat" to build 2.9 release version.
- Run "build_3_0.bat" to build 3.0 release version.
- Run "build_3_1.bat" to build 3.1 release version.
- Run "build_3_2.bat" to build 3.2 release version.
- Run "build_3_3.bat" to build 3.3 release version.
- Run "build_3_4.bat" to build 3.4 release version.
For example, you run "build_3_4.bat" second.

If selected batch files runs without errors, then inside build and install directories you will see llvm directories for selected versions.

To update current development sources to latest versions you could run "update.bat" file when needed and start to build them from second step.

Note:
You need to run batch files inside command prompt after running "_start_here.bat" file.

How to configure installed LLVM/Clang on Windows for TheIDE:
Assuming you did previous steps to install 3.4 release version.
1. Start TheIDE.
2. Open "Setup->Build Methods.." menu.
3. Press insert key on the keyboard to add new build method and name it "LLVM34_MinGW".
4. Select GCC builder.
5. Add following directories for selected tabs:
PATH - executable directories:
C:\MinGW\bin
C:\BuildSystem\install\llvm_3_4\bin
INCLUDE directories:
C:\MinGW\include
C:\BuildSystem\install\llvm_3_4\include
LIB directories:
C:\MinGW\lib
C:\BuildSystem\install\llvm_3_4\lib
6. Use "LLVM34_MinGW" build method to build software relative to LLVM 3.4 version.

To use Upp package binding (ClangBase) with Clang library you need to check "C:\BuildSystem\install\llvm_3_4\lib" for "liblibclang.dll.a" and "liblibclang.dll" files. Else you need to find them inside "C:\BuildSystem\build\llvm_3_3" directories (e.g. bin and lib) and place them respectively.
If you compile shared (or static on Windows) version of ClangBase package, then you need to place "liblibclang.dll" file into "%windir%\system32" directory or near executable file. Else you could configure %PATH% environment variable to directory with "liblibclang.dll" file. Starting from 3.1 version, the "liblibclang.dll" file installed to bin directory, therefore you could extend LIB directories with:
C:\BuildSystem\install\llvm_3_4\bin
To note: the shared library for 2.9 release version is "libclang.dll".

Tested for CMake v2.8.12, Nuwen GCC v4.8.1, Python v2.7.5, Slik SVN v1.7.10.

File Attachments

1) BuildSystem_v3.4.zip, downloaded 558 times

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sgifan on Tue, 28 Dec 2010 00:44:23 GMT
View Forum Message <> Reply to Message

nice explanation.

works as explained.

it works if the goal is to contribute to llvm or clang.

but using clang to compile u++ samples does not work.

to use the compiled clang the explanation about the build method should also include to enter clang as the compiler name.

in this case it is really clang and not gcc that is invoked to compile.

but then it fails to compile all samples
examples with PainterSVGDemo:

```
In file included from C:\upp\bazaar\PainterSvg_demo\SvgDemo.cpp:1:
In file included from C:\upp\bazaar\PainterSvg_demo/Examples.h:4:
In file included from C:\upp\uppsrc/CtrlLib/CtrlLib.h:4:
In file included from C:\upp\uppsrc/CtrlCore/CtrlCore.h:4:
In file included from C:\upp\uppsrc/RichText/RichText.h:4:
In file included from C:\upp\uppsrc/Draw/Draw.h:6:
C:\upp\uppsrc/Core/Core.h:39:10: fatal error: 'typeinfo' file not found
#include <typeinfo>
         ^
1 error generated.
PainterSvg_demo: 1 file(s) built in (0:01.07), 1076 msecs / file, duration = 1139 msecs, parallelization 0%
```

is there a way to use clang to compile u++ code ??

thanks

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Tue, 28 Dec 2010 14:48:02 GMT
View Forum Message <> Reply to Message

Hello, Sgifan.

As you can see, I explained how to compile LLVM/Clang on Windows and didn't say about using Clang compiler. That's it for Windows operating system (for now, at least).
Yes, you can use Clang to compile C++ source code (even whole Upp TheIDE) on Unix-like operating systems.

The purpose of this thread is to use Clang library (libclang) with Upp build system (as package). The libclang library is not a whole compiler, but also can suffer from its features for particular operating system.
To understand about libclang, please watch slides or video(s) from 2010 LLVM Developer's Meeting.

To use libclang with Clang compiler you need to compile it first using Clang, not GCC (but there is a possibility with the same ABI between them, of course). Also, you need to specify correct paths for Clang.
As you may know, latest GCC version which use Clang is 4.2 version (by developers). But there is a possibility to use newer versions. For example, if you install LLVM/Clang with TDM-GCC v4.5.2 then you can specify following paths for TheIDE build method:
PATH - executable directories:
C:\BuildSystem\install\llvm\bin
C:\MinGW\bin
INCLUDE directories:
C:\BuildSystem\install\llvm\include
C:\MinGW\lib\gcc\mingw32\4.5.2\include\c++
C:\MinGW\lib\gcc\mingw32\4.5.2\include\c++\mingw32
C:\MinGW\lib\gcc\mingw32\4.5.2\include\c++\backward
C:\MinGW\include
C:\MinGW\lib\gcc\mingw32\4.5.2\include
C:\MinGW\lib\gcc\mingw32\4.5.2\include-fixed
LIB directories:
C:\BuildSystem\install\llvm\lib
C:\MinGW\lib

And now you can change compiler name to "clang++" for build method and try to build C++ source code.
For example, I did it in the past and had linker errors.

If you want to improve possibility to compile C++ source code on Windows then there is LLVM

Edit (2011-04-07):
From LLVM 2.9 release, it possible to build applications on Windows. But there are stability issues of compiled applications (especially on exit) and multithreading source code issues (which tested for TDM GCC v4.5.2).

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Thu, 07 Apr 2011 16:45:24 GMT
View Forum Message <> Reply to Message

There is new LLVM 2.9 version.
The corresponding archives were updated.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Sat, 10 Dec 2011 11:00:32 GMT
View Forum Message <> Reply to Message

There is new LLVM 3.0 version.
The corresponding archives were updated.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Mon, 04 Jun 2012 19:51:48 GMT
View Forum Message <> Reply to Message

There is new LLVM 3.1 version on 22th May 2012. The corresponding archives were updated.

In this version "CXString clang_getDiagnosticCategoryName(unsigned Category);" function was deprecated and recommended to use "CXString clang_getDiagnosticCategoryText(CXDiagnostic diag);" function instead.

Also I tried to build C/C++ applications with Clang v3.1 compiler on Windows. It was successfull for Clang, compiled with GCC v4.6.1 and by using binaries, headers and libraries from GCC v4.7.0 (e.g. from nuwen.net). With following paths for TheIDE build method:
PATH - executable directories:
C:\BuildSystem\install\llvm_3_1\bin
C:\MinGWN\bin
INCLUDE directories:
C:\MinGWN\include\c++\4.7.0
C:\MinGWN\include\c++\4.7.0\i686-pc-mingw32
C:\MinGWN\include\c++\4.7.0\backward
C:\MinGWN\lib\gcc\i686-pc-mingw32\4.7.0\include
C:\MinGWN\include
C:\MinGWN\lib\gcc\i686-pc-mingw32\4.7.0\include-fixed
C:\BuildSystem\install\llvm_3_1\include
LIB directories:
C:\MinGWN\lib
C:\BuildSystem\install\llvm_3_1\lib
And "clang++" compiler name.

But there are issues with multithreading source code, such as: error: "thread-local storage is unsupported for the current target". The U++ TheIDE, compiled with GUI build flag, works well.
To note: To disable various warning messages there is "-w" compiler option. To restrict various error messages there is "-ferror-limit=n" compiler option, where n is maximum number of errors; 0 - without restrictions.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by tojocky on Sat, 14 Jul 2012 18:49:47 GMT
View Forum Message <> Reply to Message

Very interesting.

Did you test on performance?

For example for me on Ubuntu gcc compiles better than clang

in clang I have:
(20235484 B) linked in (0:03.40)
OK. (0:54.74)

and in GCC I have:
(15365512 B) linked in (0:03.28)
OK. (1:02.77)

The result with GCC is better by application size and worst by compilation performance.

I'm not sure with runtime performance.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Sat, 14 Jul 2012 19:54:52 GMT
View Forum Message <> Reply to Message

tojocky wrote on Sat, 14 July 2012 20:49Did you test on performance?
The AddressBook example (r5188), Optimal build, on Windows XP:
- TDM MinGW GCC v4.6.1:
AddressBook.exe (2508288 B) linked in (0:03.12)

OK. (3:26.01)
- Nuwen MinGW GCC v4.7.0:
AddressBook.exe (2520064 B) linked in (0:01.62)

OK. (4:02.50)
- Clang v3.1 with diagnostics:
AddressBook.exe (2880512 B) linked in (0:01.28)

OK. (2:49.35)
Note: TheIDE freezes for some time, after linking stage.
- Clang v3.1 without diagnostics:
AddressBook.exe (2880512 B) linked in (0:01.25)

OK. (2:43.45)
- MSC9 compiler:
AddressBook.exe (1243136 B) linked in (0:02.21)

OK. (1:53.06)
Other related topics:
- "Clang - Performance", "Clang vs Other Open Source Compilers" on clang.llvm.org.
- "GCC 4.6/4.7 vs. LLVM-Clang 3.0/3.1 Compilers" on phoronix.com.
- "Clang vs GCC for Linux Development: Compare and Contrast" on stackoverflow.com.

Edit: Added some results on Windows XP.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Thu, 03 Jan 2013 03:46:13 GMT
View Forum Message <> Reply to Message

There is new LLVM 3.2 version on 20 Dec 2012. The corresponding archives were updated.

This version added libxml2 dependency, which is optional (for xml validation) and not required to build.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Didier on Sun, 26 May 2013 18:11:09 GMT
View Forum Message <> Reply to Message

Hi all,

I am trying to make clang work and I run into the following error:
In file included from /users/didier/upp.out/SRC_MII/CtrlLib/CLANG.Blitz.Gui.Shared/$blitz.cpp:3:
In file included from /users/didier/upp/uppsrc/CtrlLib/LabelBase.cpp:1:
In file included from /users/didier/upp/uppsrc/CtrlLib/CtrlLib.h:4:
In file included from /users/didier/upp/uppsrc/CtrlCore/CtrlCore.h:4:
In file included from /users/didier/upp/uppsrc/RichText/RichText.h:4:
In file included from /users/didier/upp/uppsrc/Draw/Draw.h:6:
In file included from /users/didier/upp/uppsrc/Core/Core.h:213:
In file included from /usr/bin/../lib/gcc/i686-redhat-linux/4.7.2/../../../../include/c++/4.7.2/string:53:
In file included from
/usr/bin/../lib/gcc/i686-redhat-linux/4.7.2/../../../../include/c++/4.7.2/bits/basic_string.h:40:
/usr/bin/../lib/gcc/i686-redhat-linux/4.7.2/../../../../include/c++/4.7.2/ext/atomicity.h:48:45: error: use of undeclared identifier '__ATOMIC_ACQ_REL'
  { return __atomic_fetch_add(__mem, __val, __ATOMIC_ACQ_REL); }

Obviously my BM file is incomplete/wrong but what is wrong ??

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Sun, 26 May 2013 19:04:31 GMT
View Forum Message <> Reply to Message

Hello, Didier.
Didier wrote on Sun, 26 May 2013 20:11Obviously my BM file is incomplete/wrong but what is wrong ??
Usually, I use following command to get default include directories for GCC compiler:

gcc -v -fsyntax-only -x c++ test.c
where test.c is path to file with following contents:

```
#include <stdio.h>

int main() {
 return 0;
}
```

The excerpt of output might look like follows:
#include "..." search starts here:
#include <...> search starts here:
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/include/c++
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/include/c++/mingw32
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/include/c++/backward
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/include
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/../../../../include
 c:\mingw\bin\../lib/gcc/mingw32/4.7.1/include-fixed
End of search list.Then the default include directories (for concrete compiler version) are:
c:\mingw\lib\gcc\mingw32\4.7.1\include\c++
c:\mingw\lib\gcc\mingw32\4.7.1\include\c++\mingw32
c:\mingw\lib\gcc\mingw32\4.7.1\include\c++\backward
c:\mingw\lib\gcc\mingw32\4.7.1\include
c:\mingw\include
c:\mingw\lib\gcc\mingw32\4.7.1\include-fixed

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Didier on Mon, 27 May 2013 19:06:37 GMT
View Forum Message <> Reply to Message

Hi Sender Ghost,

I will try you're suggestion.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Mon, 24 Jun 2013 00:15:46 GMT
View Forum Message <> Reply to Message

There is new LLVM 3.3 version on 17 Jun 2013. The corresponding archives were updated.

At the moment of writing this message, there are no Clang v3.3 pre-built binaries for MinGW (as of 184675 revision). I managed to build it by excluding AArch64 LLVM target:

set CUR_DIR=%~p0

```
set GENERATOR="MinGW Makefiles"
set GENERATOR_DIR="%CUR_DIR%build\llvm_3_3"
set GENERATOR_FILES="%CUR_DIR%source\llvm_3_3"
set INSTALL_DIR="%CUR_DIR%install\llvm_3_3"
rem all targets, except AArch64, to exclude the error
set
TARGETS_TO_BUILD="ARM;CppBackend;Hexagon;Mips;MBlaze;MSP430;NVPTX;PowerPC;Sp
arc;SystemZ;X86;XCore"
set GENERATOR_DEFINES=-DCMAKE_INSTALL_PREFIX:PATH=%INSTALL_DIR%
-DCMAKE_BUILD_TYPE:STRING="RELEASE"
-DLLVM_TARGETS_TO_BUILD:STRING=%TARGETS_TO_BUILD%

cd %GENERATOR_DIR%
cmake -G %GENERATOR% %GENERATOR_DEFINES% %GENERATOR_FILES%
mingw32-make
mingw32-make install/fast
```

To note: there is also possible to use "Ninja" CMake generator instead of "MinGW Makefiles" (for example, to use all CPU cores for building):

```
Toggle batch file
set CUR_DIR=%~p0
set GENERATOR="Ninja"
set GENERATOR_DIR="%CUR_DIR%build\llvm_3_3"
set GENERATOR_FILES="%CUR_DIR%source\llvm_3_3"
set INSTALL_DIR="%CUR_DIR%install\llvm_3_3"
rem all targets, except AArch64, to exclude the error
set
TARGETS_TO_BUILD="ARM;CppBackend;Hexagon;Mips;MBlaze;MSP430;NVPTX;PowerPC;Sp
arc;SystemZ;X86;XCore"
set GENERATOR_DEFINES=-DCMAKE_INSTALL_PREFIX:PATH=%INSTALL_DIR%
-DCMAKE_BUILD_TYPE:STRING="RELEASE"
-DLLVM_TARGETS_TO_BUILD:STRING=%TARGETS_TO_BUILD%

cd %GENERATOR_DIR%
cmake -G %GENERATOR% %GENERATOR_DEFINES% %GENERATOR_FILES%
ninja
ninja install
```

To create "Ninja build system" from source code on Windows, just use configured "_start_here.bat" file from the above message (and re2c scanner generator binary, if needed):

```
cd ninja
python bootstrap.py --platform=mingw
```

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by koldo on Sun, 20 Apr 2014 13:54:44 GMT
View Forum Message <> Reply to Message

Hello Sender

I have tried to install clang on TheIDE in windows unsuccessfully. I have used the precompiled files.

I have added the include, lib and path directories, and added clang++ as "Compiler Name". However the compiler does not find #include <typeinfo> in Core.h. Just removing clang++ as "Compiler Name" and the files are compiled without problem (with MinGW).

May you help me?

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Sun, 20 Apr 2014 20:19:53 GMT
View Forum Message <> Reply to Message

Hello Koldo.

koldo wrote on Sun, 20 April 2014 15:54I have tried to install clang on TheIDE in windows unsuccessfully. I have used the precompiled files.

You didn't say about which precompiled files you used, but if they are official v3.4 "Pre-built Binaries: Clang for Windows" from llvm.org download page, then they didn't work here also (perhaps, because of 64-bit compiler build, which don't support 32-bit Windows operating system).
koldo wrote on Sun, 20 April 2014 15:54I have added the include, lib and path directories, and added clang++ as "Compiler Name". However the compiler does not find #include <typeinfo> in Core.h.

The plain "include" directory from MinGW is not enough to configure Clang compiler on Windows, in case of unsupported search directories in the source code. To solve this with TheIDE, you could add full include directories, as explained in 40030 message.
koldo wrote on Sun, 20 April 2014 15:54Just removing clang++ as "Compiler Name" and the files are compiled without problem (with MinGW).

Yes, because of "c++" compiler name by default for GCC builder.

In conclusion, I recommend to download/install necessary software and build LLVM/Clang by yourself, e.g. as explained in 30322 message, to try it out. But you could try to find different precompiled LLVM/Clang (e.g. from http://www.drangon.org/mingw/) or use following temporary links (by clicking "Direct download link"):
LLVM/Clang v3.3 release (32-bit, 86.5 Mb).
LLVM/Clang v3.4 release (32-bit, 128 Mb).

The last working version for GCC builder of TheIDE is 3.3. The 3.4 version uses error messages

instead of warning messages for unsupported compiler/linker flags (e.g. "-finline-limit", "-mwindows", "-mconsole", etc.) and will not build (e.g., without excluding/changing some flags for GCC builder). But this is not true, if you just need to use libclang library, compiled by MinGW GCC, with GCC compiler.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by koldo on Tue, 22 Apr 2014 10:58:59 GMT
View Forum Message <> Reply to Message

Hello Sender

Thank you for your aid.

With 3.3 version from http://www.drangon.org/mingw/, clang++ complained because of the lack of libgcc_s_seh-1.dll.

With your v3.3 file, llvm package compiles well after adding include folders, although clang++ emits many "'&&' within '||' [-Wlogical-op-parentheses]" warnings. Unfortunately the link fails because of "undefined reference to `_Unwind_Resume'"

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Tue, 22 Apr 2014 22:41:35 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 22 April 2014 12:58With your v3.3 file, llvm package compiles well after adding include folders, although clang++ emits many "'&&' within '||' [-Wlogical-op-parentheses]" warnings

It's possible to disable warnings with "-w" compiler option (or to specify "-ferror-limit=n", where n is number of diagnostics, as said in "Clang compiler user's manual").
koldo wrote on Tue, 22 April 2014 12:58Unfortunately the link fails because of "undefined reference to `_Unwind_Resume'"
Looks like, you used compiler with a different stack unwinding method: Dwarf-2 (DW2) or SJLJ (setjmp/longjmp). I used Nuwen GCC compiler (10.3 distribution version, before it became 64-bit only, but available 10.4 version might be ok) to build 3.3 version.

In case, if you used TDM-GCC 4.8.1 compiler, I rebuild 3.3 version of LLVM/Clang with it and got the same "undefined reference to `_Unwind_Resume'" errors. So, it might be compiler dependent.

I fixed issue with "unknown argument" for Clang (v3.4 release) with following changes:
Toggle patch file
diff -ruN llvm_3_4.orig/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td llvm_3_4/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td
--- llvm_3_4.orig/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td 2014-01-05 02:28:17 +0400

```
+++ llvm_3_4/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td 2014-04-22 21:07:04
+0400
@@ -82,7 +82,7 @@

 def err_drv_I_dash_not_supported : Error<
   "'%0' not supported, please use -iquote instead">;
-def err_drv_unknown_argument : Error<"unknown argument: '%0'">;
+def err_drv_unknown_argument : Warning<"unknown argument: '%0'">;
 def err_drv_invalid_value : Error<"invalid value '%1' in '%0'">;
 def err_drv_invalid_int_value : Error<"invalid integral value '%1' in '%0'">;
 def err_drv_invalid_remap_file : Error<
diff -ruN llvm_3_4.orig/tools/clang/lib/Frontend/CompilerInvocation.cpp
llvm_3_4/tools/clang/lib/Frontend/CompilerInvocation.cpp
--- llvm_3_4.orig/tools/clang/lib/Frontend/CompilerInvocation.cpp 2014-01-05 02:25:13 +0400
+++ llvm_3_4/tools/clang/lib/Frontend/CompilerInvocation.cpp 2014-04-22 21:07:54 +0400
@@ -1644,7 +1644,7 @@
   for (arg_iterator it = Args->filtered_begin(OPT_UNKNOWN),
         ie = Args->filtered_end(); it != ie; ++it) {
     Diags.Report(diag::err_drv_unknown_argument) << (*it)->getAsString(*Args);
-    Success = false;
+    //Success = false;
   }

   Success = ParseAnalyzerArgs(*Res.getAnalyzerOpts(), *Args, Diags) && Success;
diff -ruN llvm_3_4.orig/tools/clang/tools/driver/cc1as_main.cpp
llvm_3_4/tools/clang/tools/driver/cc1as_main.cpp
--- llvm_3_4.orig/tools/clang/tools/driver/cc1as_main.cpp 2014-01-05 02:28:27 +0400
+++ llvm_3_4/tools/clang/tools/driver/cc1as_main.cpp 2014-04-22 21:08:41 +0400
@@ -167,7 +167,7 @@
   for (arg_iterator it = Args->filtered_begin(cc1asoptions::OPT_UNKNOWN),
         ie = Args->filtered_end(); it != ie; ++it) {
     Diags.Report(diag::err_drv_unknown_argument) << (*it) ->getAsString(*Args);
-    Success = false;
+    //Success = false;
   }

   // Construct the invocation.
```

which turns such kind of error to warning message and doesn't stop. But while it compiles and builds, the result executables starts with exception error. For reference, I uploaded 3.4 version (with applied patch) to the following temporary links:
LLVM/Clang v3.4 release (32-bit, compiled by TDM GCC 4.8.1, 118.8 Mb).
LLVM/Clang v3.4 release (32-bit, compiled by Nuwen GCC 4.8.1, 128 Mb).

Then I decided to apply the same changes to development version of Clang (3.5):
Toggle patch file
diff -ruN llvm.orig/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td

```
llvm/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td
--- llvm.orig/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td 2014-04-22 22:39:01 +0400
+++ llvm/tools/clang/include/clang/Basic/DiagnosticDriverKinds.td 2014-04-22 22:42:48 +0400
@@ -87,7 +87,7 @@

 def err_drv_I_dash_not_supported : Error<
   "'%0' not supported, please use -iquote instead">;
-def err_drv_unknown_argument : Error<"unknown argument: '%0'">;
+def err_drv_unknown_argument : Warning<"unknown argument: '%0'">;
 def err_drv_invalid_value : Error<"invalid value '%1' in '%0'">;
 def err_drv_invalid_int_value : Error<"invalid integral value '%1' in '%0'">;
 def err_drv_invalid_remap_file : Error<
diff -ruN llvm.orig/tools/clang/lib/Frontend/CompilerInvocation.cpp
llvm/tools/clang/lib/Frontend/CompilerInvocation.cpp
--- llvm.orig/tools/clang/lib/Frontend/CompilerInvocation.cpp 2014-04-22 22:39:06 +0400
+++ llvm/tools/clang/lib/Frontend/CompilerInvocation.cpp 2014-04-22 22:43:25 +0400
@@ -1708,7 +1708,7 @@
   for (arg_iterator it = Args->filtered_begin(OPT_UNKNOWN),
         ie = Args->filtered_end(); it != ie; ++it) {
     Diags.Report(diag::err_drv_unknown_argument) << (*it)->getAsString(*Args);
-    Success = false;
+    //Success = false;
   }

   Success = ParseAnalyzerArgs(*Res.getAnalyzerOpts(), *Args, Diags) && Success;
diff -ruN llvm.orig/tools/clang/tools/driver/cc1as_main.cpp
llvm/tools/clang/tools/driver/cc1as_main.cpp
--- llvm.orig/tools/clang/tools/driver/cc1as_main.cpp 2014-04-22 22:39:41 +0400
+++ llvm/tools/clang/tools/driver/cc1as_main.cpp 2014-04-22 22:43:41 +0400
@@ -168,7 +168,7 @@
   for (arg_iterator it = Args->filtered_begin(cc1asoptions::OPT_UNKNOWN),
         ie = Args->filtered_end(); it != ie; ++it) {
     Diags.Report(diag::err_drv_unknown_argument) << (*it) ->getAsString(*Args);
-    Success = false;
+    //Success = false;
   }

   // Construct the invocation.
```

The simple console applications starts without exception errors, but others (which compiles by 3.3, 3.4 versions) is not compiled. For reference, I uploaded 3.5 development version (with applied patch) to the following temporary link:
LLVM/Clang v3.5 devel (32-bit, compiled by Nuwen GCC 4.8.1, 137.5 Mb).

Subject: Re: Upp package binding for LLVM/Clang library (libclang)

Posted by koldo on Wed, 23 Apr 2014 06:12:06 GMT
View Forum Message <> Reply to Message

Hello Sender

For sure the problem is in the MinGW compiler version. I use TDM 4.8.1.

So I see the only way to get LLVM is by compiling it all . I hate CMake.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Wed, 23 Apr 2014 09:42:03 GMT
View Forum Message <> Reply to Message

koldo wrote on Wed, 23 April 2014 08:12So I see the only way to get LLVM is by compiling it all

Well, it quite easy, if you automate it a bit with some batch files.

For example, I just configure paths (to GCC, CMake, Python, SVN) for _start_here.bat file (or copy it to a different name for particular compiler). Then I choose or copy build.bat file to a different name and configure (if necessary) GENERATOR (type of CMake generator for particular build system, e.g. MinGW Makefiles, Ninja, etc.), GENERATOR_DIR (where to place build files), GENERATOR_FILES (root of LLVM sources path), INSTALL_DIR (where to install files). In next step I run _start_here.bat file and type build.bat, push <ENTER>. If there are no errors, it will be done after some time.

The same steps are true for other projects, which use CMake. The difference is in configuration.

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by mirek on Wed, 23 Apr 2014 11:18:49 GMT
View Forum Message <> Reply to Message

Hi,

sorry to interrupt here, but recently I am playing with the idea of creating Win32 CLANG based U++ release, probably with many 3rd party stuff included, like SDL, OpenSSL etc... Simply said, single download to get everything working...

Would that be viable? How good is clang on windows nowadays?

Mirek

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by unodgs on Wed, 23 Apr 2014 12:44:11 GMT
View Forum Message <> Reply to Message

It's quite good now and progressing in this area (but I don't know if the latest build parses windows headers without problems). It also accepts cl command line and can be easily used with VS. The newest builds can be found here: http://llvm.org/builds/

Subject: Re: Upp package binding for LLVM/Clang library (libclang)
Posted by Sender Ghost on Wed, 23 Apr 2014 12:56:44 GMT
View Forum Message <> Reply to Message

Hello Mirek.
mirek wrote on Wed, 23 April 2014 13:18How good is clang on windows nowadays?

In my experience, it depends from particular LLVM/Clang version. Latest 3.3 and 3.4 release versions have some exception errors for started applications, while some development versions runs without errors.

For example, LLVM/Clang v3.5 devel from 43006 message is able to build U++ "Simple address book application" (download), TheIDE (download) in single-threaded optimal mode.
But TheIDE in multi-threaded optimal mode fails with following error(s):
Toggle error(s)In file included from C:\upp\uppsrc\ide\Common\ComDlg.cpp:1:
In file included from C:\upp\uppsrc\ide\Common/Common.h:4:
In file included from C:\upp\uppsrc\ide/Core/Core.h:4:
In file included from C:\upp\uppsrc\Esc/Esc.h:4:
In file included from C:\upp\uppsrc\Core/Core.h:336:
C:\upp\uppsrc\Core/InVector.hpp:31:8: error: thread-local storage is unsupported for the current target
extern thread__ int64 invector_cache_serial_;
^
C:\upp\uppsrc\Core/Mt.h:14:18: note: expanded from macro 'thread__'
#define thread__ __thread

So, without further investigation, it's not quite possible to build multi-threaded applications on Windows operating system.

There are other issues, like compile/runtime speed, executable sizes, etc.

There is a possibility to build Clang by Clang compiler, may be even MinGW libraries by Clang compiler (but not sure, because of some unsupported GCC extensions on Clang). There are LLVM linker and "libc++" C++ Standard Library in development. I didn't test them on Windows to say about results.

Personally, I have successful experience on FreeBSD with LLVM/Clang. It's more mature here, than on Windows. But from development point of view, this all was required to adapt/patch many FreeBSD ports by developers/maintainers.

mirek wrote on Wed, 23 April 2014 13:18Would that be viable?

The idea is interesting, but realisation is complicated, in current development stage.
You could try it out, may be even with Visual C++ compiler, as Uno said. Then it will be possible to get more affirmative answer.

Edit:
I tried clang-cl.exe with MSC9 builder with following application:
Toggle source code#include <iostream>

```
using namespace std;

int main()
{
 int factorialnumber = 0;

 cout << "Please enter a number: ";
 cin >> factorialnumber;

 int zero_count = 0;

 for (int five_factor = 5; five_factor <= factorialnumber; five_factor *= 5)
  zero_count += factorialnumber / five_factor;

 cout << "Trailing zeros of " << factorialnumber;
 cout << "! is " << zero_count << endl;

 return 0;
}
```

And got following errors:
Toggle errorserror: cannot mangle RTTI descriptors for type 'failure' yet
error: cannot mangle the name of type 'failure' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'runtime_error' yet
error: cannot mangle the name of type 'runtime_error' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'exception' yet
error: cannot mangle the name of type 'exception' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'failure' yet
error: cannot mangle the name of type 'failure' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'runtime_error' yet
error: cannot mangle the name of type 'runtime_error' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'exception' yet
error: cannot mangle the name of type 'exception' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'failure' yet
error: cannot mangle the name of type 'failure' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'runtime_error' yet
error: cannot mangle the name of type 'runtime_error' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'exception' yet

```
error: cannot mangle the name of type 'exception' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'runtime_error' yet
fatal error: too many errors emitted, stopping now [-ferror-limit=]
44 warnings and 20 errors generated.
```

To be able change compiler name for TheIDE to "clang-cl", I changed MSC builder with following source code:

```
Toggle changesdiff -ruN uppsrc/ide/Builders/Builders.h uppsrc/ide/Builders/Builders.h
--- uppsrc/ide/Builders/Builders.h 2014-01-28 21:38:52 +0400
+++ uppsrc/ide/Builders/Builders.h 2014-04-23 18:56:50 +0400
@@ -120,6 +120,7 @@
  virtual bool   Preprocess(const String& package, const String& file, const String& target, bool asmout);

  String CmdLine(const String& package, const Package& pkg);
+ String CompilerName() const;
  String MachineName() const;
  String LinkerName() const;
  String PdbPch(String package, int slot, bool do_pch) const;
diff -ruN uppsrc/ide/Builders/MscBuilder.icpp uppsrc/ide/Builders/MscBuilder.icpp
--- uppsrc/ide/Builders/MscBuilder.icpp 2014-03-18 09:12:15 +0400
+++ uppsrc/ide/Builders/MscBuilder.icpp 2014-04-23 18:56:36 +0400
@@ -61,23 +61,7 @@

 String MscBuilder::CmdLine(const String& package, const Package& pkg)
 {
- String cc;
- if(HasFlag("ARM"))
-  cc = "clarm";
- else
- if(HasFlag("MIPS"))
-  cc = "clmips";
- else
- if(HasFlag("SH3"))
-  cc = "shcl /Qsh3";
- else
- if(HasFlag("SH4"))
-  cc = "shcl /Qsh4";
- else
- if(HasFlag("MSC8ARM"))
-  cc = "cl -GS- ";
- else
-  cc = HasFlag("INTEL") ? "icl" : "cl";
+ String cc(CompilerName());
 // TRC 080605-documentation says Wp64 works in 32-bit compilation only
 // cc << (IsMsc64() ? " -nologo -Wp64 -W3 -GR -c" : " -nologo -W3 -GR -c");
  cc << " -nologo -W3 -GR -c";
```

```
@@ -86,6 +70,17 @@
  return cc;
 }

+String MscBuilder::CompilerName() const
+{
+ if(!IsNull(compiler))  return compiler;
+ if(HasFlag("ARM"))     return "clarm";
+ if(HasFlag("MIPS"))    return "clmips";
+ if(HasFlag("SH3"))     return "shcl /Qsh3";
+ if(HasFlag("SH4"))     return "shcl /Qsh4";
+ if(HasFlag("MSC8ARM")) return "cl -GS-";
+ return HasFlag("INTEL") ? "icl" : "cl";
+}
+
 String MscBuilder::MachineName() const
 {
  if(HasFlag("ARM"))     return "ARM";
```

This changes are not needed if you copy clang-cl.exe to cl.exe or use cl.exe from llvm\msbuild-bin path. They are the same executables, but with different names, which activates Visual C++ or GCC (in case of clang++.exe) driver mode (as shown on tools/driver/driver.cpp 218, 224 and 227 lines).
