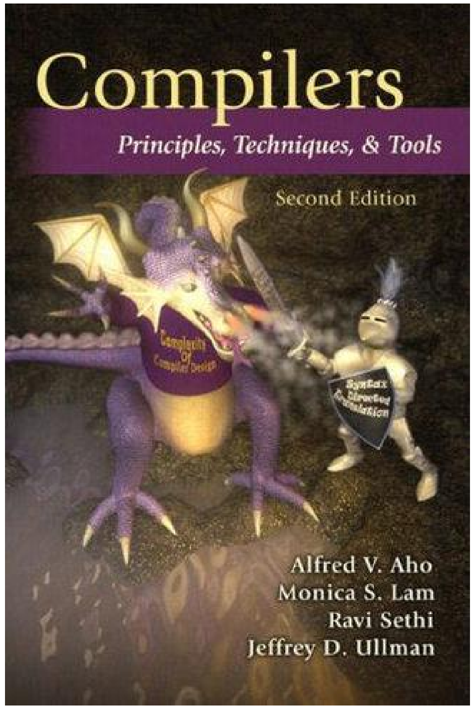
clang-cl

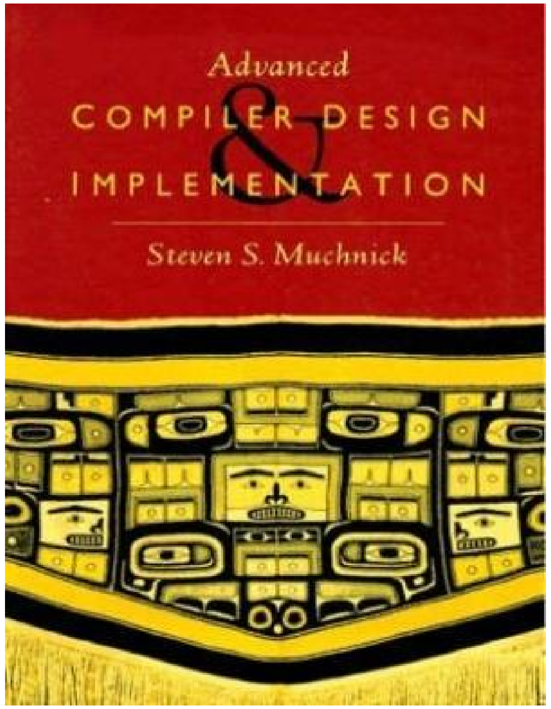
What it is, how it works, and how to use it

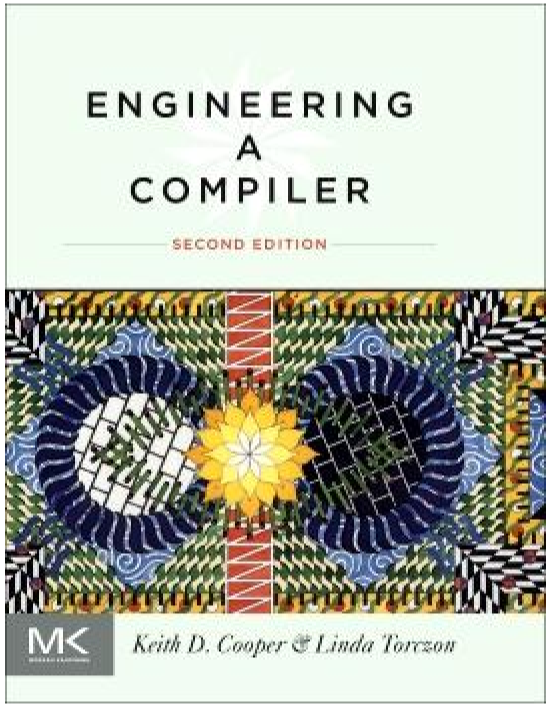
Hans Wennborg

Euro-LLVM 2014

Why give a talk about a compiler driver?

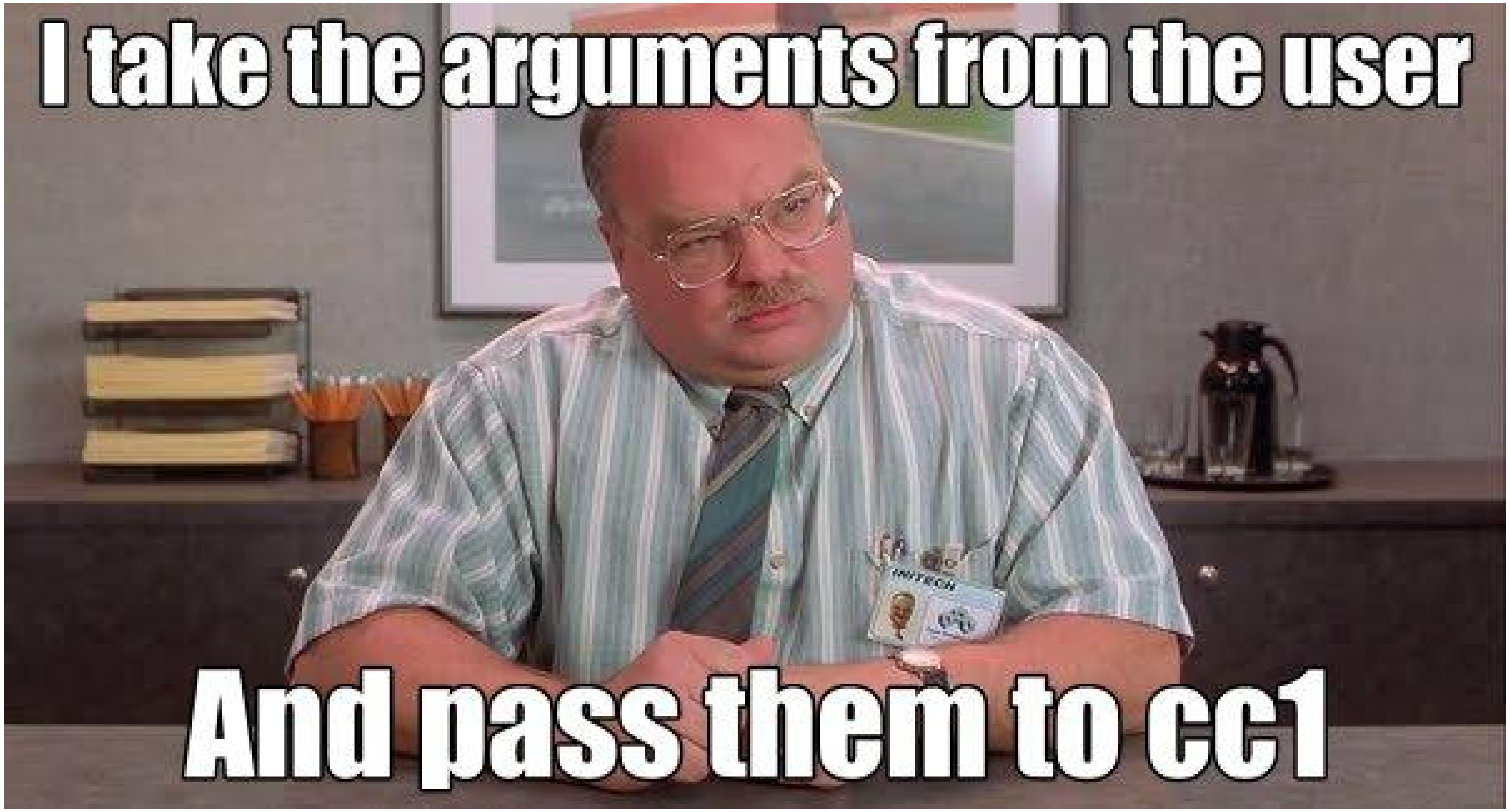






Chapters about drivers = 0

What is the driver good for?

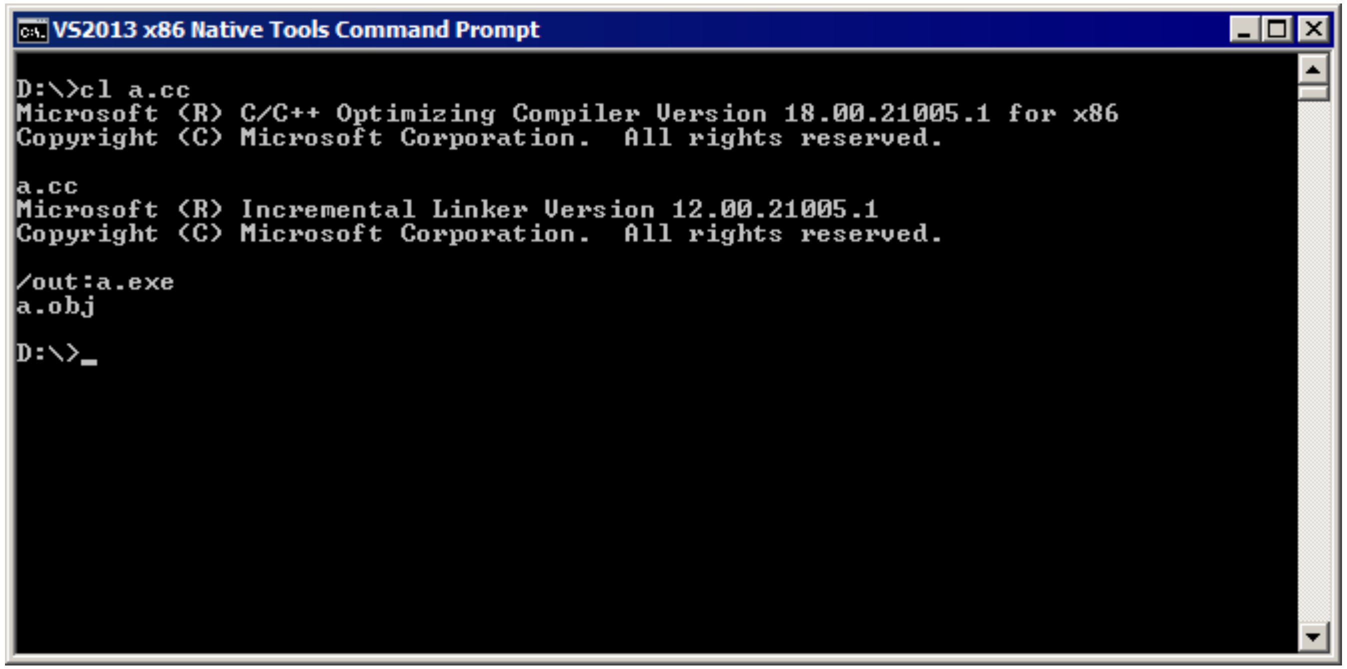


Do we even need the driver?

What did the driver ever do for us?

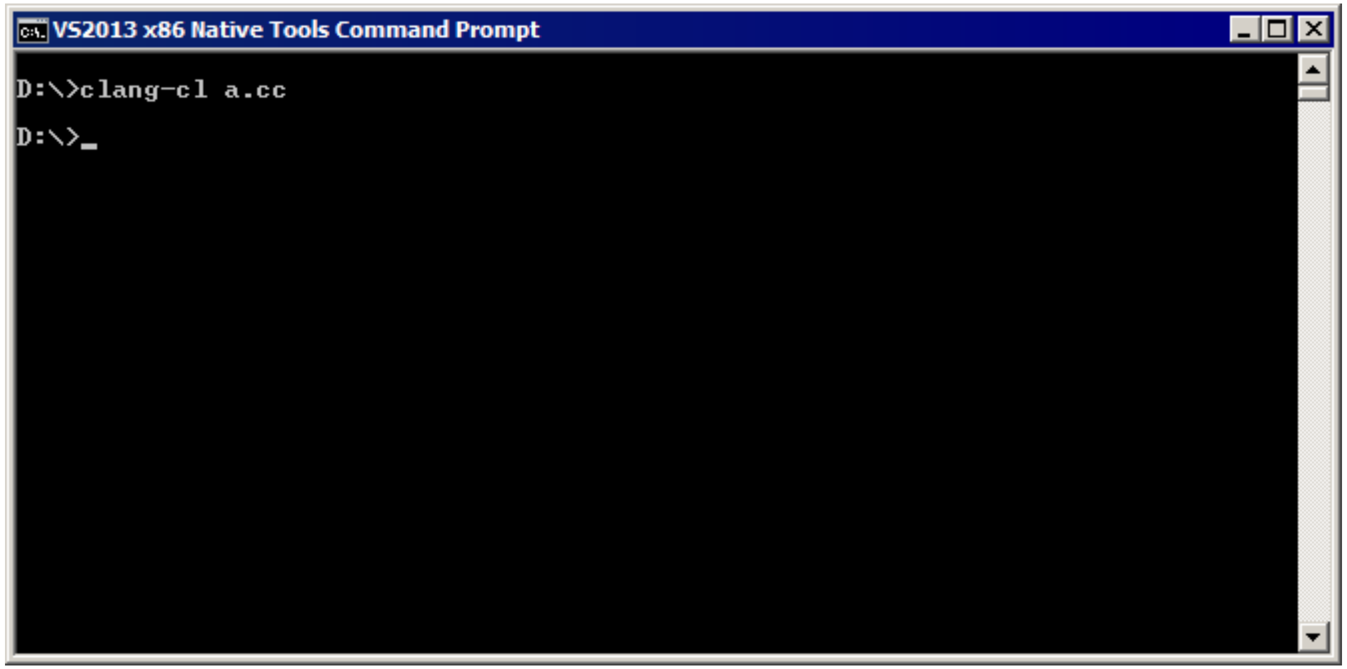
- The driver allows us to build real programs
- It is a great compatibility layer
- Chrome Linux/Mac clang build very similar to gcc
- Now trying to do the same on Windows.

This is cl.exe



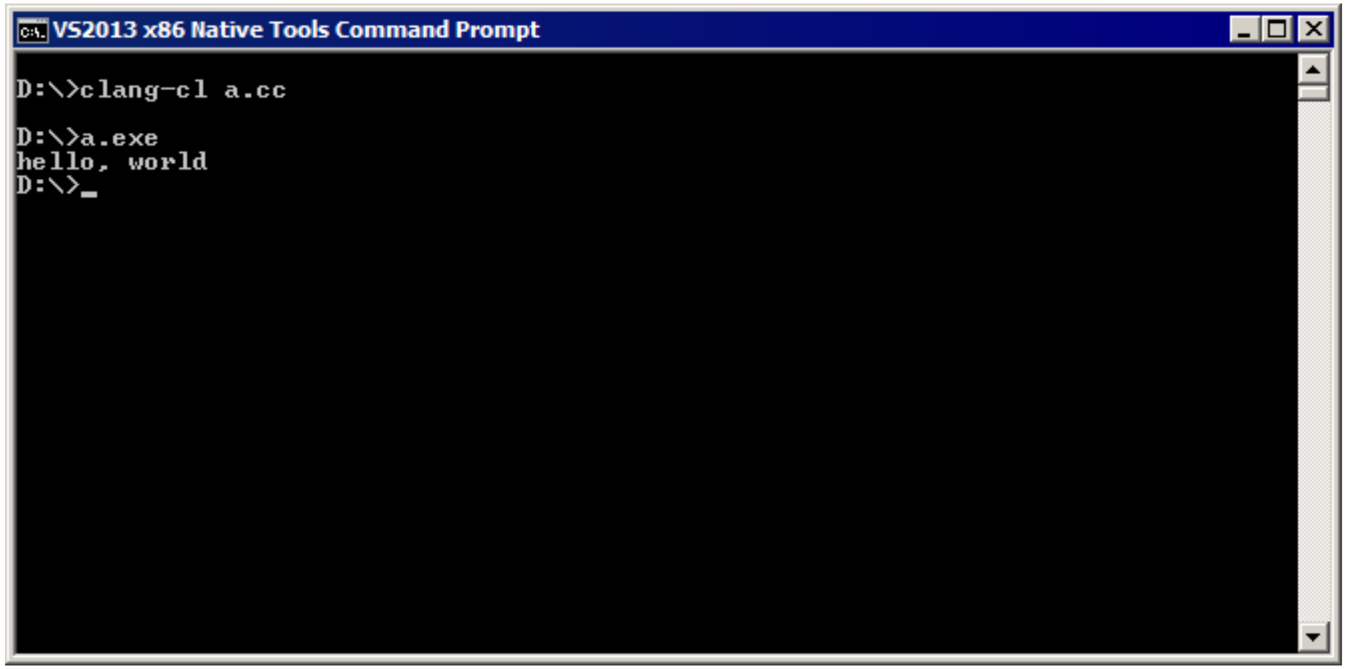
```
D:\>cl a.cc
Microsoft (R) C/C++ Optimizing Compiler Version 18.00.21005.1 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

a.cc
Microsoft (R) Incremental Linker Version 12.00.21005.1
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:a.exe
a.obj

D:\>_
```

This is clang-cl.exe



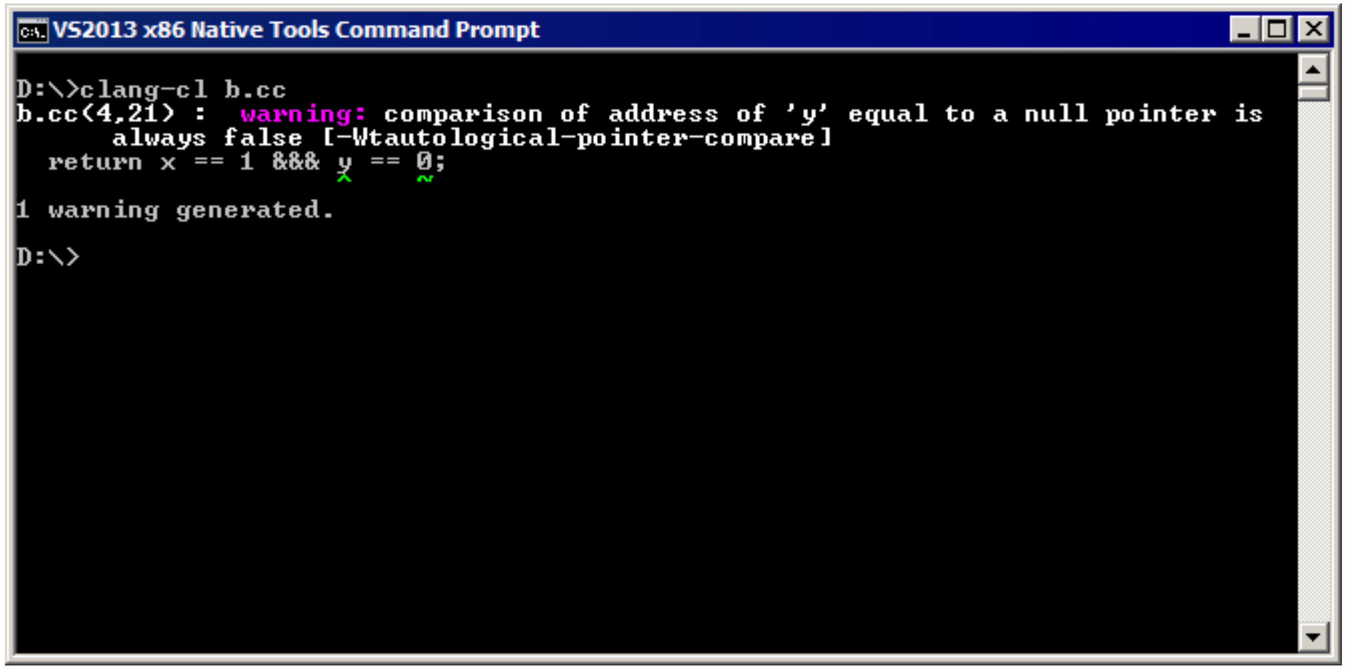
```
D:\>clang-cl a.cc

D:\>_
```

This is clang-cl.exe



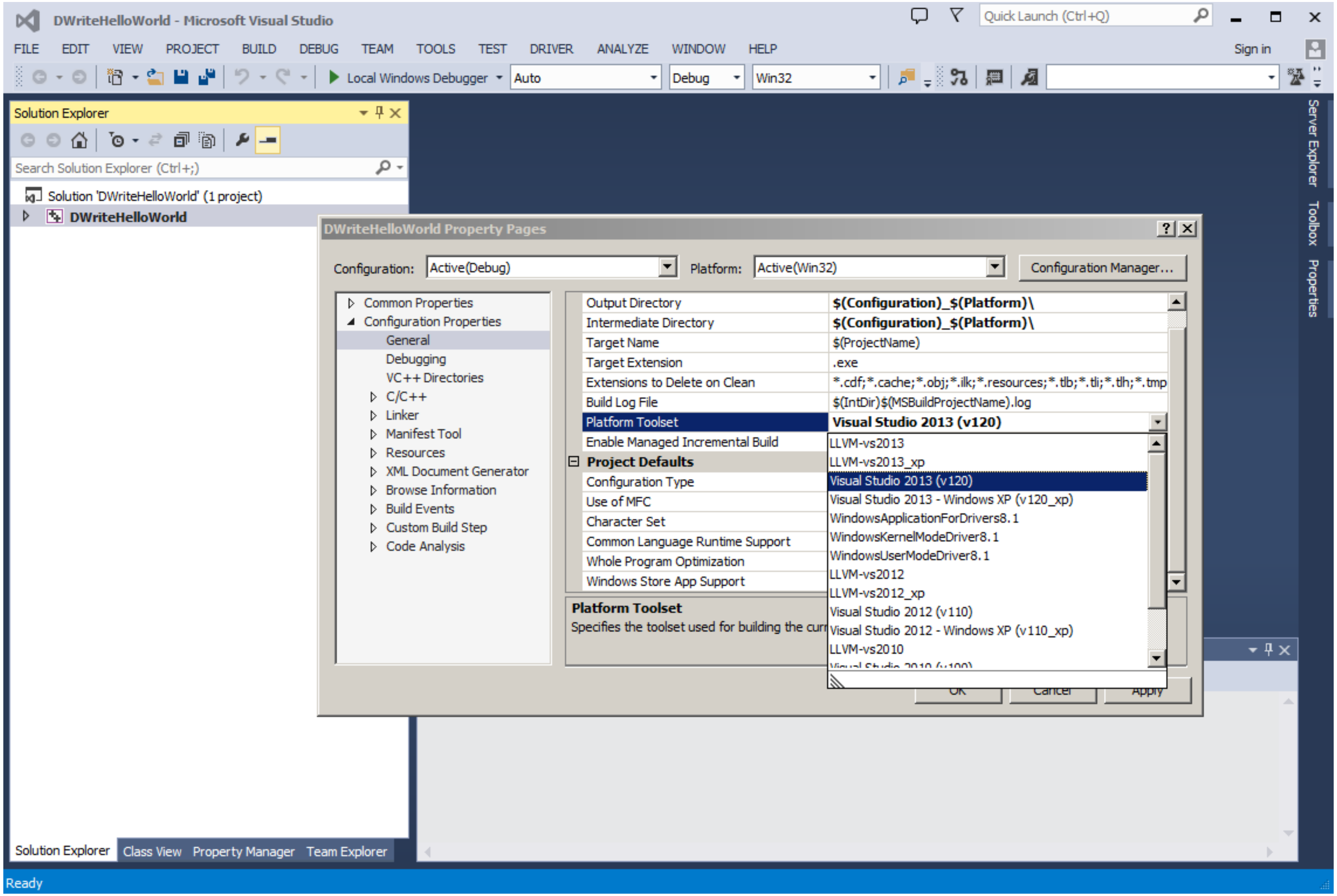
```
D:\>clang-cl a.cc

D:\>a.exe
hello, world
D:\>_
```

This is clang-cl.exe


VS2013 x86 Native Tools Command Prompt

```
D:\>clang-cl b.cc
b.cc(4,21) :  warning: comparison of address of 'y' equal to a null pointer is
      always false [-Wtautological-pointer-compare]
  return x == 1 &&& y == 0;
                    ^    ~

1 warning generated.

D:\>
```

clang-cl in Visual Studio


DWriteHelloWorld Property Pages

Configuration: Active(Debug)   Platform: Active(Win32)

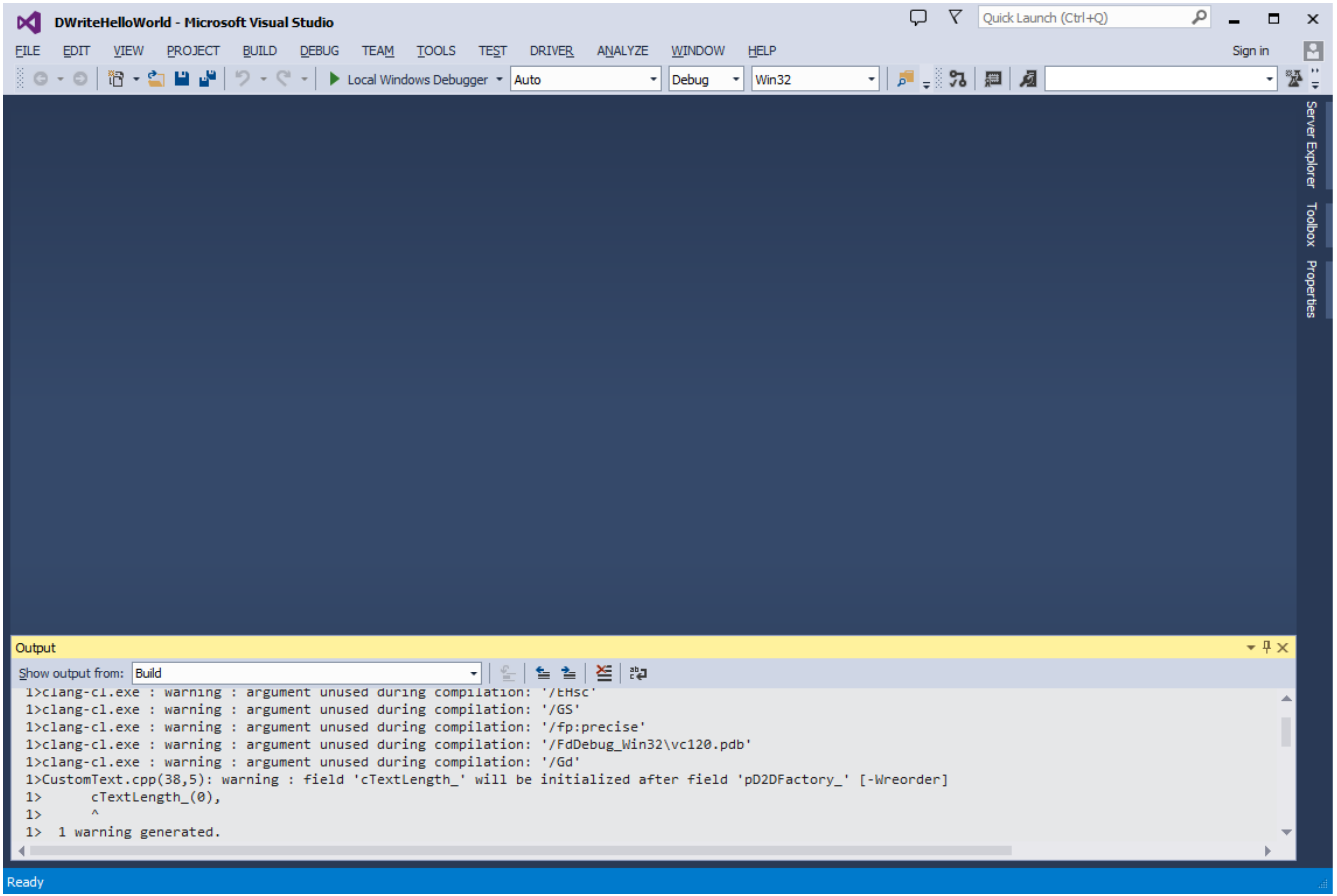
Output Directory	$(Configuration)_$(Platform)\
Intermediate Directory	$(Configuration)_$(Platform)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*.tlh;*.tmp
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	Visual Studio 2013 (v120)

Platform Toolset options:
- LLVM-vs2013
- LLVM-vs2013_xp
- Visual Studio 2013 (v120)
- Visual Studio 2013 - Windows XP (v120_xp)
- WindowsApplicationForDrivers8.1
- WindowsKernelModeDriver8.1
- WindowsUserModeDriver8.1
- LLVM-vs2012
- LLVM-vs2012_xp
- Visual Studio 2012 (v110)
- Visual Studio 2012 - Windows XP (v110_xp)
- LLVM-vs2010

Platform Toolset
Specifies the toolset used for building the curr

DWriteHelloWorld - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST DRIVER ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Solution Explorer
Solution 'DWriteHelloWorld' (1 project)
DWriteHelloWorld
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
Manifest Tool
Resources
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Enable Managed Incremental Build
Project Defaults
Configuration Type
Use of MFC
Character Set
Common Language Runtime Support
Whole Program Optimization
Windows Store App Support
Configuration Manager...
OK Cancel Apply
Solution Explorer Class View Property Manager Team Explorer
Server Explorer Toolbox Properties
Ready

clang-cl in Visual Studio

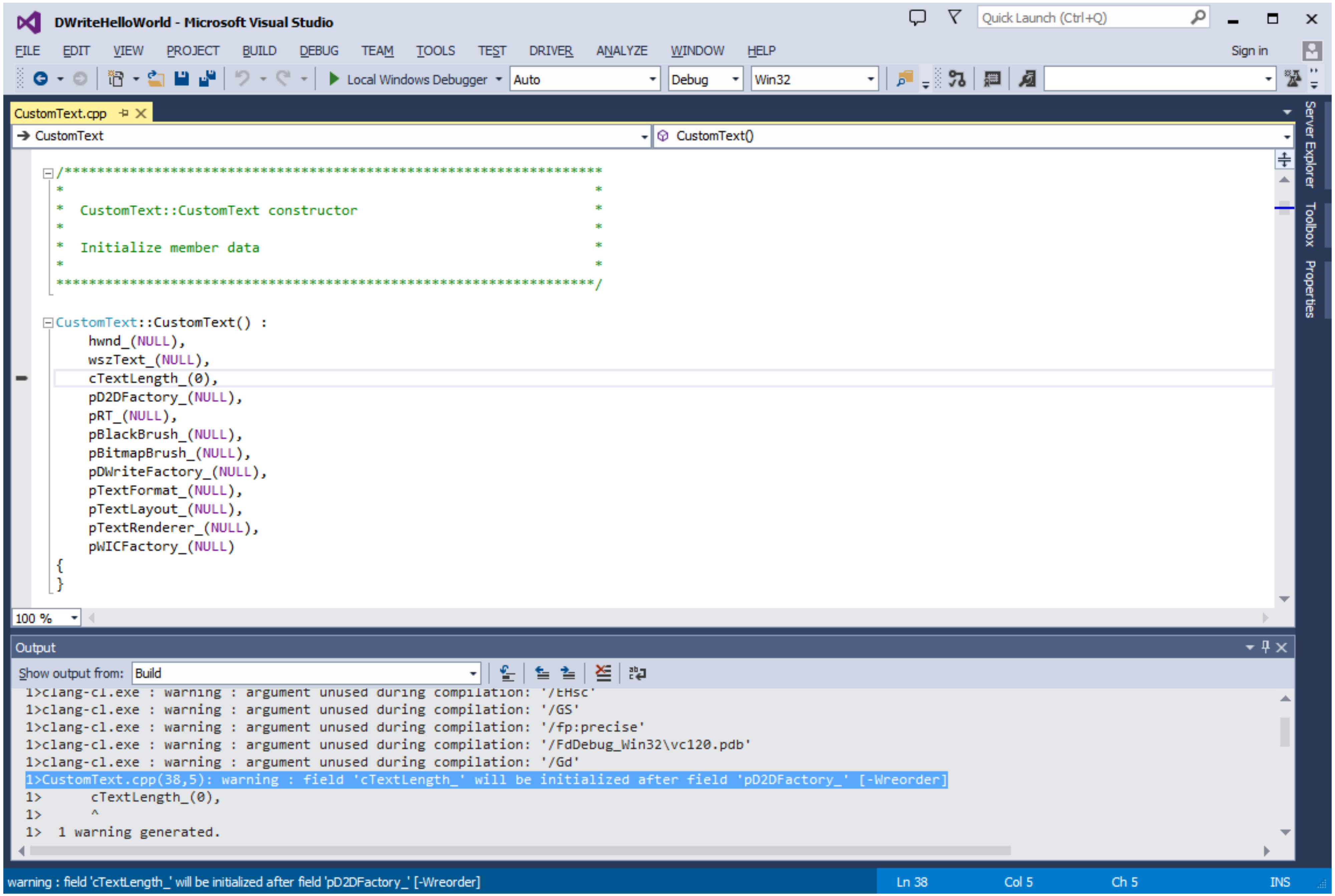


Auto | Debug | Win32

Show output from: Build

```
1>clang-cl.exe : warning : argument unused during compilation: '/EHsc'
1>clang-cl.exe : warning : argument unused during compilation: '/GS'
1>clang-cl.exe : warning : argument unused during compilation: '/fp:precise'
1>clang-cl.exe : warning : argument unused during compilation: '/FdDebug_Win32\vc120.pdb'
1>clang-cl.exe : warning : argument unused during compilation: '/Gd'
1>CustomText.cpp(38,5): warning : field 'cTextLength_' will be initialized after field 'pD2DFactory_' [-Wreorder]
1>      cTextLength_(0),
1>      ^
1>  1 warning generated.
```

clang-cl in Visual Studio

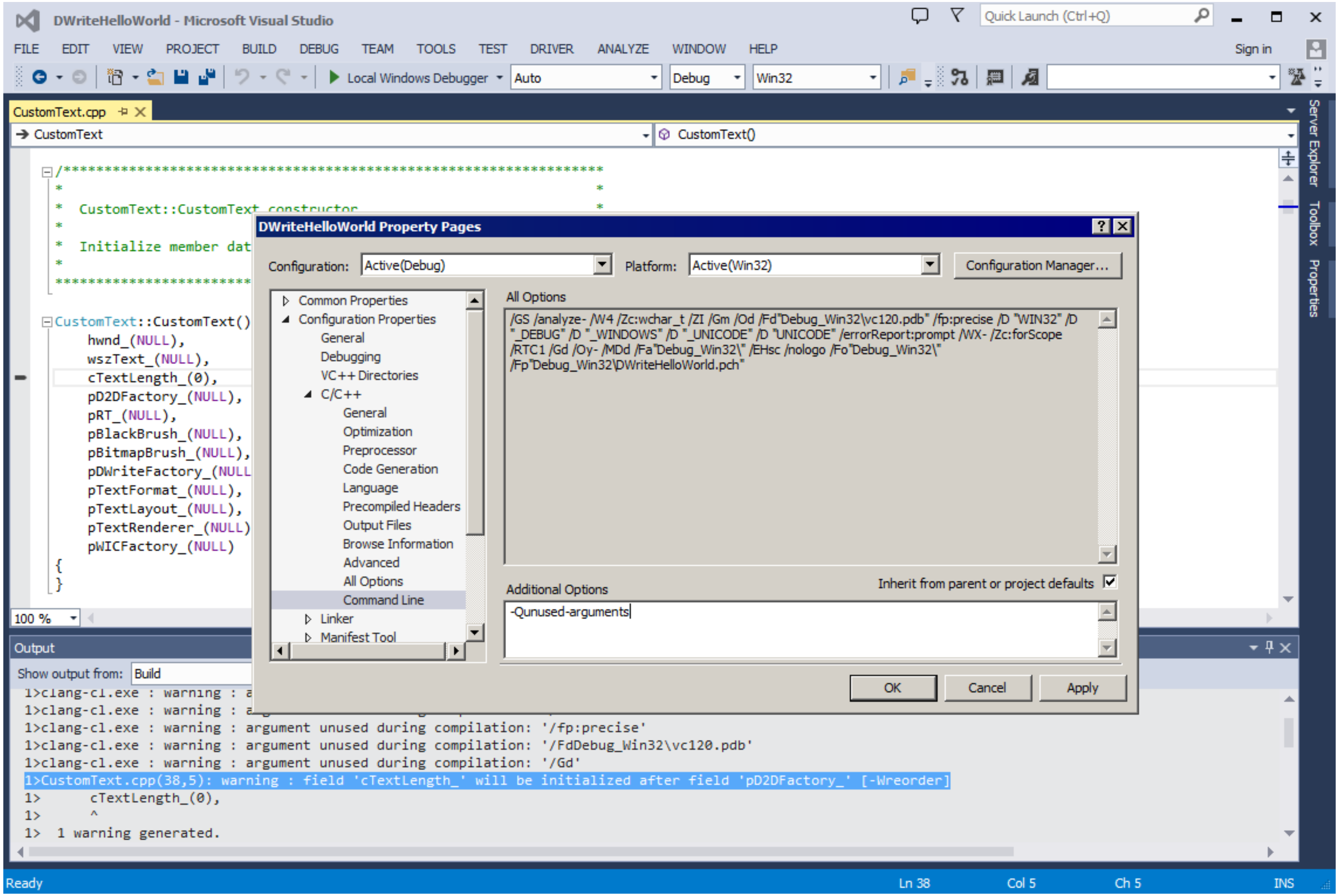


```
/*********************************************************************
*                                                                   *
*  CustomText::CustomText constructor                               *
*                                                                   *
*  Initialize member data                                           *
*                                                                   *
*********************************************************************/

CustomText::CustomText() :
    hwnd_(NULL),
    wszText_(NULL),
    cTextLength_(0),
    pD2DFactory_(NULL),
    pRT_(NULL),
    pBlackBrush_(NULL),
    pBitmapBrush_(NULL),
    pDWriteFactory_(NULL),
    pTextFormat_(NULL),
    pTextLayout_(NULL),
    pTextRenderer_(NULL),
    pWICFactory_(NULL)
{
}
```

```
1>clang-cl.exe : warning : argument unused during compilation: '/EHsc'
1>clang-cl.exe : warning : argument unused during compilation: '/GS'
1>clang-cl.exe : warning : argument unused during compilation: '/fp:precise'
1>clang-cl.exe : warning : argument unused during compilation: '/FdDebug_Win32\vc120.pdb'
1>clang-cl.exe : warning : argument unused during compilation: '/Gd'
1>CustomText.cpp(38,5): warning : field 'cTextLength_' will be initialized after field 'pD2DFactory_' [-Wreorder]
1>        cTextLength_(0),
1>        ^
1>  1 warning generated.
```

warning : field 'cTextLength_' will be initialized after field 'pD2DFactory_' [-Wreorder]

clang-cl in Visual Studio



DWriteHelloWorld - Microsoft Visual Studio

```
/***********************************************************************
 *                                                                     *
 *  CustomText::CustomText constructor                                 *
 *                                                                     *
 *  Initialize member dat
 *
 ***********************

CustomText::CustomText()
    hwnd_(NULL),
    wszText_(NULL),
    cTextLength_(0),
    pD2DFactory_(NULL),
    pRT_(NULL),
    pBlackBrush_(NULL),
    pBitmapBrush_(NULL),
    pDWriteFactory_(NULL
    pTextFormat_(NULL),
    pTextLayout_(NULL),
    pTextRenderer_(NULL)
    pWICFactory_(NULL)
{
}
```

DWriteHelloWorld Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

All Options

```
/GS /analyze- /W4 /Zc:wchar_t /ZI /Gm /Od /Fd"Debug_Win32\vc120.pdb" /fp:precise /D "WIN32" /D
"_DEBUG" /D "_WINDOWS" /D "_UNICODE" /D "UNICODE" /errorReport:prompt /WX- /Zc:forScope
/RTC1 /Gd /Oy- /MDd /Fa"Debug_Win32\" /EHsc /nologo /Fo"Debug_Win32\"
/Fp"Debug_Win32\DWriteHelloWorld.pch"
```

Additional Options

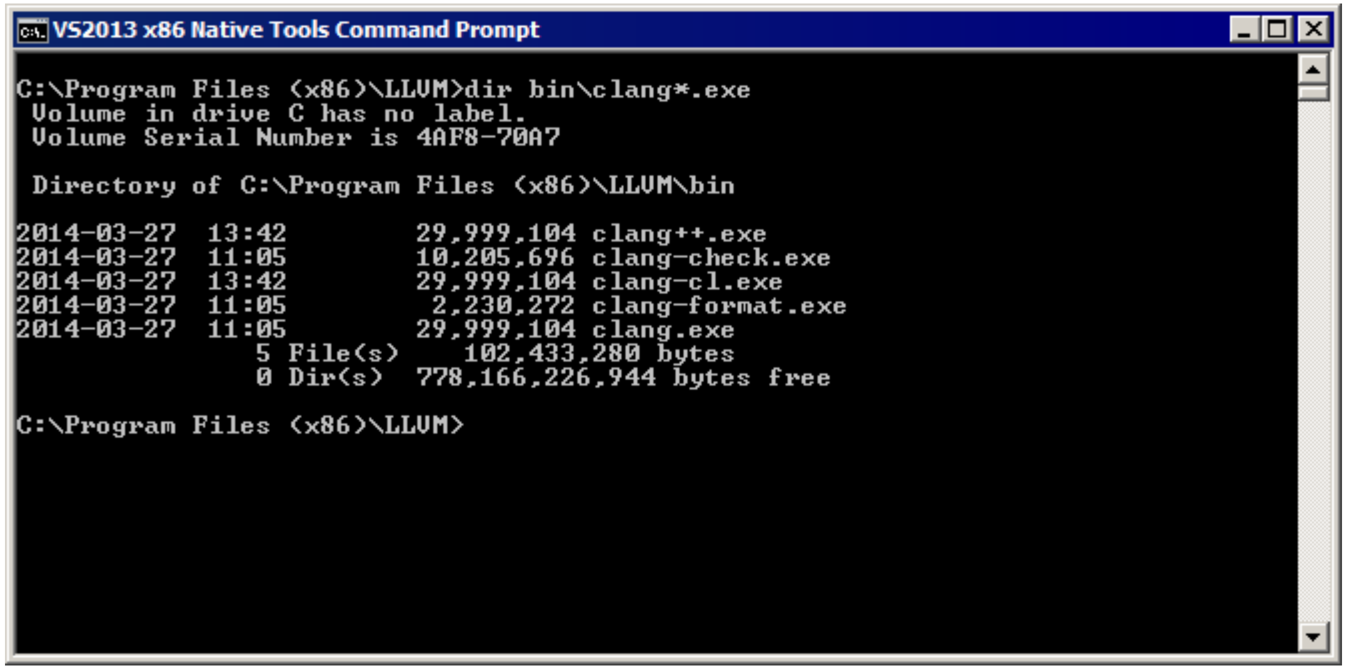
```
-Qunused-arguments
```

Inherit from parent or project defaults ✓

Output — Show output from: Build

```
1>clang-cl.exe : warning : a
1>clang-cl.exe : warning : a
1>clang-cl.exe : warning : argument unused during compilation: '/fp:precise'
1>clang-cl.exe : warning : argument unused during compilation: '/FdDebug_Win32\vc120.pdb'
1>clang-cl.exe : warning : argument unused during compilation: '/Gd'
1>CustomText.cpp(38,5): warning : field 'cTextLength_' will be initialized after field 'pD2DFactory_' [-Wreorder]
1>      cTextLength_(0),
1>      ^
1>  1 warning generated.
```

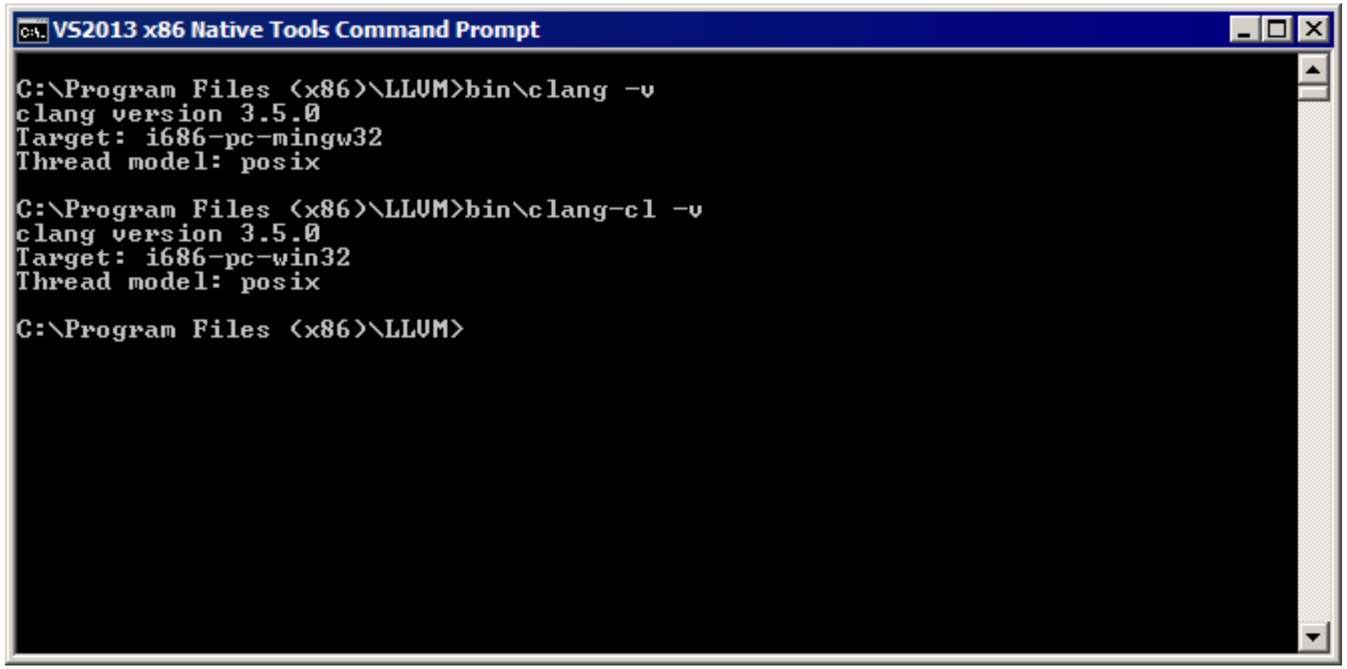
How does clang-cl work?



```
C:\Program Files (x86)\LLVM>dir bin\clang*.exe
 Volume in drive C has no label.
 Volume Serial Number is 4AF8-70A7

 Directory of C:\Program Files (x86)\LLVM\bin

2014-03-27  13:42        29,999,104 clang++.exe
2014-03-27  11:05        10,205,696 clang-check.exe
2014-03-27  13:42        29,999,104 clang-cl.exe
2014-03-27  11:05         2,230,272 clang-format.exe
2014-03-27  11:05        29,999,104 clang.exe
               5 File(s)    102,433,280 bytes
               0 Dir(s)  778,166,226,944 bytes free

C:\Program Files (x86)\LLVM>
```

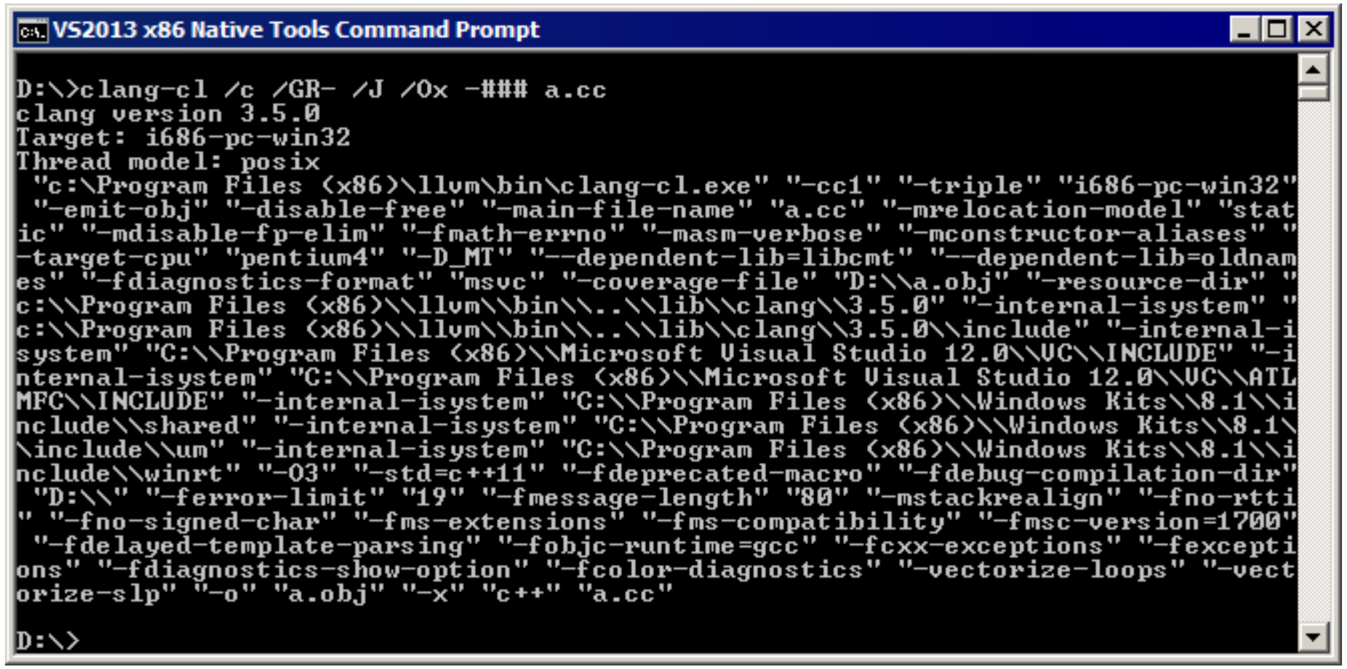
clang-cl.exe == clang.exe --driver-mode=cl

How does clang-cl work?



```
C:\Program Files (x86)\LLVM>bin\clang -v
clang version 3.5.0
Target: i686-pc-mingw32
Thread model: posix

C:\Program Files (x86)\LLVM>bin\clang-cl -v
clang version 3.5.0
Target: i686-pc-win32
Thread model: posix

C:\Program Files (x86)\LLVM>
```

How does clang-cl work?



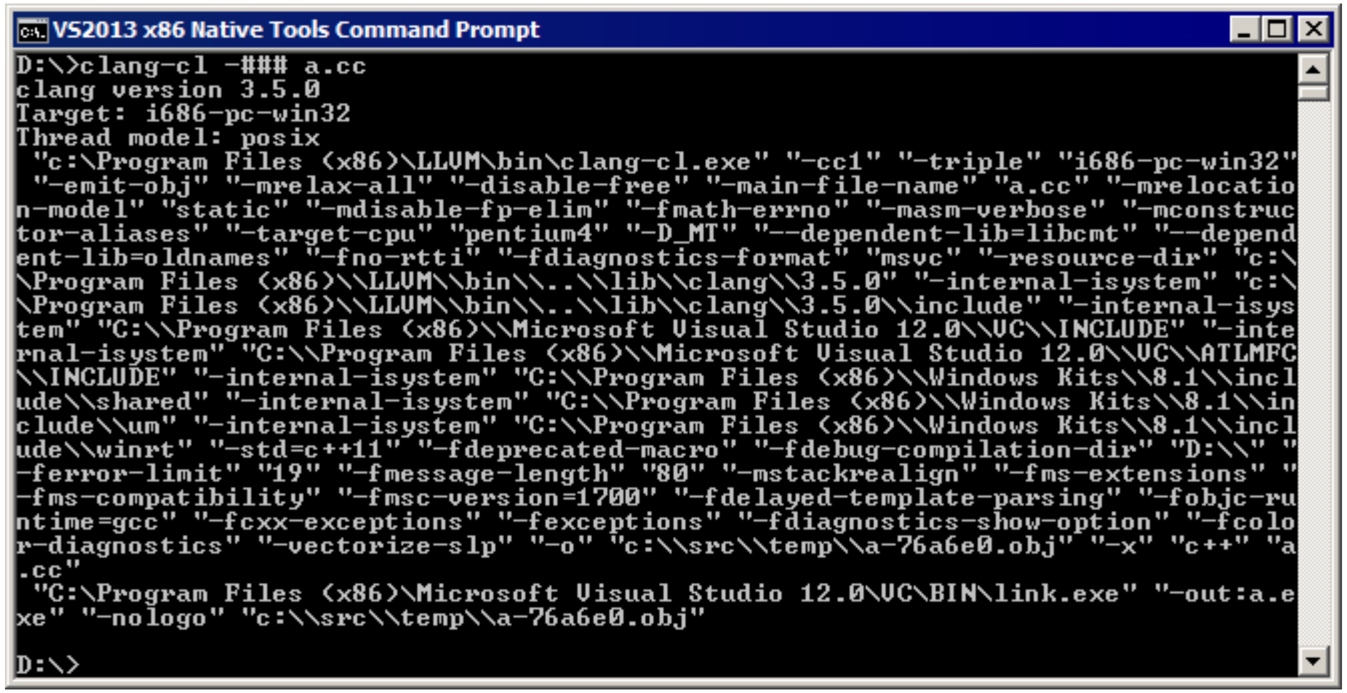
VS2013 x86 Native Tools Command Prompt

```
D:\>clang-cl /c /GR- /J /Ox -### a.cc
clang version 3.5.0
Target: i686-pc-win32
Thread model: posix
 "c:\Program Files (x86)\llvm\bin\clang-cl.exe" "-cc1" "-triple" "i686-pc-win32"
 "-emit-obj" "-disable-free" "-main-file-name" "a.cc" "-mrelocation-model" "stat
ic" "-mdisable-fp-elim" "-fmath-errno" "-masm-verbose" "-mconstructor-aliases" "
-target-cpu" "pentium4" "-D_MT" "--dependent-lib=libcmt" "--dependent-lib=oldnam
es" "-fdiagnostics-format" "msvc" "-coverage-file" "D:\\a.obj" "-resource-dir" "
c:\\Program Files (x86)\\llvm\\bin\\..\\lib\\clang\\3.5.0" "-internal-isystem" "
c:\\Program Files (x86)\\llvm\\bin\\..\\lib\\clang\\3.5.0\\include" "-internal-i
system" "C:\\Program Files (x86)\\Microsoft Visual Studio 12.0\\VC\\INCLUDE" "-i
nternal-isystem" "C:\\Program Files (x86)\\Microsoft Visual Studio 12.0\\VC\\ATL
MFC\\INCLUDE" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\\i
nclude\\shared" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\
\include\\um" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\\i
nclude\\winrt" "-O3" "-std=c++11" "-fdeprecated-macro" "-fdebug-compilation-dir"
 "D:\\" "-ferror-limit" "19" "-fmessage-length" "80" "-mstackrealign" "-fno-rtti
" "-fno-signed-char" "-fms-extensions" "-fms-compatibility" "-fmsc-version=1700"
 "-fdelayed-template-parsing" "-fobjc-runtime=gcc" "-fcxx-exceptions" "-fexcepti
ons" "-fdiagnostics-show-option" "-fcolor-diagnostics" "-vectorize-loops" "-vect
orize-slp" "-o" "a.obj" "-x" "c++" "a.cc"

D:\>
```

How does clang-cl work?

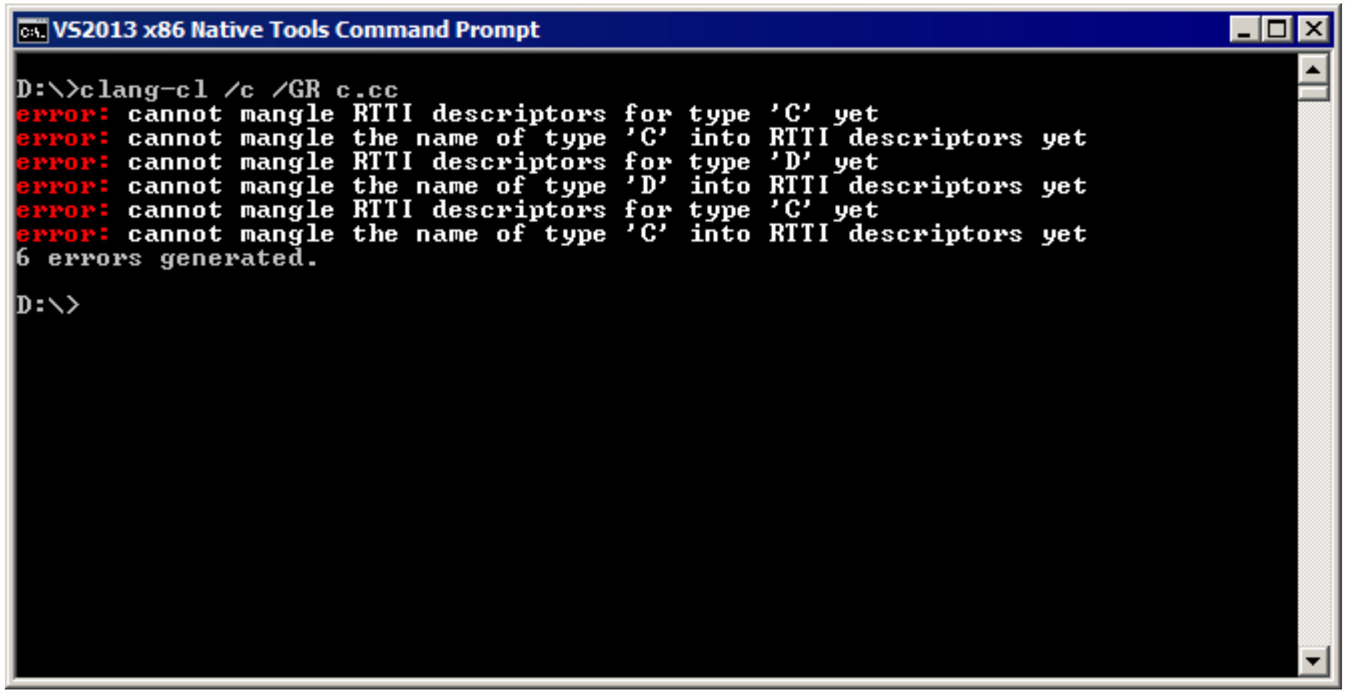
```
def _SLASH_C : CLFlag<"C">, HelpText<"Don't discard comments when preprocessing"
>,
  Alias<C>;
def _SLASH_c : CLFlag<"c">, HelpText<"Compile only">, Alias<c>;
def _SLASH_D : CLJoinedOrSeparate<"D">, HelpText<"Define macro">,
  MetaVarName<"<macro[=value]>">, Alias<D>;
def _SLASH_E : CLFlag<"E">, HelpText<"Preprocess to stdout">, Alias<E>;
def _SLASH_GR : CLFlag<"GR">, HelpText<"Enable RTTI">, Alias<frtti>;
def _SLASH_GR_ : CLFlag<"GR-">, HelpText<"Disable RTTI">, Alias<fno_rtti>;
def _SLASH_GF_ : CLFlag<"GF-">, HelpText<"Disable string pooling">,
  Alias<fwritable_strings>;
def _SLASH_Gy : CLFlag<"Gy">, HelpText<"Put each function in its own section">,
  Alias<ffunction_sections>;
def _SLASH_Gy_ : CLFlag<"Gy-">, HelpText<"Don't put each function in its own
section">,
  Alias<fno_function_sections>;
```

How does clang-cl work?


VS2013 x86 Native Tools Command Prompt

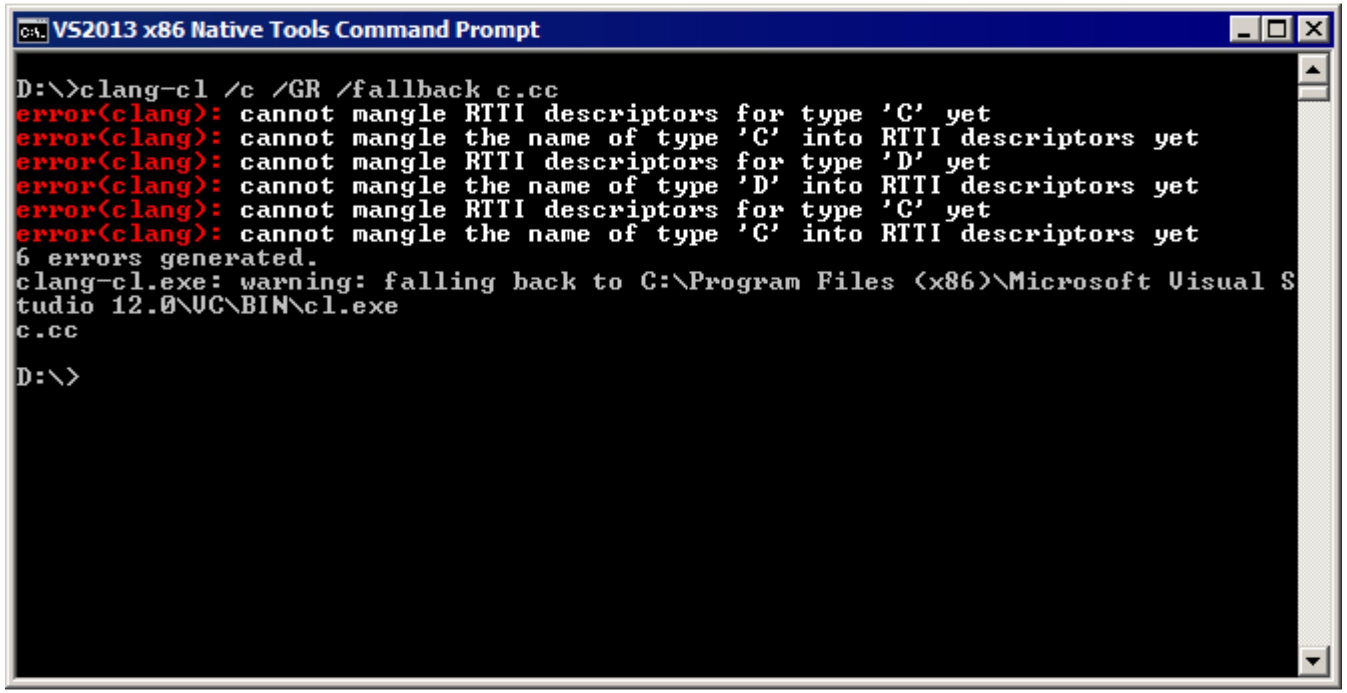
```
D:\>clang-cl -### a.cc
clang version 3.5.0
Target: i686-pc-win32
Thread model: posix
 "c:\Program Files (x86)\LLVM\bin\clang-cl.exe" "-cc1" "-triple" "i686-pc-win32"
 "-emit-obj" "-mrelax-all" "-disable-free" "-main-file-name" "a.cc" "-mrelocatio
n-model" "static" "-mdisable-fp-elim" "-fmath-errno" "-masm-verbose" "-mconstruc
tor-aliases" "-target-cpu" "pentium4" "-D_MT" "--dependent-lib=libcmt" "--depend
ent-lib=oldnames" "-fno-rtti" "-fdiagnostics-format" "msvc" "-resource-dir" "c:\
\Program Files (x86)\\LLVM\\bin\\..\\lib\\clang\\3.5.0" "-internal-isystem" "c:\
\Program Files (x86)\\LLVM\\bin\\..\\lib\\clang\\3.5.0\\include" "-internal-isys
tem" "C:\\Program Files (x86)\\Microsoft Visual Studio 12.0\\VC\\INCLUDE" "-inte
rnal-isystem" "C:\\Program Files (x86)\\Microsoft Visual Studio 12.0\\VC\\ATLMFC
\\INCLUDE" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\\incl
ude\\shared" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\\in
clude\\um" "-internal-isystem" "C:\\Program Files (x86)\\Windows Kits\\8.1\\incl
ude\\winrt" "-std=c++11" "-fdeprecated-macro" "-fdebug-compilation-dir" "D:\\" "
-ferror-limit" "19" "-fmessage-length" "80" "-mstackrealign" "-fms-extensions" "
-fms-compatibility" "-fmsc-version=1700" "-fdelayed-template-parsing" "-fobjc-ru
ntime=gcc" "-fcxx-exceptions" "-fexceptions" "-fdiagnostics-show-option" "-fcolo
r-diagnostics" "-vectorize-slp" "-o" "c:\\src\\temp\\a-76a6e0.obj" "-x" "c++" "a
.cc"
 "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\BIN\link.exe" "-out:a.e
xe" "-nologo" "c:\\src\\temp\\a-76a6e0.obj"

D:\>
```

How does clang-cl work?



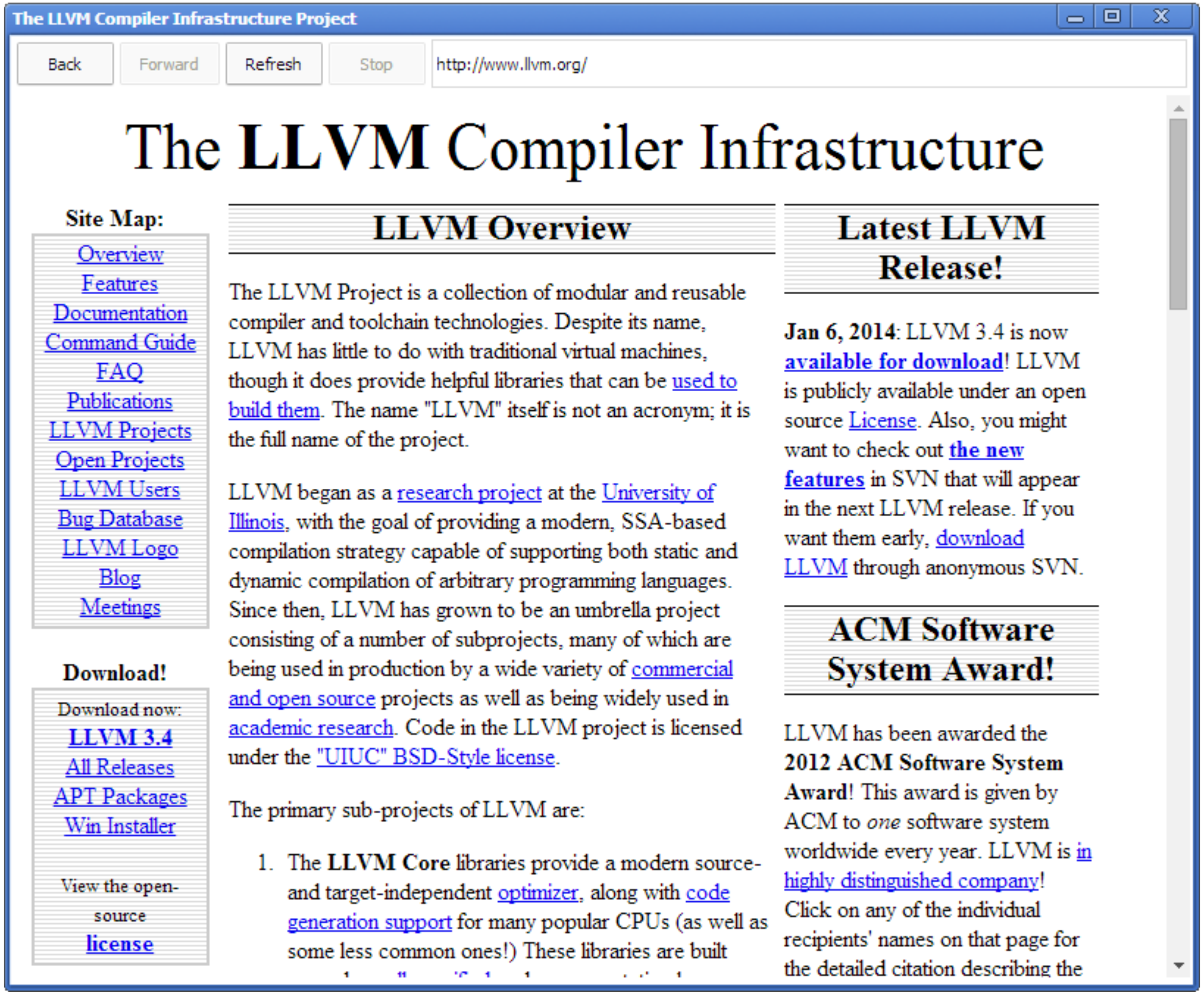
```
D:\>clang-cl /c /GR c.cc
error: cannot mangle RTTI descriptors for type 'C' yet
error: cannot mangle the name of type 'C' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'D' yet
error: cannot mangle the name of type 'D' into RTTI descriptors yet
error: cannot mangle RTTI descriptors for type 'C' yet
error: cannot mangle the name of type 'C' into RTTI descriptors yet
6 errors generated.

D:\>
```

How does clang-cl work?



```
D:\>clang-cl /c /GR /fallback c.cc
error(clang): cannot mangle RTTI descriptors for type 'C' yet
error(clang): cannot mangle the name of type 'C' into RTTI descriptors yet
error(clang): cannot mangle RTTI descriptors for type 'D' yet
error(clang): cannot mangle the name of type 'D' into RTTI descriptors yet
error(clang): cannot mangle RTTI descriptors for type 'C' yet
error(clang): cannot mangle the name of type 'C' into RTTI descriptors yet
6 errors generated.
clang-cl.exe: warning: falling back to C:\Program Files (x86)\Microsoft Visual S
tudio 12.0\VC\BIN\cl.exe
c.cc

D:\>
```

Chromium's content_shell built with clang-cl



The LLVM Compiler Infrastructure

Site Map:

Overview
Features
Documentation
Command Guide
FAQ
Publications
LLVM Projects
Open Projects
LLVM Users
Bug Database
LLVM Logo
Blog
Meetings

Download!

Download now:
LLVM 3.4
All Releases
APT Packages
Win Installer

View the open-source **license**

LLVM Overview

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Despite its name, LLVM has little to do with traditional virtual machines, though it does provide helpful libraries that can be used to build them. The name "LLVM" itself is not an acronym; it is the full name of the project.

LLVM began as a research project at the University of Illinois, with the goal of providing a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation of arbitrary programming languages. Since then, LLVM has grown to be an umbrella project consisting of a number of subprojects, many of which are being used in production by a wide variety of commercial and open source projects as well as being widely used in academic research. Code in the LLVM project is licensed under the "UIUC" BSD-Style license.

The primary sub-projects of LLVM are:

1. The **LLVM Core** libraries provide a modern source- and target-independent optimizer, along with code generation support for many popular CPUs (as well as some less common ones!) These libraries are built

Latest LLVM Release!

Jan 6, 2014: LLVM 3.4 is now **available for download**! LLVM is publicly available under an open source License. Also, you might want to check out **the new features** in SVN that will appear in the next LLVM release. If you want them early, download LLVM through anonymous SVN.

ACM Software System Award!

LLVM has been awarded the **2012 ACM Software System Award**! This award is given by ACM to *one* software system worldwide every year. LLVM is in highly distinguished company! Click on any of the individual recipients' names on that page for the detailed citation describing the

- The driver provides convenience and compatibility
- clang-cl is a cl.exe compatible driver mode for clang
- It understands the environment, the flags, and the tools
- Integrates with Visual Studio
- /fallback allows bring-up of large projects.

hans@chromium.org