**作业 4: 服务质量保障**

**1) 代码静态分析**

**1.1 下载 HBase 和 Cassandra 项目的最新版本源代码，使用 WALA 工具统计含有 Replica/Replication 关键字的类个数，并分析这些类之间的调用依赖；**

**1.2 选择自己之前作业中开发的某个服务，使用 LLVM 进行符号执行分析，并进行自动化的用例生成和测试；**

**2) 代码动态分析**

**2.1 下载 Map Reduce 项目的最新版本源代码，编译运行自带的 word count 例子。基于Java agent机制和javassist工具生成word count例子运行过程中Map Reduce产生的trace，包括进入和退出每个方法的时间戳、线程号等，分析其中可能的性能瓶颈。**

1. 代码静态分析

以前的作业都是用java写的，LLVM大部分处理都是C/C++代码，这里的话我选择先熟悉下LLVM的基本使用。

PS：这里发现一个网址可以在线运行LLVM进行代码分析。（方便老师教学使用）

<http://ellcc.org/demo/index.cgi>

提交作业前发现一个叫做Jlang的东西

An LLVM backend for the Polyglot compiler

<https://polyglot-compiler.github.io/JLang/index.html>

首先看下clang+LLVM安装要求（这里尝试从头编译出可运行的程序而不是直接用人家编译好）：



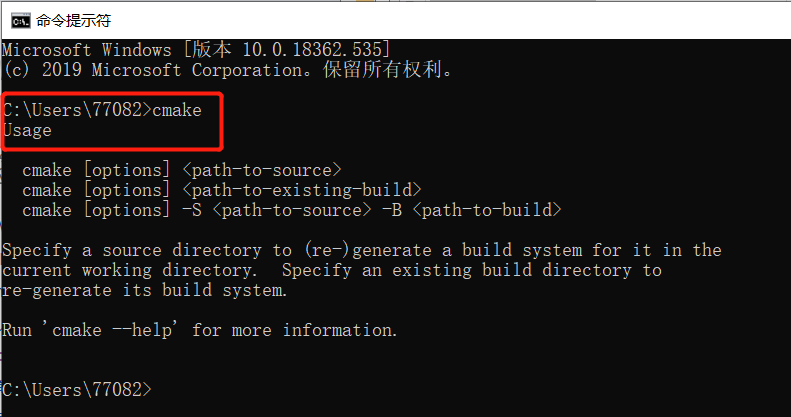
1.1 首先安装下Cmake

首先下载CMake官网下载地址：<https://cmake.org/download/>

里面好多版本，根据自己需要版本进行下载。

【注意】选择好自己电脑是什么系统，以及是32位还是64位。（P.S.x86指的是32位系统；x64指的是64位系统）

安装过程参考网址：<https://blog.csdn.net/u011231598/article/details/80338941>



1.2 安装Git

<https://git-scm.com/download>

1.3 安装Python

<https://www.python.org/download/>

1.4 安装GnuWin32 tools

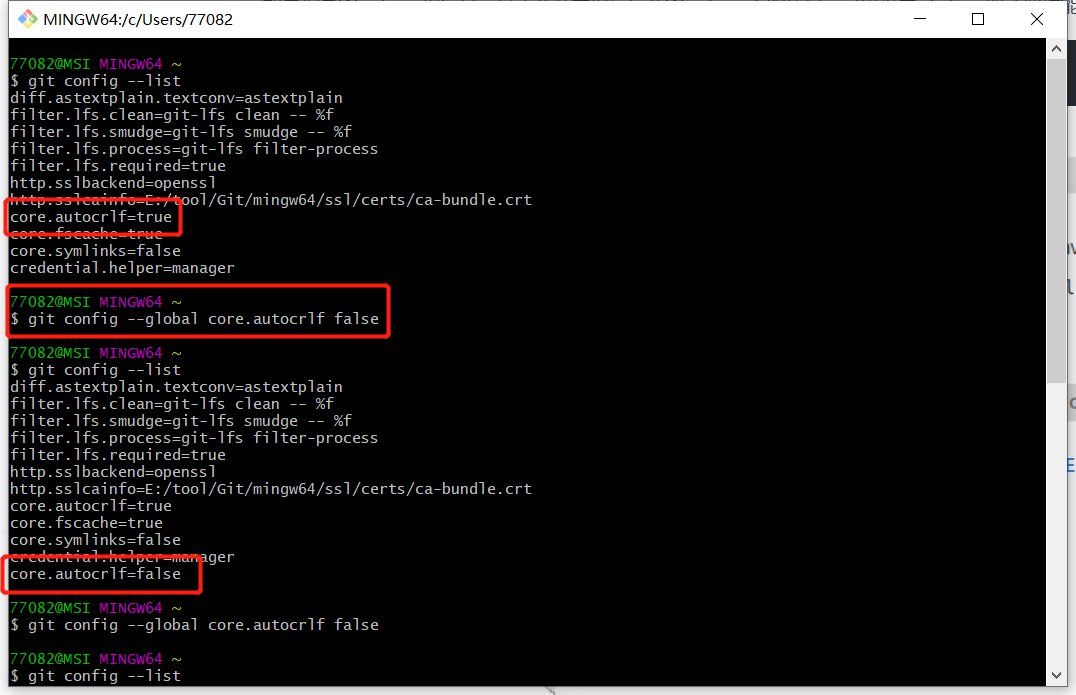
<http://getgnuwin32.sourceforge.net/>

1.5 Visual Studio 2017 or later

电脑上已经有Visual Studio 2015的版本，vs2015-up1以上版本是支持clang的这里尝试配置看看。

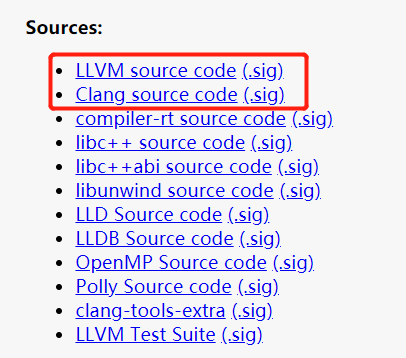
1.6首先修改git core.autocrlf配置说明

参考<https://www.cnblogs.com/youpeng/p/11243871.html>



1.7 下载LLVM和CLANG的源代码（这里选择6.0.0的版本原则需要5.0.1版本以上）

<http://releases.llvm.org/download.html#6.0.0>



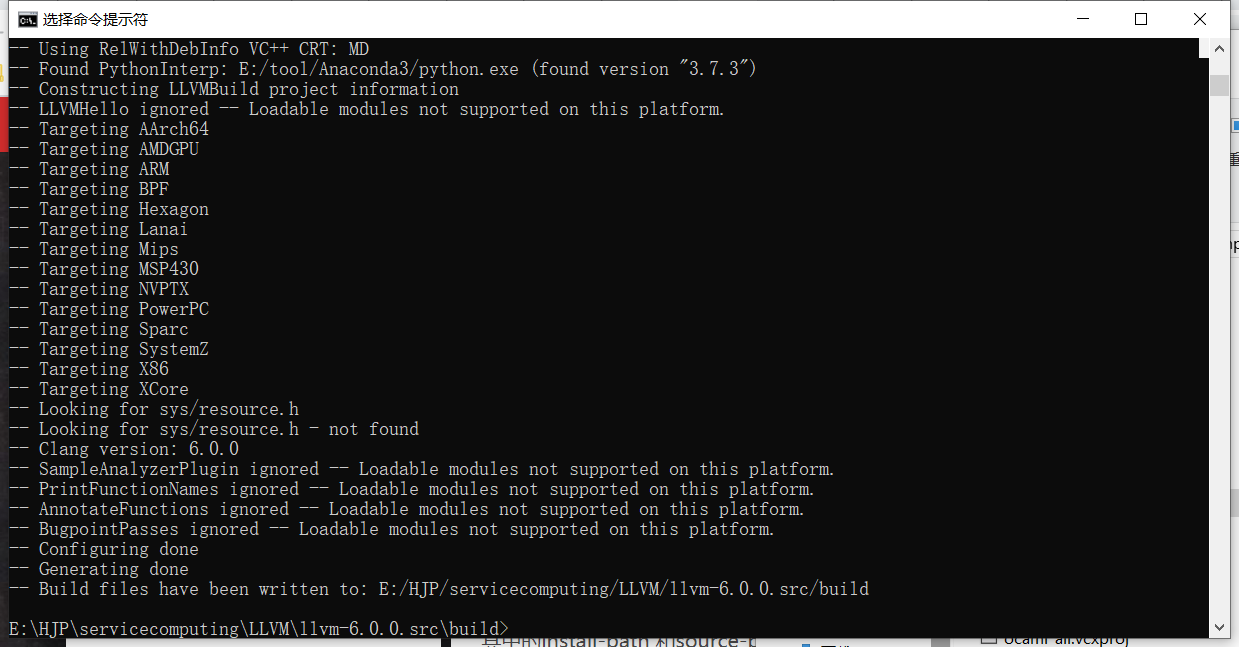
1.8 下载好代码以后解压，解压后，在llvm文件夹下面的../tools文件夹下建立clang文件夹，把clang解压缩后的文件拷贝进去

在llvm根目录下面建立一个名为build空文件夹用来放后续编译内容

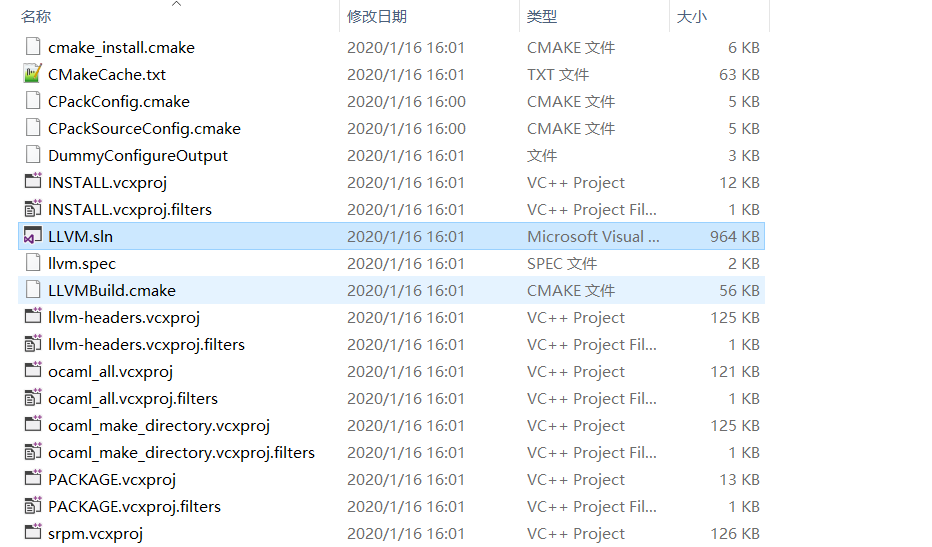
1.9 打开cmd，转到build文件夹下，执行

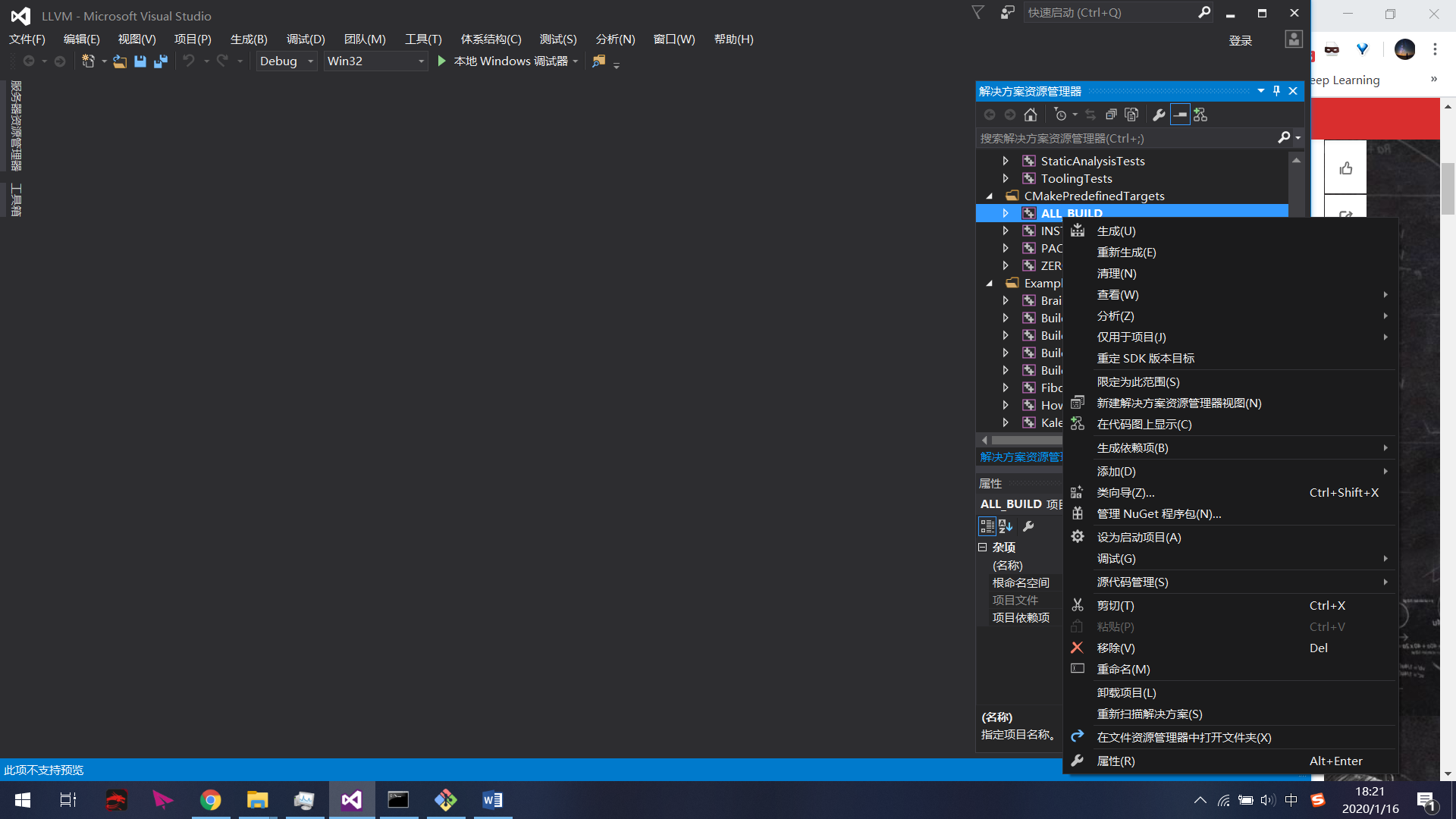
cmake -Thost=x64 -G "Visual Studio 14" E:\HJP\servicecomputing\LLVM\llvm-6.0.0.src

后面的路径为下载的llvm源码根目录，即cmakelist.txt所在的文件夹



如果成功编译，则可以在build文件夹下看到生成的llvm.sln工程，双击用VS打开，下拉找到ALL\_BUILD工程，选中点击 '生成' 里面的生成解决方案进行build，这步耗时较长，若这步执行成功，可以看到在build\Debug\bin中包含了clang.exe等多个可执行文件，将该路径加入到环境变量中，就安装完成了。



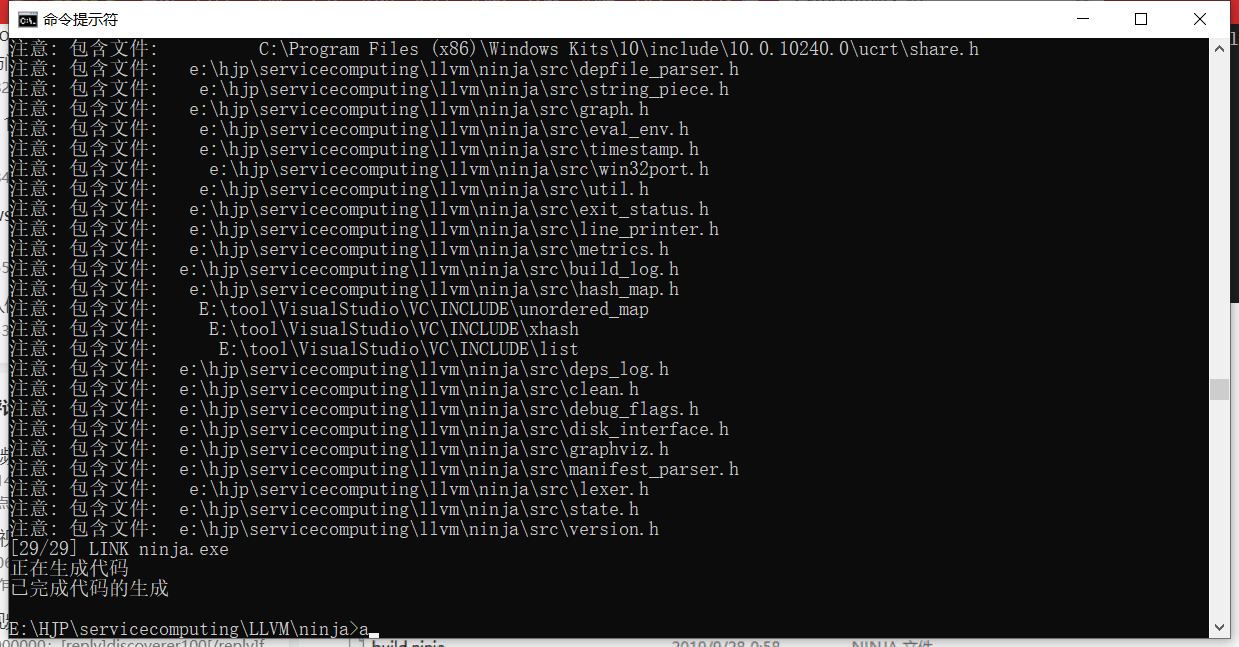


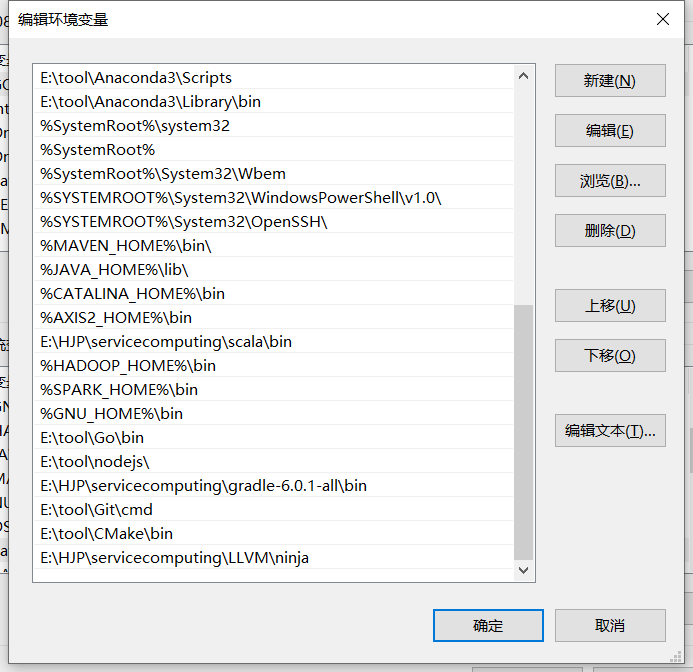
1.10 VS2015编译时间太长了放弃了，太恶心了，还报了一堆错误。查阅资料发现有人说用VS编译就是头铁，编译时间慢还特别多问题。所以编译换成ninja来编译看看clang+llvm。

1.11 安装ninja

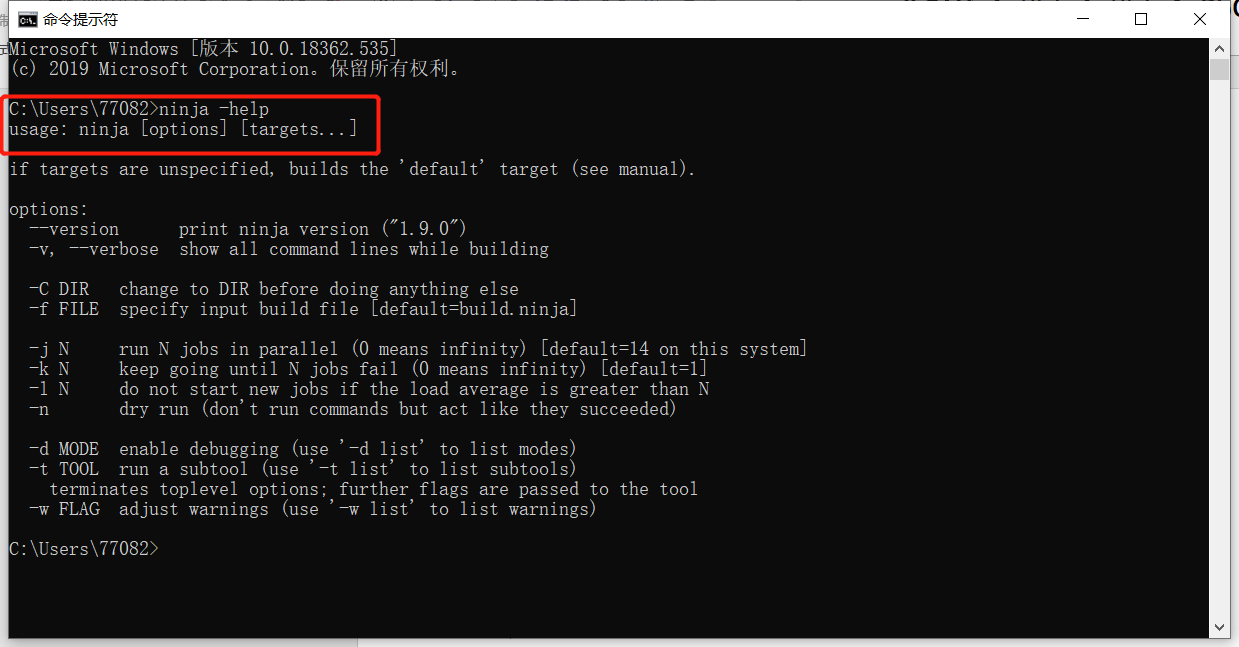
参考<https://blog.csdn.net/discoverer100/article/details/101574807>

坑：window就不用了编译了直接下载编译好的ninja.exe就行。真的服了，我还按照教程把文件编译出来，没想到人家github上面已经提供了。

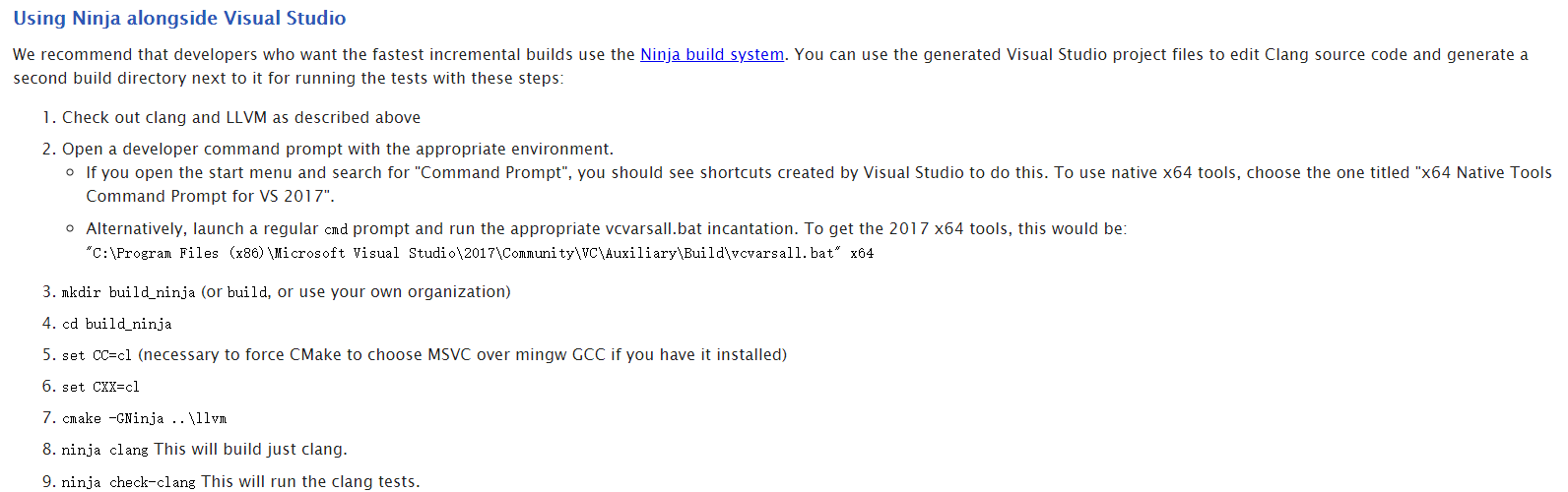




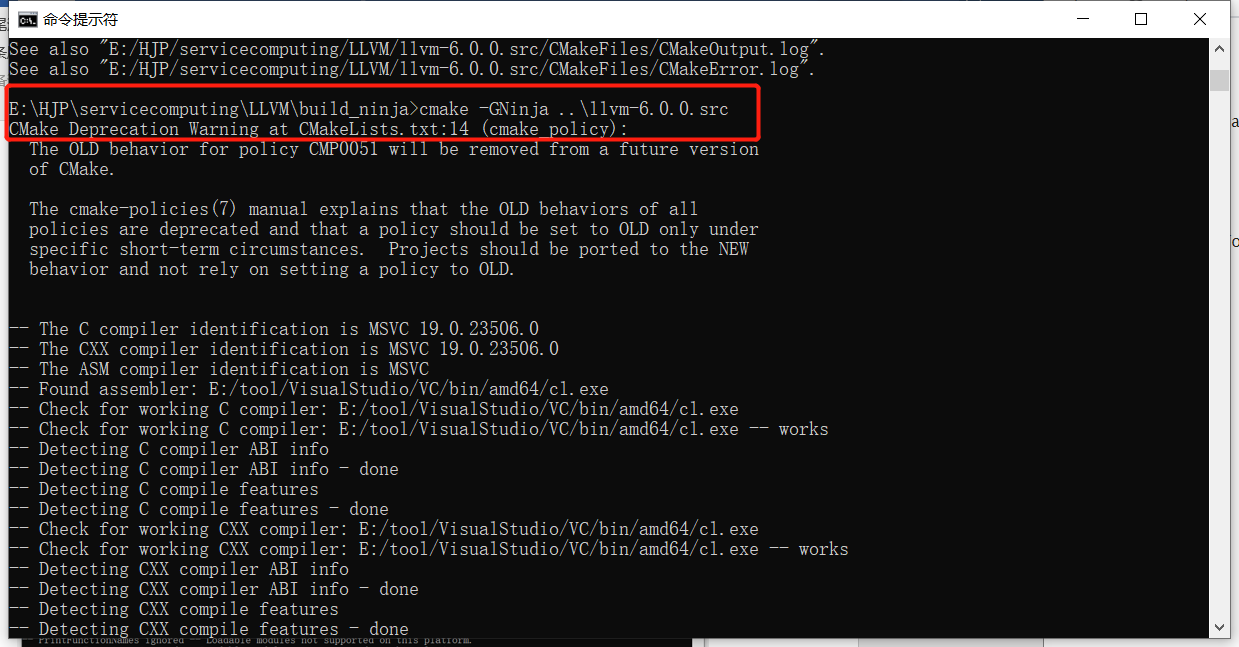
配置好环境变量。

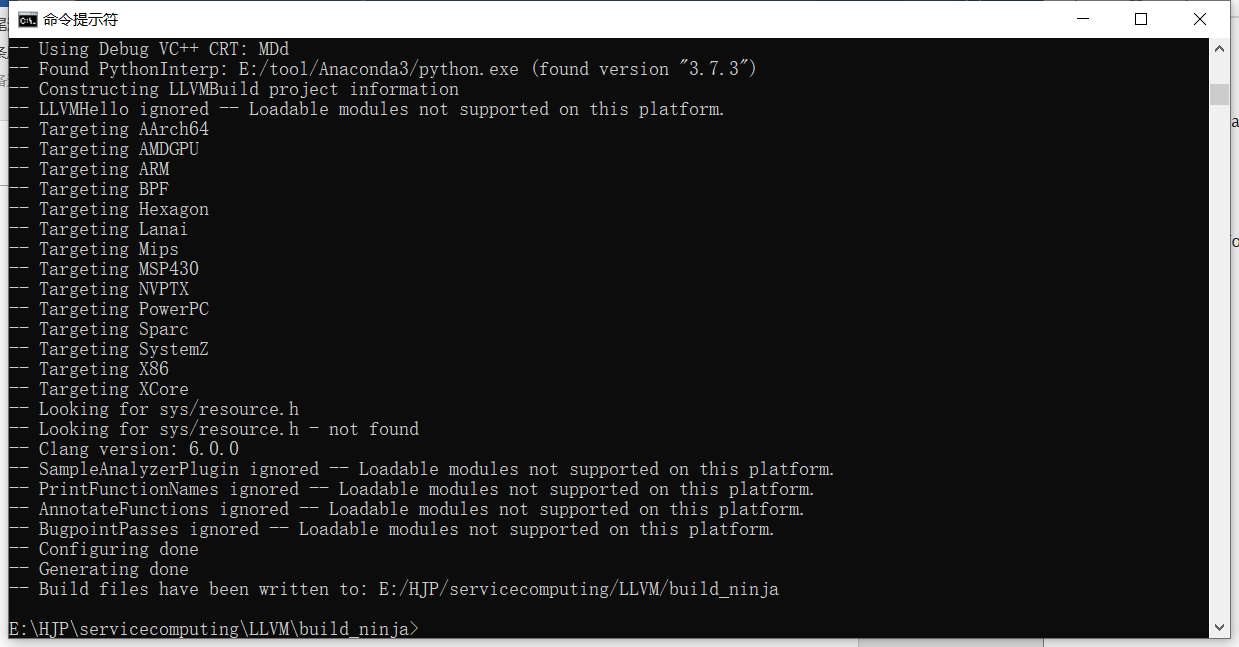


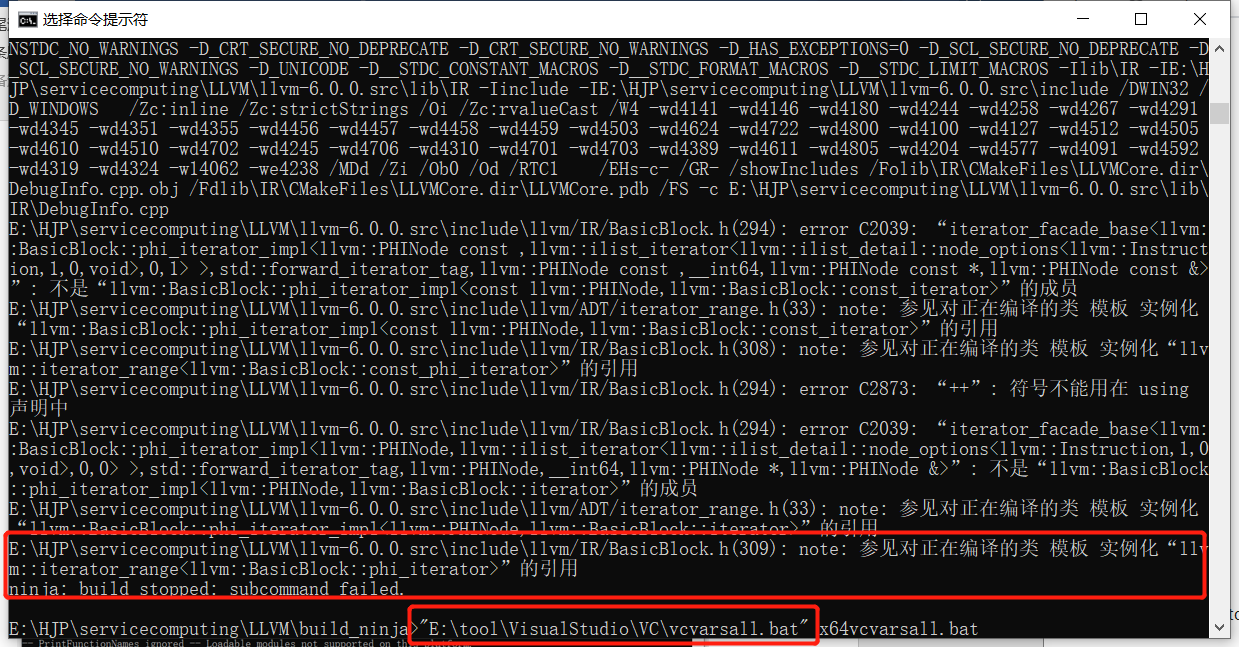
1.12 按照以下命令运行



输出：



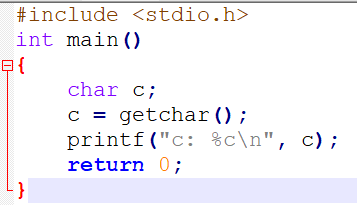


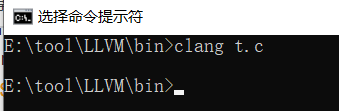


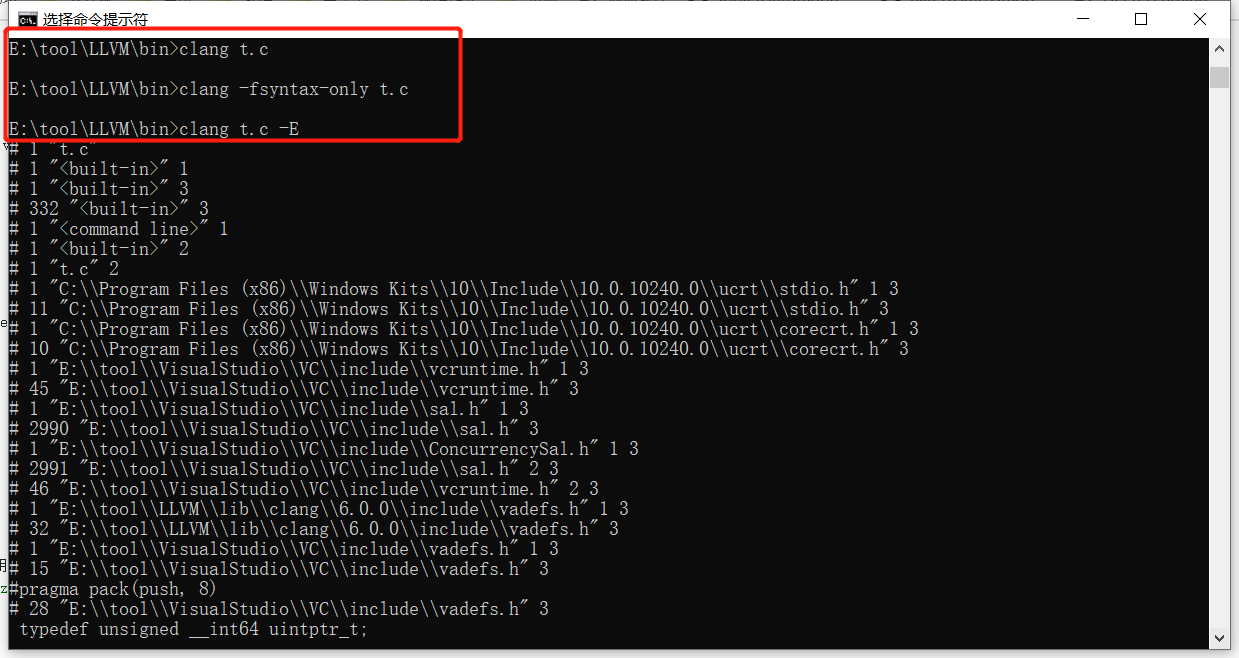
安装失败，推测是用VS2015编译用不了。

1.13成功以后我们直接使用clang去编译文件

代码：

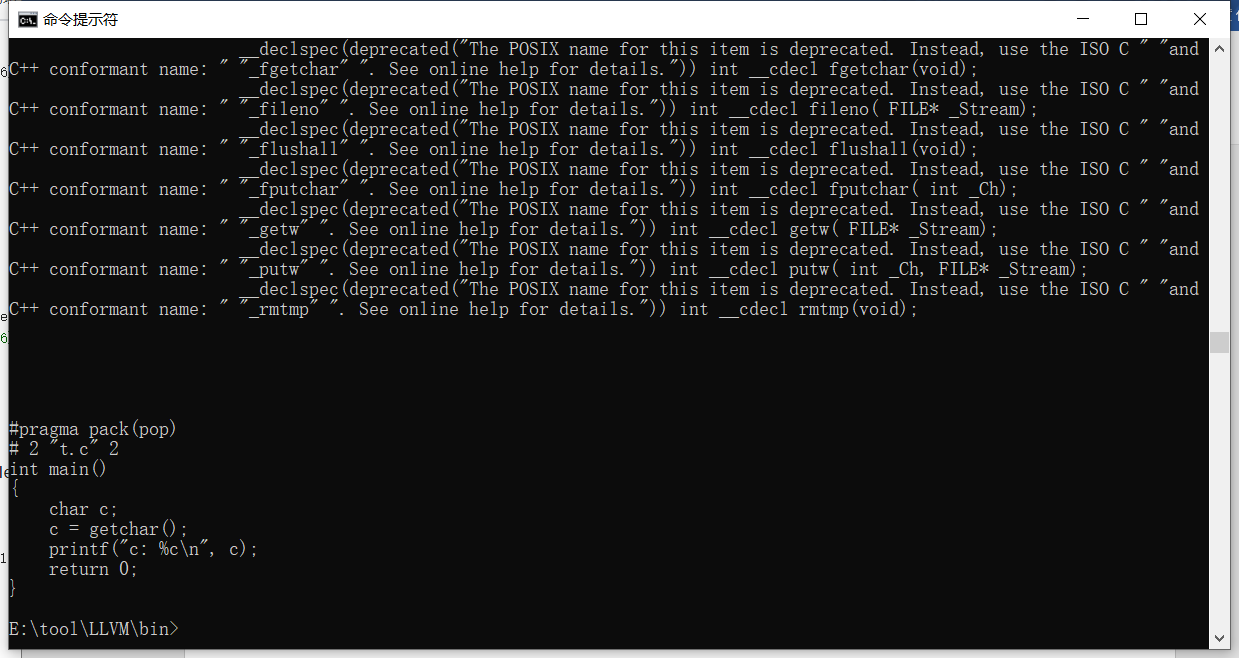




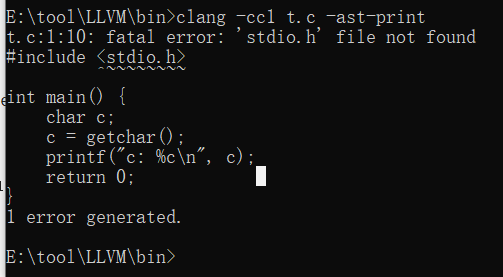


第一个是编译，第二个是类型检查，第三个是预处理

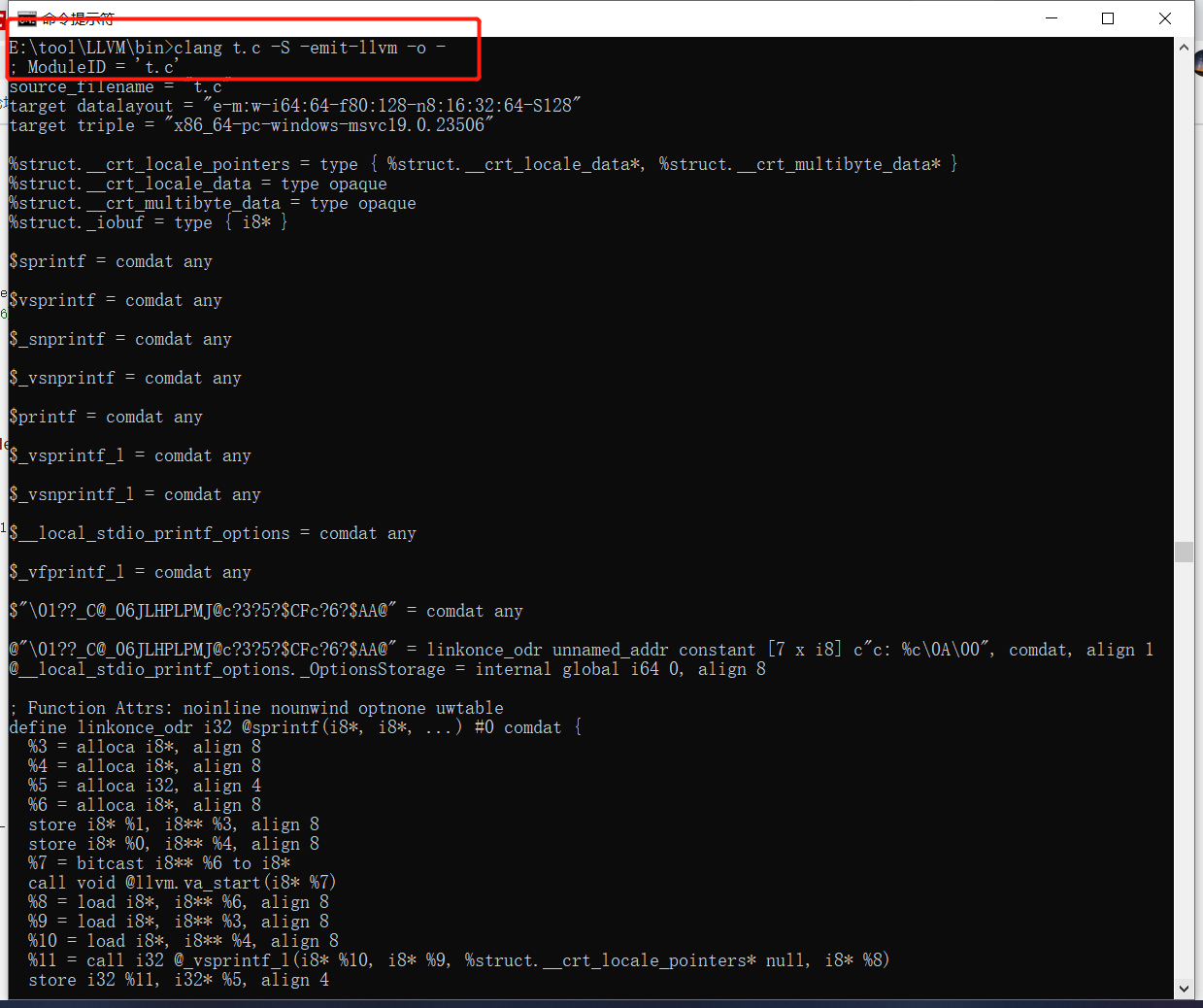
预处理结果：



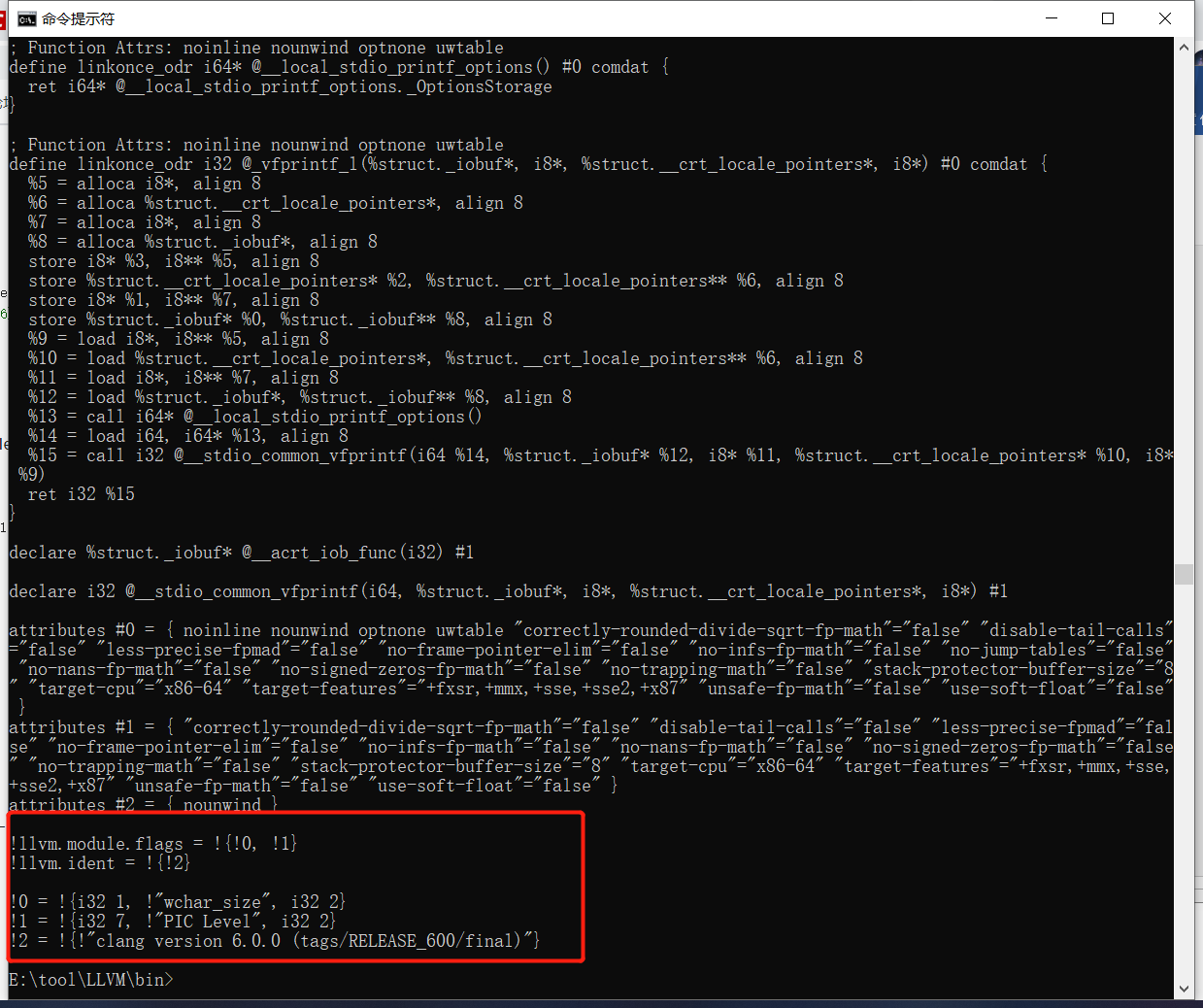
AST印刷精美：注意，-cc1参数指示应该运行编译器前端，而不是驱动程序。编译器前端还具有其他一些Clang特定的功能，这些功能不会通过GCC兼容的驱动程序接口公开。



使用LLVM生成代码：



输出较长，只展示部分结果：



在x86\_64上的结果：

