0.10 - 配置编译器：编译器扩展

亚历克斯于2018年9月19日| 最后由ALEX于2018年9月26日修改

C ++标准定义了程序在特定情况下应如何表现的规则。在大多数情况下，编译器将遵循这些规则。但是，许多编译器实现了对语言的更改，通常是为了增强与其他语言版本（例如C99）的兼容性，或者出于历史原因。这些特定于编译器的行为称为**编译器扩展**。

编写使用编译器扩展的程序允许您编写与C ++标准不兼容的程序。使用非标准扩展的程序通常不会在其他编译器（不支持那些相同的扩展）上编译，或者如果它们这样做，它们可能无法正确运行。

令人沮丧的是，默认情况下通常会启用编译器扩展。这对新学习者来说尤其具有破坏性，他们可能认为某些行为是官方C ++标准的一部分，而实际上他们的编译器过于宽容。

由于编译器扩展从来不是必需的，并且导致程序不符合C ++标准，因此我们建议关闭编译器扩展。

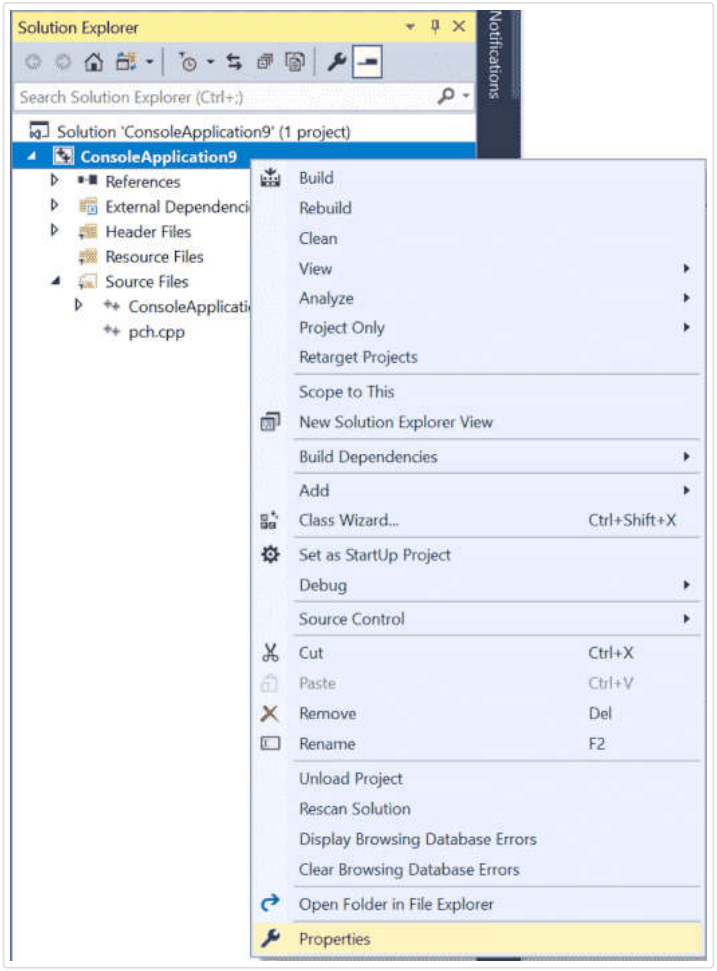
最佳实践

禁用编译器扩展以确保您的程序（和编码实践）符合C ++标准，并且可以在任何系统上运行。

禁用编译器扩展

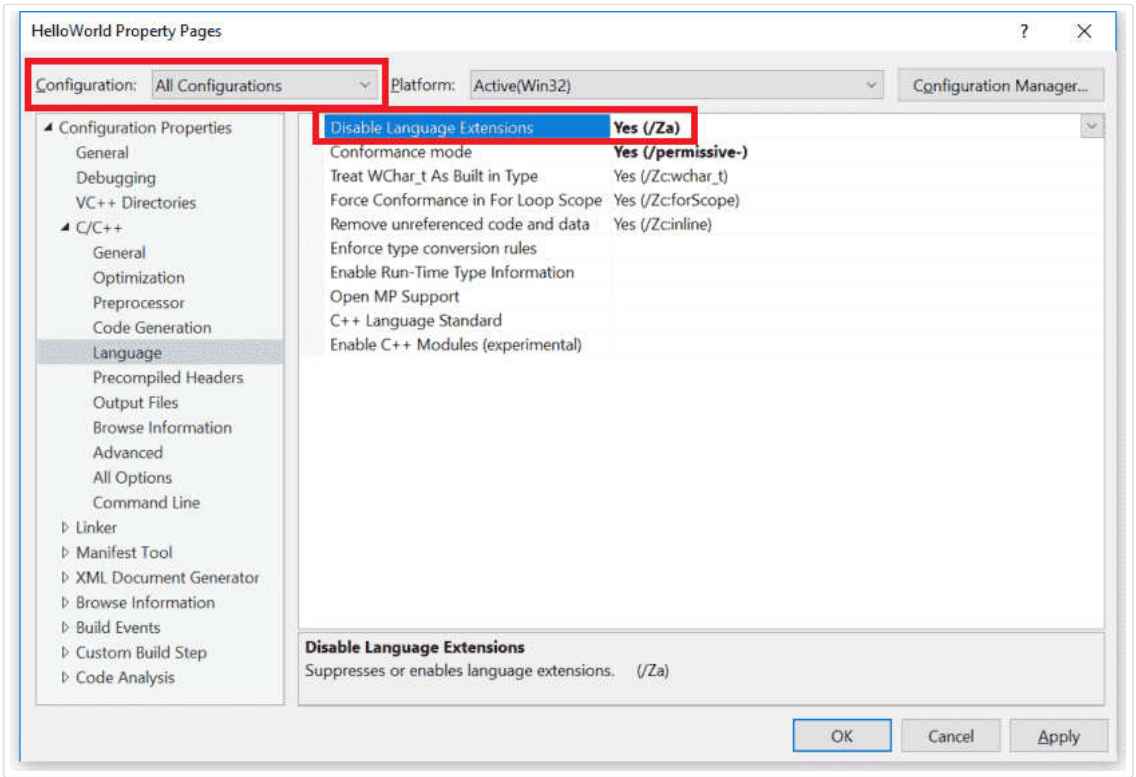
对于Visual Studio用户

要禁用编译器扩展，请在*Solution Explorer*窗口中右键单击项目名称，然后选择*Properties*：



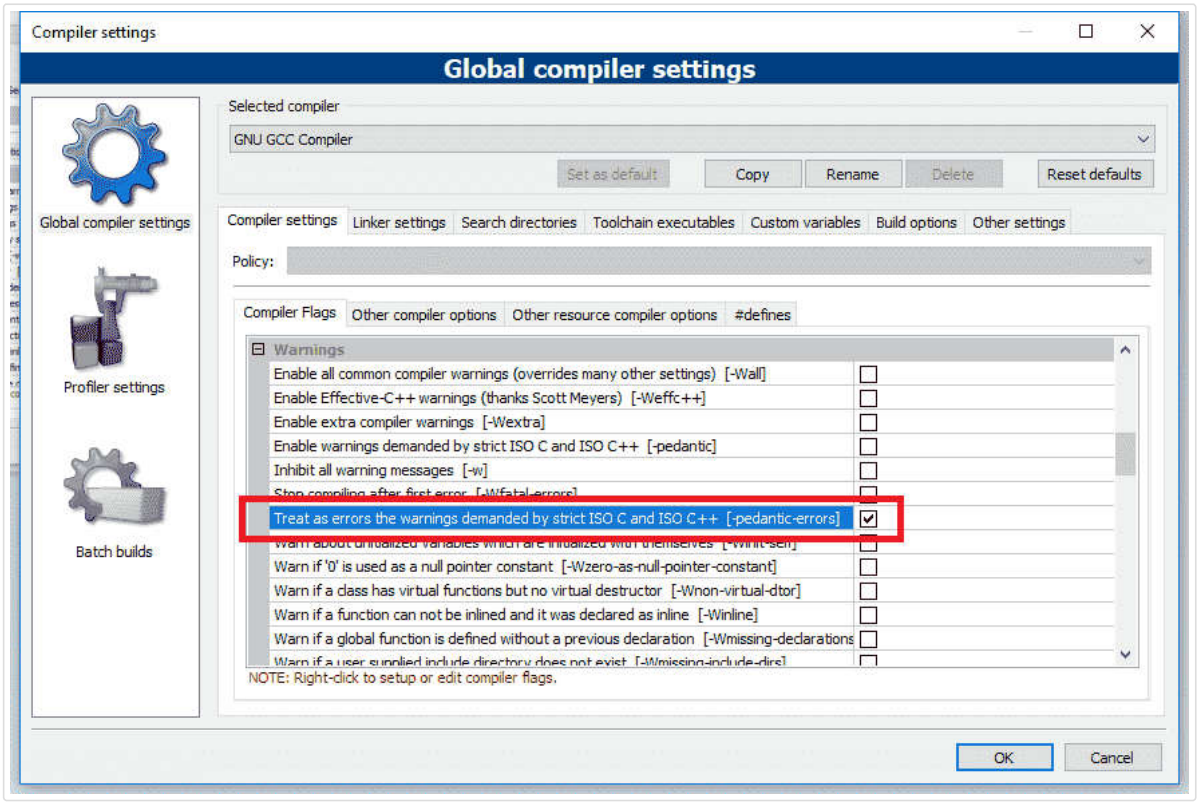
在“*项目*”对话框中，首先确保“*配置*”字段设置为“*所有配置*”。

然后，单击“*C / C ++*”>“*语言*”*选项卡*，并将“*禁用语言扩展*”设置为“*是*”（*/ Za*）。



对于Code :: Blocks用户

通过*“设置”菜单>“编译器”>“编译器标志”选项卡*禁用编译器扩展，然后查找并检查*-pedantic-errors*选项。



对于GCC / G ++用户

您可以通过将*-pedantic-errors*标志添加到编译命令行来禁用编译器扩展。

0.11 - 配置编译器：警告和错误级别

指数

0.9 - 配置编译器：构建配置

分享这个：

C ++教程 | 打印本文

19条评论到0.10 - 配置编译器：编译器扩展

Quackky
2018年11月11日晚上11:14 ·回复

我刚刚完成了HelloWorld项目的语言扩展禁用。我是否每次都要为将来的项目再做一次？

nascardriver
2018年11月12日上午9:46 ·回复

除非您的IDE有办法更改默认设置，否则您必须为每个新项目执行此操作。

戴夫
November 9, 2018 at 8:20 pm · Reply

Got the extensions disabled (Codeblocks) but there was another option checked already which was:
Have g++ follow the C++ I4ISOC++ language standard[std=C++I4]
Should that be left checked?
Sorry if is a dumb question but I know nothing about working with a compiler and only have used perl and php and want to have as much right as possible.

nascardriver
November 10, 2018 at 8:47 am · Reply

If that's the newest supported version, yes. If there's a newer version of C++ available in your compiler, use that.

Dave
November 10, 2018 at 10:33 am · Reply

Thanks, I just downloaded the compiler
codeblocks-17.12mingw-setup.exe from Sourceforge. When disabling the extensions was when I noticed.
Have a great day □

Leo
October 24, 2018 at 3:08 pm · Reply

How can i disable the complier extensions on Xcode? I have searched sound a bit and can't find anything on it. Is there something better I could be using on a mac? I could even install windows on another portion or something it it would be easier to use windows. What do you think?

Qluefqen
October 20, 2018 at 4:21 am · Reply

Hi, Alex,

In keeping with your advice to experiment, I've tried using this command-line compiler line:

```
g++ -Wall -pedantic-errors -std=c++17 -o <filename>.app <filename>.cpp
```

That works fine with g++, but when I try any variant of it with gcc, I get:

```
/usr/bin/ld: /tmp/ccWeSps4.o: in function `main':
hello.cpp:(.text+0xe): undefined reference to `std::cout'

```

```
/usr/bin/ld: hello.cpp:(.text+0x13): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&, char const*)'
/usr/bin/ld: hello.cpp:(.text+0x1d): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::endl<char, std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&)'
/usr/bin/ld: hello.cpp:(.text+0x28): undefined reference to `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
/usr/bin/ld: /tmp/ccWeSps4.o: in function `__static_initialization_and_destruction_0(int, int)':
hello.cpp:(.text+0x58): undefined reference to `std::ios_base::Init::Init()'
/usr/bin/ld: hello.cpp:(.text+0x6d): undefined reference to `std::ios_base::Init::~Init()'
```

Also, I found that this works best for me and would ask for your comments:

```
#include <iostream>

int main()
{
    std::cout << "The sum of 5 and 7 is "
              << 5+7
              << "."
              << std::endl;

    return 0;
}
```

I just find it easier to read and my compiler seems to have no problem with me not putting the semicolon at the end of every line. Am I screwing up my learning in any way?

nascardriver
October 20, 2018 at 4:26 am · Reply

Hi!

gcc compiles C, g++ compiles C++.

> Am I screwing up my learning in any way?
Nope, that's fine.

Hiruna
September 28, 2018 at 8:59 pm · Reply

Alex when I build solution after I give Yes(/Za) to disable language extentions in visual studio 2017 , It show an error, in it they say that (/Za) is an invalid value to disable language extentions

Alex
October 1, 2018 at 8:16 am · Reply

Can you share the exact error message you're getting?

bibek
September 22, 2018 at 10:49 pm · Reply

i really could not understand how to disable compiler extensions in G++
CAN U PLEASE CLARIFY IT

nascardriver
September 23, 2018 at 1:09 am · Reply

```
g++ -pedantic-errors ./main.cpp
```

Khang
September 20, 2018 at 2:44 am · Reply

Hi Alex,
If I disabled the compiler extension in Visual Studio 2015, when I use a header guard and include it inside my main file:
Header file (header.h):

```
#ifndef SOMEHEADER_H
#define SOMEHEADER_H

// some code here

#endif  // will get error C1004 : unexpected end-of-file found
```

Main file (main.cpp):

```
#include <iostream>
#include "header.h"

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

The compiler will throw an error telling me that #endif is illegal, but if I delete it, it will tell me that #endif is missing. Can you please tell me what is the problem here? Thank you.

Alex
September 20, 2018 at 8:47 am · Reply

I've having no issues compiling your code with compiler extensions disabled (substituting in a template class for // some code here)
If you remove all of the "// some code here", does it work?
Does it work if you disable compiler extensions again?

Khang
September 20, 2018 at 10:29 pm · Reply

I think I have found the problem, it seems like when I disable the compiler extension, I must press enter after the "#endif" preprocessor to make a newline.

```
#ifndef HAHA_H
#define HAHA_H

#endif   //press enter here
//must have an empty line here, not even a comment can be put here
```

If I enter a newline, the program will compile, and vice versa.

On the other hand, my "main.cpp" file doesn't need any empty line on Visual Studio 2015 with the compiler extension off, but when I tried it on C-free 5.0, it will cause an error and tell me it need a newline at the end. (The raw options for C-free compiler is "-g -DDEBUG -pedantic-errors")

```
#include <iostream>
#include "header.h"

int main()
{
   return 0;
}
// no need for newline here in VS2015, but needed in C-free
```

This have confused me a lot, if the compiler extension is unnecessary and potentially dangerous, why did they include it in the compiler in the first place?

nascardriver
September 21, 2018 at 1:15 am · Reply

Hi Khang!

TJ Seabrooks and Rakete1111 posted a nice explanation on stackoverflow (https://stackoverflow.com/a/72409/9364954).
Omitting the line feed at the end of a file could cause problems prior to C++11. Every C++11 and newer compiler should automatically add the line feed and not complain.

nascardriver
September 20, 2018 at 12:41 am · Reply

Hi Alex!

To my understanding -pedantic only disables extensions that prevent standard-conform programs from compiling and issues warnings when other extensions are used, but allows compilation. -pedantic-errors disables all extensions.

Alex
September 20, 2018 at 8:37 am · Reply

Thanks, I missed that nuance. Lesson updated!

Ahmed
September 19, 2018 at 2:45 pm · Reply

Thank you