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).