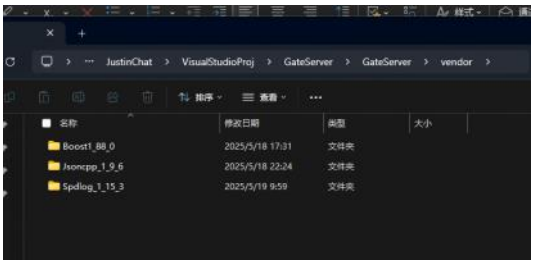
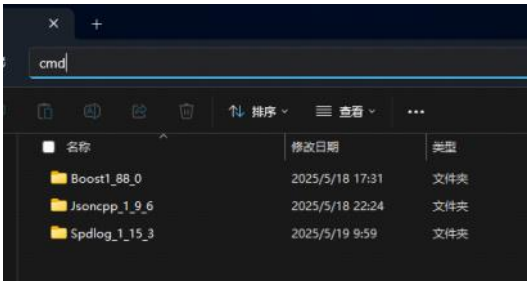
JstChat3

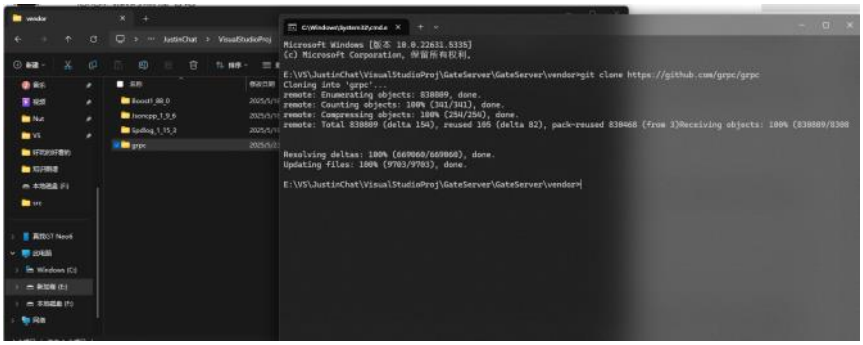
2025年5月22日 17:56

---------- Ep8 配置和使用 grpc ----------

》》》》打开VS项目的 vendor 目录



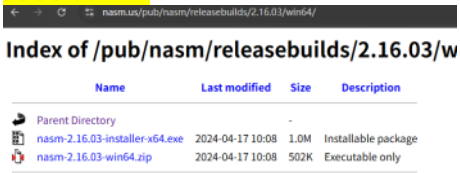




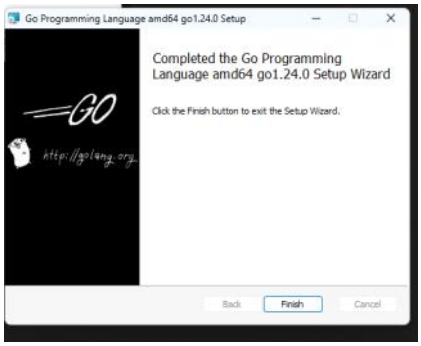
不要忘了 cd 一下

```
git clone https://github.com/grpc/grpc.git
cd grpc
git submodule update --init
```

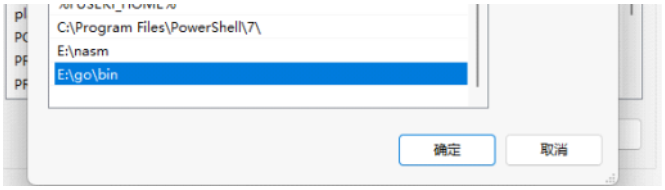
》》》》下载 nasm

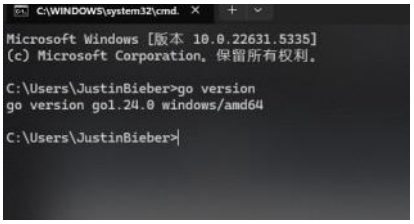


》》》》go(.msi文件)



下载好的环境中，我们找到 bin 目录，并将其加入 Path 环境变量中



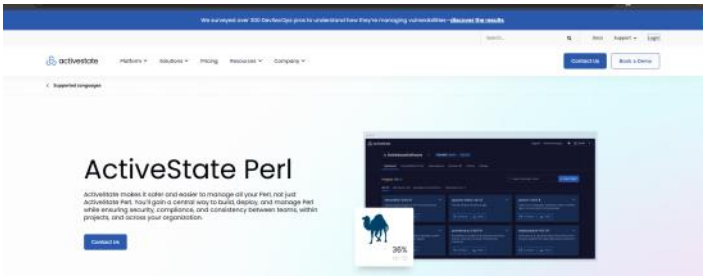


》》》》perl 安装

~~下载网址：https://www.perl.org/get.html~~

~~选择 Windows~~

~~我想下载 strawberry（因为草莓好吃），但是 UP 下载的是 ActiveState，那我下一个 ActivePerl 好了。~~

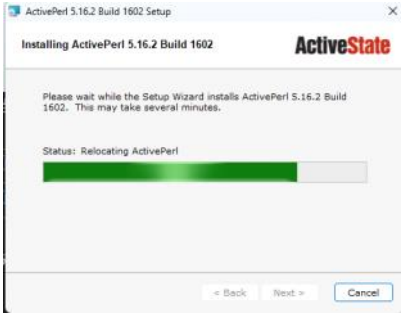


~~找了几分钟没找到 download 在哪。。。还是看了文档才知道怎么弄，头一次见下载还需要看教程的网站，我服了。~~

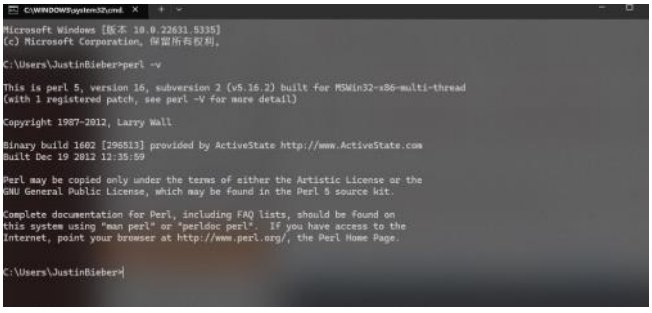
~~步骤指南（仅供参考）：~~	~~https://docs.activestate.com/platform/start/getting-started/~~
~~指南2（仅供参考）~~	~~https://docs.activestate.com/platform/state/~~

说实话没看懂，装了半天没弄出来个所以然。
从UP 的网盘下载 或者 去网上搜一个安装包吧，在安装上吃瘪也是无语了。

他妈的，得劲。



T_T



》》》》配置

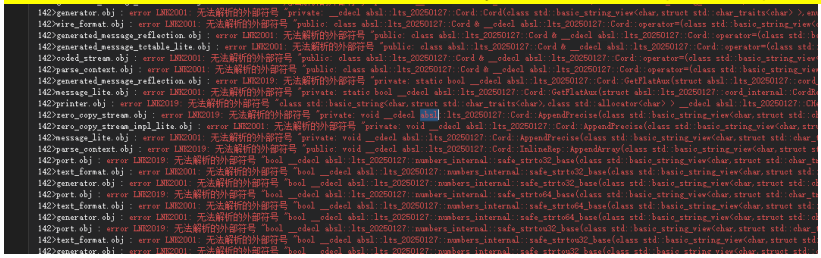
前情提要：在配置 grpc 之前我已经安装了如下文件。（最新版本相对于 2025.5.24）

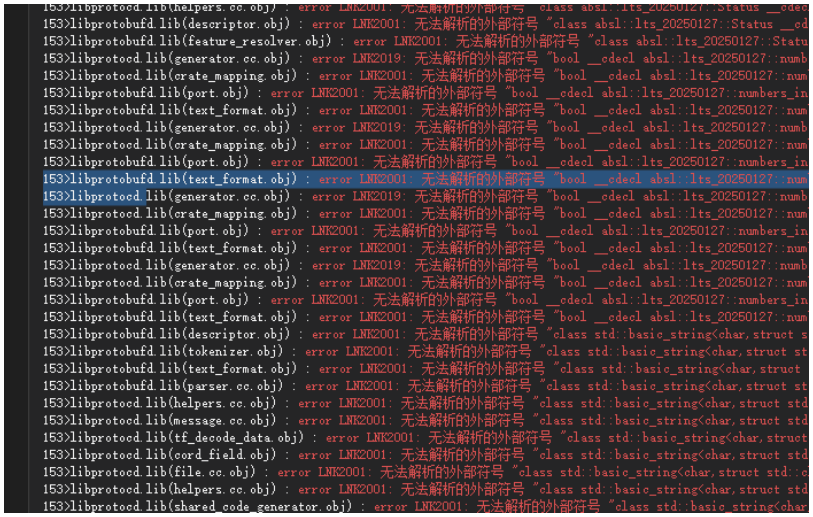
grpc	最新 (1.72.0)
cmake	3.27.2（很久之前安装的）
NASM	最新 (2.16.03)

go	最新(1.24.0)
Perl	UP主提供的版本(5.16.2)

操作和 UP主保持一致，但是编译时出现错误。

大概看了一眼报错：都是 absl 的缺失（或者未识别），还有是 .obj 报的错，还有一部分是 protobuf 和 protoc 这两个 lib 库报的错（Debug 模式下）

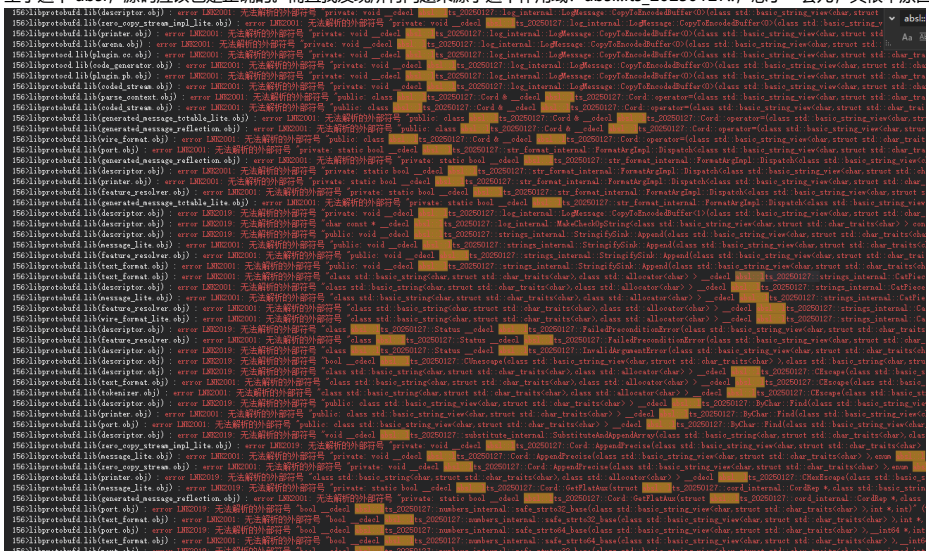




分析一手：

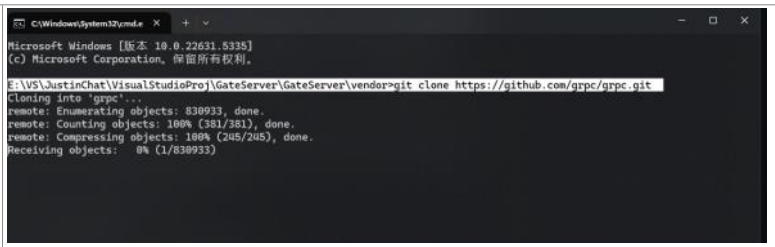
proto 源码应该是正确的，因为 grpc 会从子模版中下载其源码。

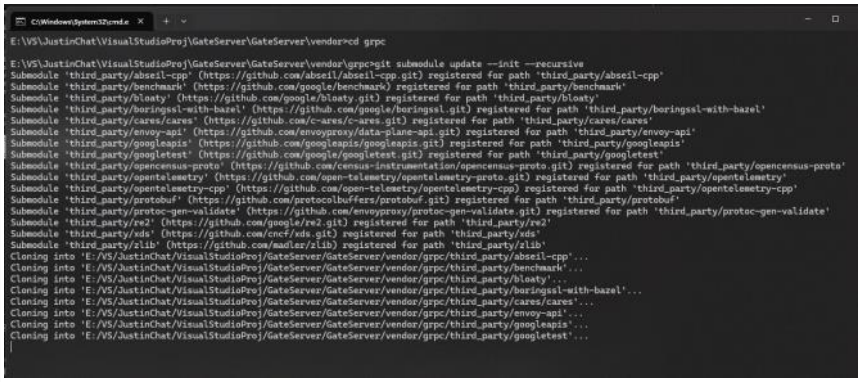
至于这个 absl，源码应该也是正确的。而且我发现所有问题来源于这个作用域：absl::lts_20250127::，想了一会儿，其根本原因应该是Abseil库版本与gRPC不兼容或未正确链接。

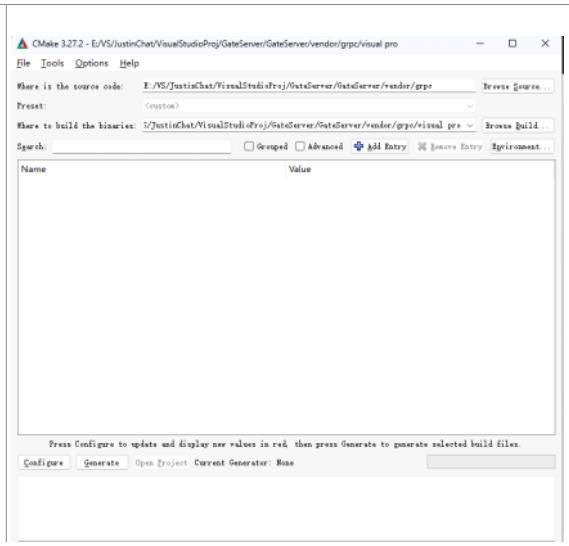


那么我们在配置过程中一定要注意 absl (abseil) 的版本问题，也要注意某些操作会不会导致链接失败。可是absl应该和其他子模板一样，都是远程下载下来的，理应不会出错才是。

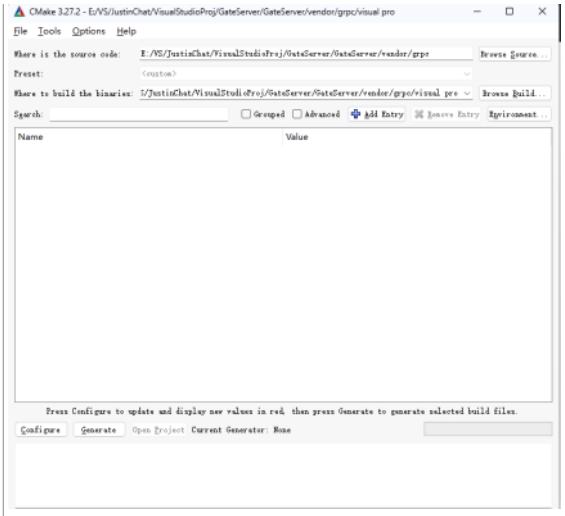
但我决定重新来一次，为了避免错误，我删除了 grpc 然后重新安装它，从头再来过。

第一步： （我全程采用 github 管理，而非 git，这对科学上网有一定要求）	 此时的存储大小为：

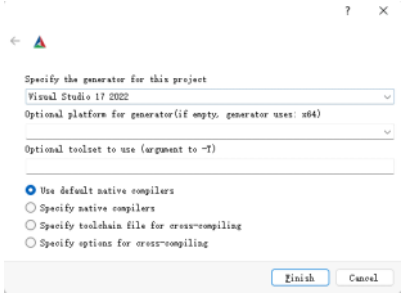
	大小: 553 MB (580,342,126 字节) 占用空间: 570 MB (597,942,272 字节)
第二步：	（很关键）更新子模板，并下载子模板中的依赖项 （子模板指的是某一个库的在线依赖项，比如 grpc 在这里与其子模板库保持远程联系，通过.gitmodules 文件我们可以查看到所关联的库。git脚本也可以通过这个文件索引到正确的库，并及时更新，保证 grpc 可以引用最新的源码）  持续更新/初始化中：  最终存储结果： 大小: 2.58 GB (2,770,460,647 字节) 占用空间: 2.65 GB (2,854,240,256 字节)

第三步：
（可以使用命令行，也可以使用 cmake GUI，前提是你下载了 cmake）
	去查了一下，发现要配置 C++ 17 需要将 cmake 更新为 3.8 版本以上的，我是 3.27。
	`C:\Users\JustinBieber>cmake --version`
	`cmake version 3.27.2`
	
	`CMake suite maintained and supported by Kitware (kitware.com/cmake).`
	
	GUI 操作流程：
	第一步：打开 Cmake GUI，然后选择文件夹
	（第一个是源码目录，第二个是构建之后的项目输出目录）
	 

选择文件夹
（第一个是源码目录，第二个是构建之后的项目输出目录）

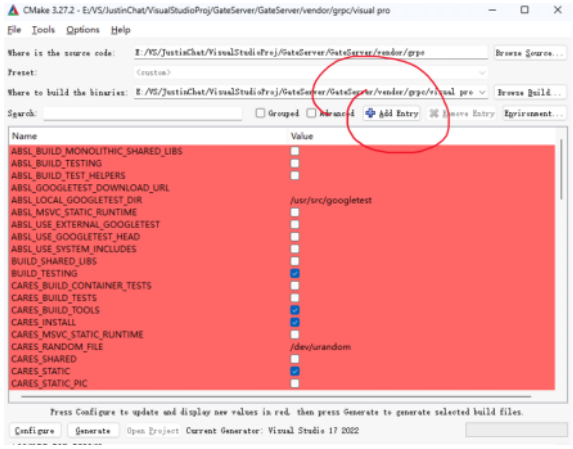


第二步：左下角的 configure，单击一下。
选择一个合适的 VS 编辑器版本，剩下的默认就行，点击 finish，先运行一下

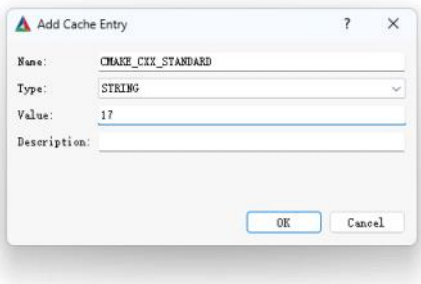


第三步：配置之后，我们添加一下3个指令：

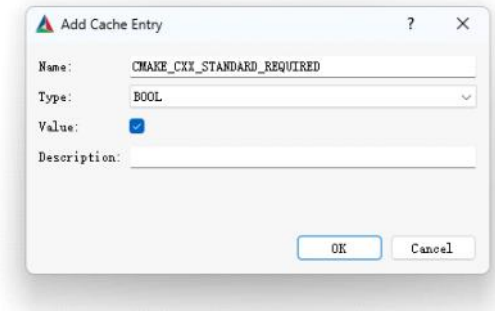
为了指明 C++ 17，这样操作



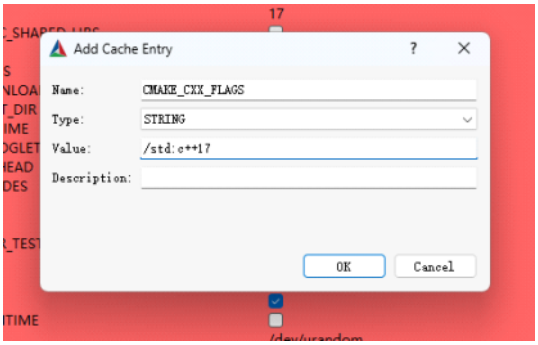
输入第一个：（指定C++ 17）



第二个：（表示允许 Cmake 为项目指定 C++ 版本）



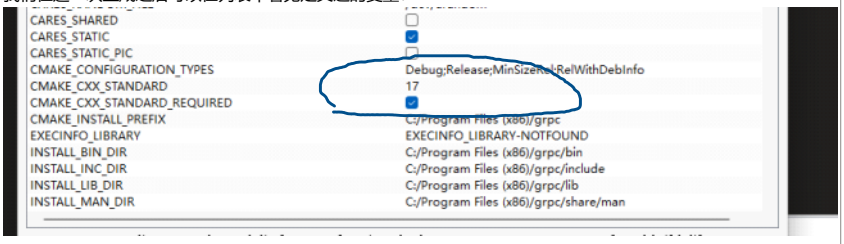
第三个：（如果使用 Visual Studio，需显式指定编译器标志）



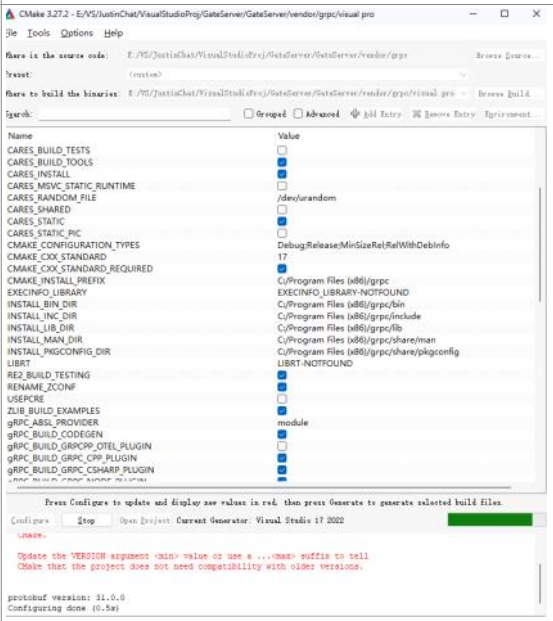
可以看见在这里：

然后再次单击 Configure，确保新添加的变量生效

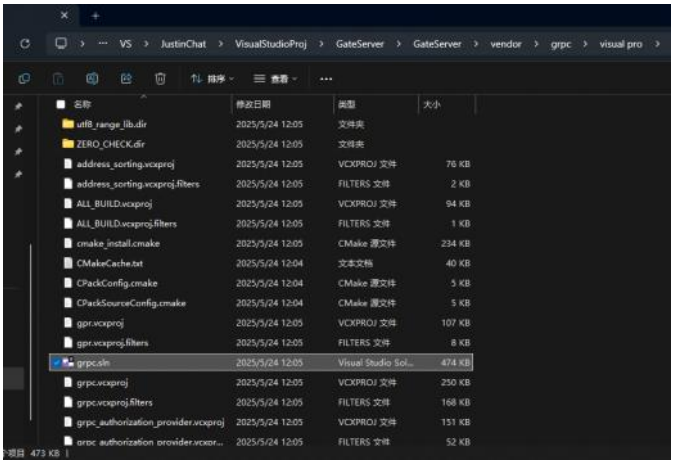
我们在这一次生成之后可以在列表中看见定义过的变量：



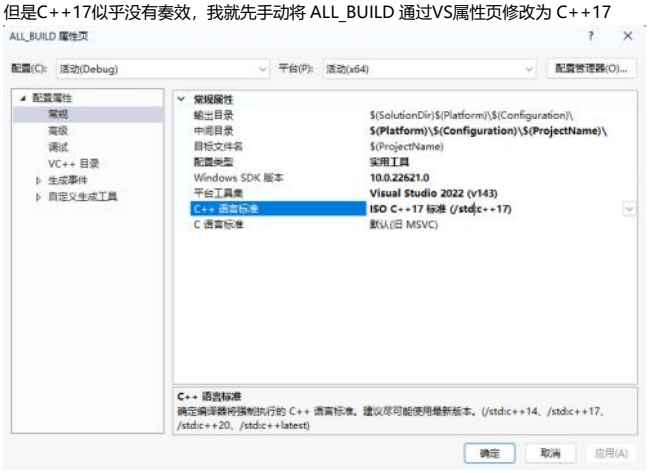
最后，单击左下角的Generate生成一下项目：

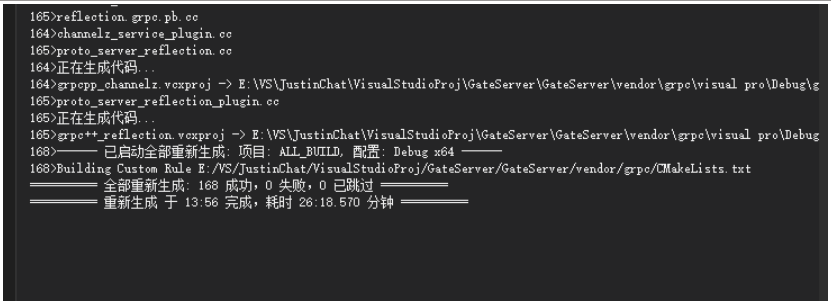


生成结束后，我们去visual pro 文件夹下找一下生成的 .sln 文件，并打开（如果生成的是 VS 2022，记得用 VS 2022 打开）



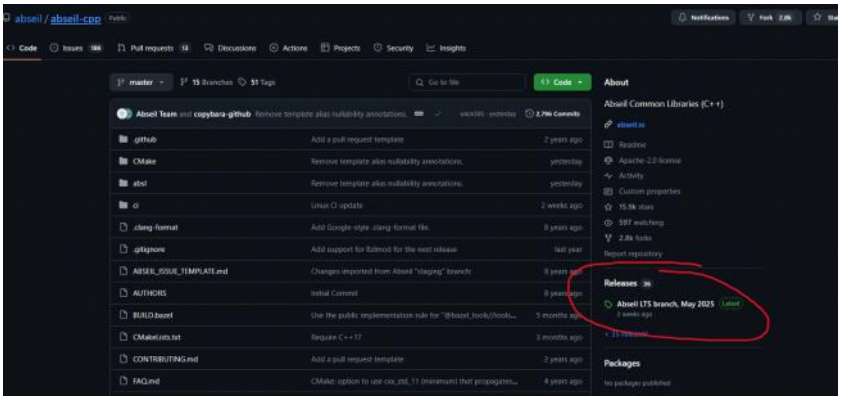
但是C++17似乎没有奏效，我就先手动将 ALL_BUILD 通过VS属性页修改为 C++17

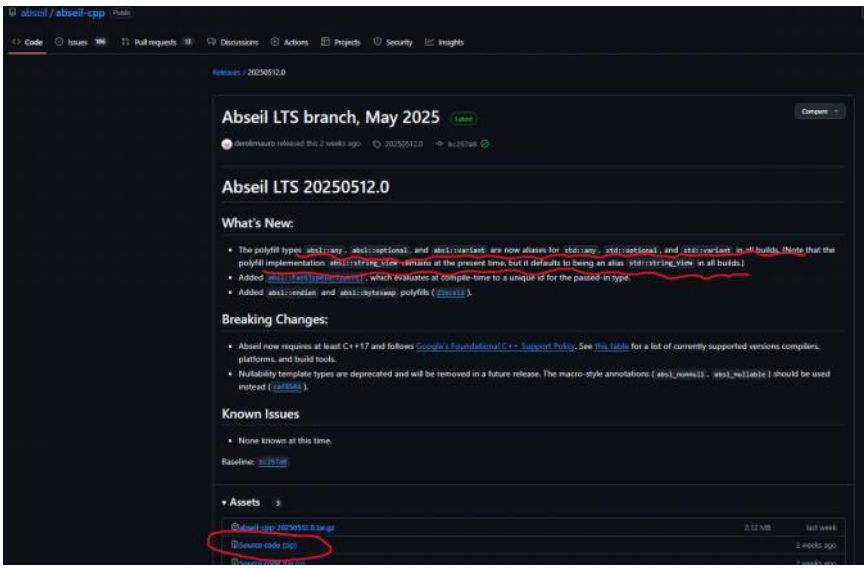


第四步：	然后进行重新生成。
提示：	在此期间，我手动下载了最新版本的 Abseli，不知道是不是这个因素修复了 Absl::Its 的问题。 （因为有可能VS 编辑器在寻找不到 Abseli 的时候，会在系统文件中搜索，而刚好找到了我新下载的文件。我下载的grpc和 abseli均是最新版的，所以两者应该不会发生冲突。） 或者可以尝试把 abseli 中新下载的文件复制一下，然后将 grpc 中 abseli(absl）的相关文件替换一下。
最后，仔细端详一下我的编译成功信息。 我操，运行成功这一下太爽了，希望我的操作流程对大家有参考价值。	

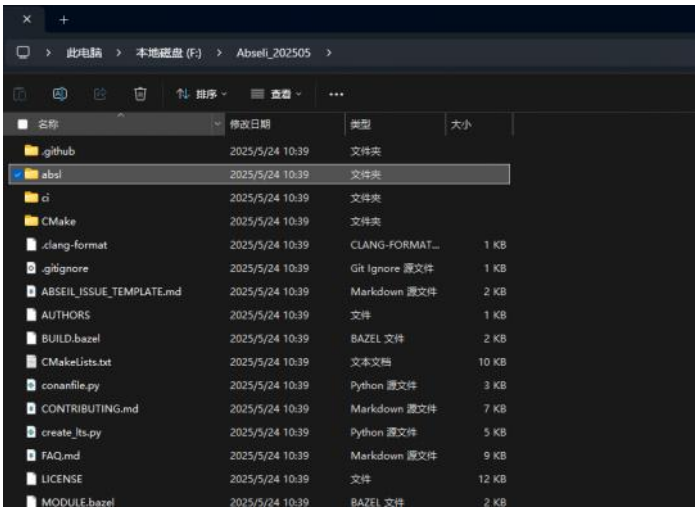
》》》》参考（我也不确定有没有奏效）

一怒之下，我去 github 下载了最新的 abseli 的压缩包

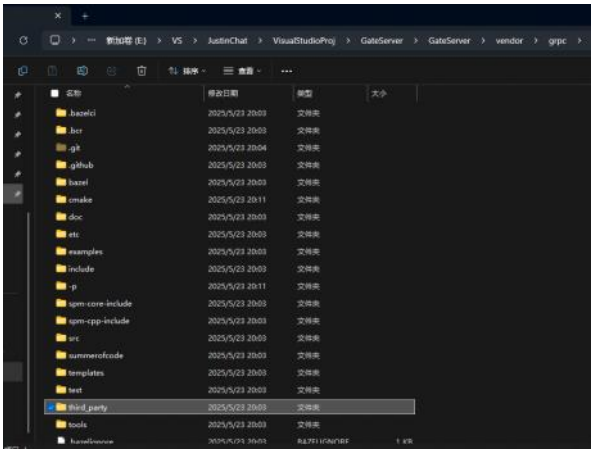


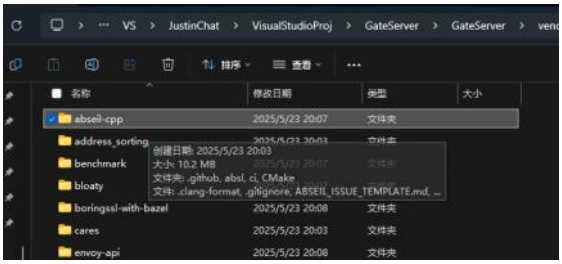


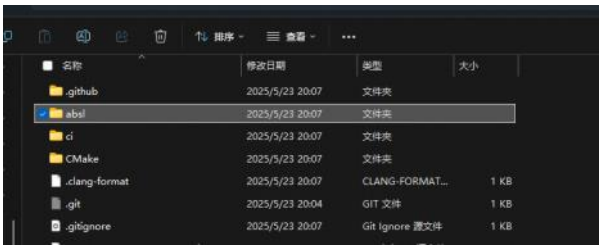
解压之后，将ABseil 的子文件夹 absl 中的所有文件复制一下



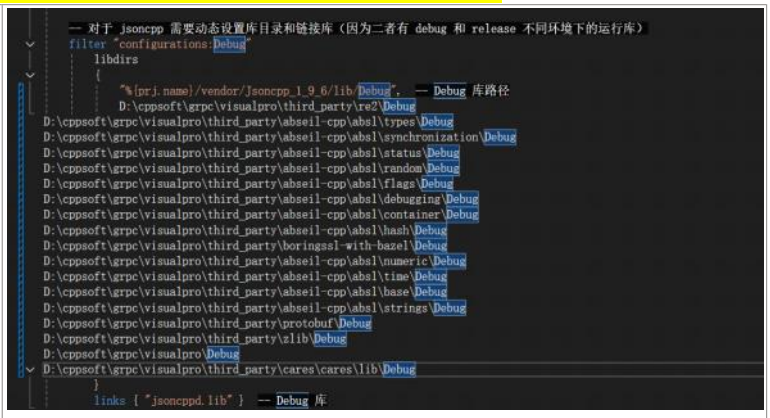
然后将这些文件放入 grpc 的 third_party/abseli-cpp/absl中：





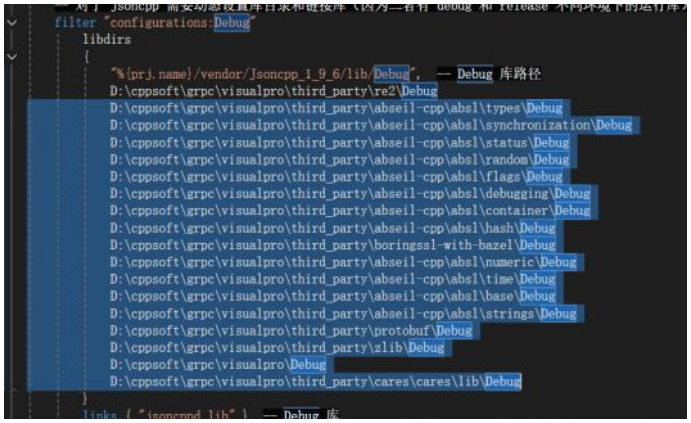


》》》》 接下来就是导入库了，而我不使用属性管理器这个东西，这里你将看到使用 premake 会有多么方便。

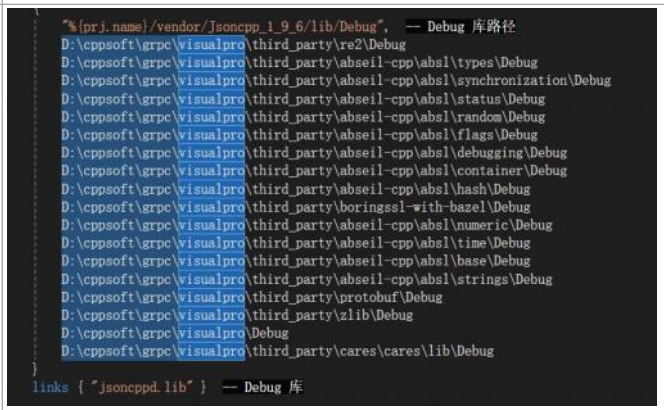
将需要添加的库贴过来	

```
    -- 对于 jsoncpp 需要动态设置库目录和链接库（因为二者有 debug 和 release 不同环境下的运行库）
    filter "configurations:Debug"
        libdirs
        {
            "%{prj.name}/vendor/Jsoncpp_1_9_6/lib/Debug",  -- Debug 库路径
            D:\cppsoft\grpc\visualpro\third_party\re2\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\types\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\synchronization\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\status\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\random\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\flags\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\debugging\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\container\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\hash\Debug
D:\cppsoft\grpc\visualpro\third_party\boringssl-with-bazel\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\numeric\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\time\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\base\Debug
D:\cppsoft\grpc\visualpro\third_party\abseil-cpp\absl\strings\Debug
D:\cppsoft\grpc\visualpro\third_party\protobuf\Debug
D:\cppsoft\grpc\visualpro\third_party\zlib\Debug
D:\cppsoft\grpc\visualpro\Debug
D:\cppsoft\grpc\visualpro\third_party\cares\cares\lib\Debug
        }
        links { "jsoncppd.lib" }  -- Debug 库
```

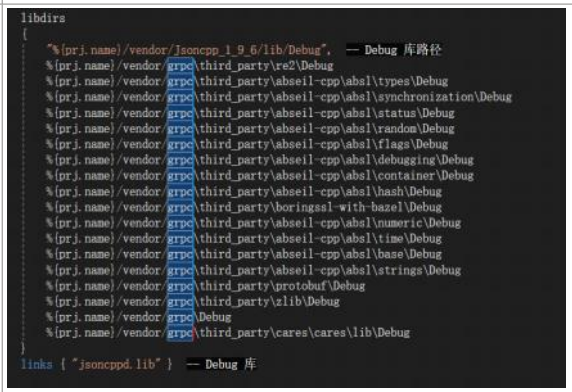
选中，然后按 tab，快速调整



然后按住 alt 用鼠标选择需要修改的目录：



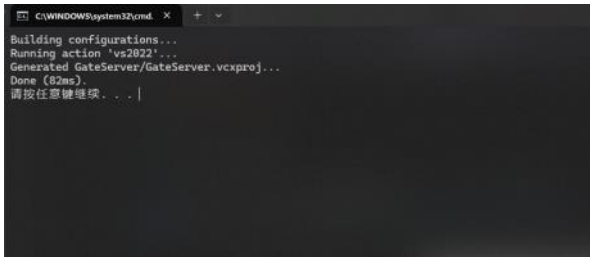
键入文字：



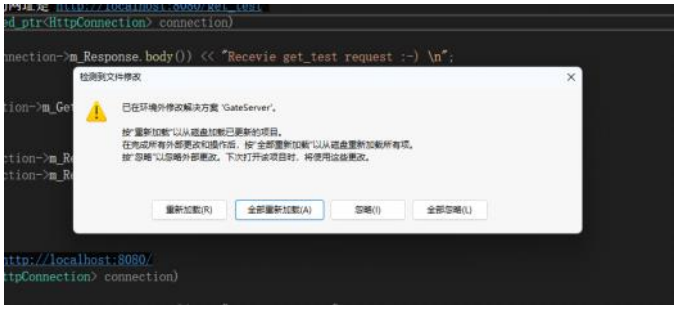
加上 “”

这样脚本就编写完成了。

然后我们去运行一下 .bat 文件



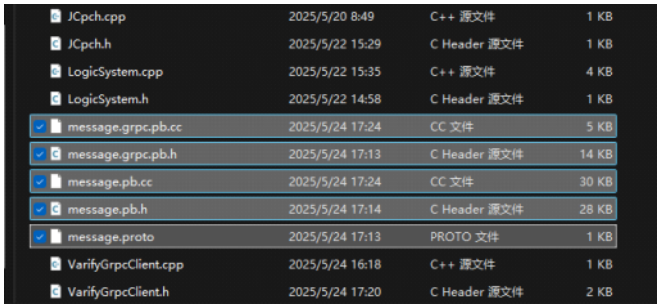
重载一下项目：



》》》》为了测试 grcp 必须先写一个 message.proto，然后在 cmd 中运行命令，并将生成的文件放入 src 区域

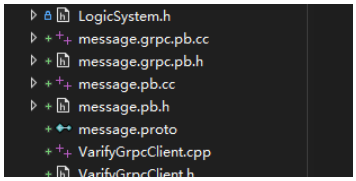
在项目根目录下建立 message.proto，并在这个位置运行 cmd ,然后运行指令。

运行完成之后将文件放入项目源码目录中。



》》》》新生成的文件，需要告诉 premake 脚本，说项目需要包含这些文件

打开VS之后，你会发现这些文件的确被包含在项目中了



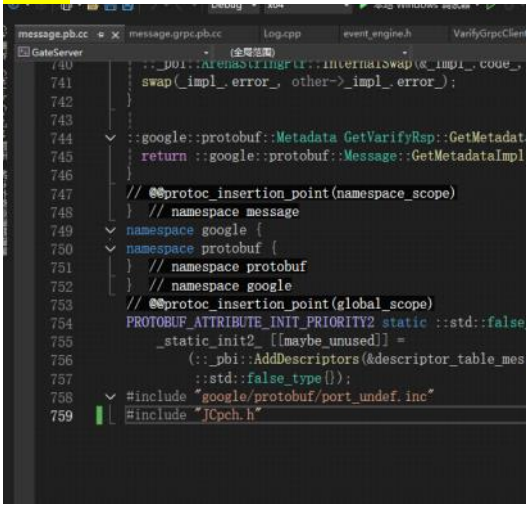
》》》》由于使用了 absl 等等库，这些库又使用了 c++ 17 新特性，所以需要在 premake 中指定一下项目是 C++ 17

```
15
16  project "GateServer"
17      location "GateServer"
18      kind "ConsoleApp"
19      language "C++"
20      cppdialect "C++17"          --C++标准（编译时）
21
22      targetdir ("bin/%{cfg.buildcfg}-%{cfg.system}-%{cf
23      objdir ("bin-int/%{cfg.buildcfg}-%{cfg.system}-%{c
```

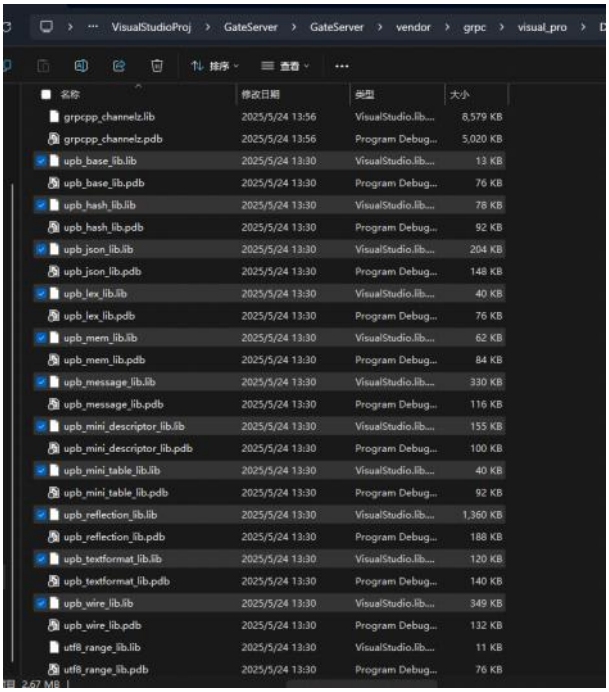
可以解决以下错误：

C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C7525	内联变量至少需要 "/std:c++17"	GateServer	generated_me...	62
C7525	内联变量至少需要 "/std:c++17"	GateServer	generated_me...	65
C7525	内联变量至少需要 "/std:c++17"	GateServer	generated_me...	66
C7525	内联变量至少需要 "/std:c++17"	GateServer	generated_me...	67
C7525	内联变量至少需要 "/std:c++17"	GateServer	generated_me...	68
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C7525	内联变量至少需要 "/std:c++17"	GateServer	message.h	1648
C7525	内联变量至少需要 "/std:c++17"	GateServer	message.h	1650
C7525	内联变量至少需要 "/std:c++17"	GateServer	message.h	1652
C7525	内联变量至少需要 "/std:c++17"	GateServer	message.h	1654
C7525	内联变量至少需要 "/std:c++17"	GateServer	message.h	1656
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119
C2338	static_assert failed: 'Protobuf only supports C++17 and newer.'	GateServer	port_def.inc	119

》》》》由于我使用了预编译头文件，所以会提示说这两个 .cc 的文件没有包含 JCpch.h ， 我们在文件末尾包含一下。



》》》》旧版本的 grpc 中生成的 upd.lib 库，如今在新版的 grpc 中被分开定义为多个 .lib

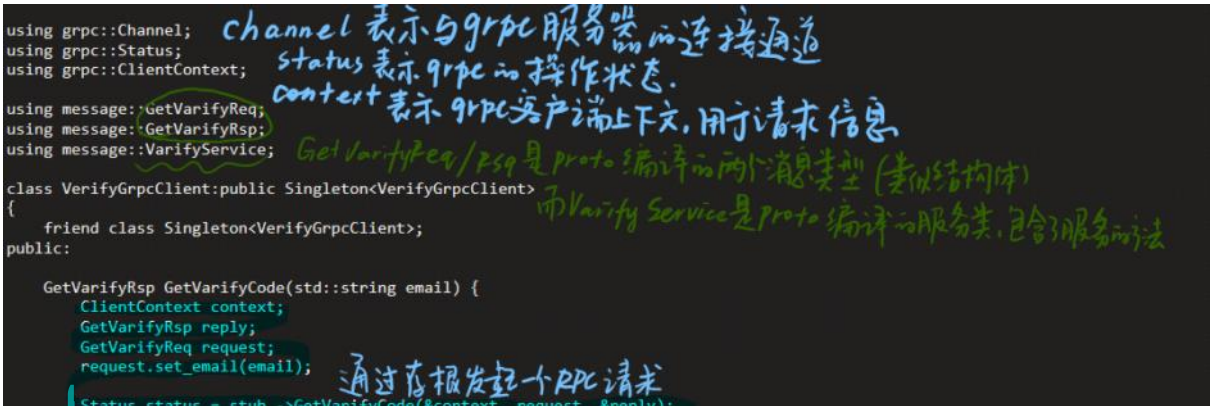


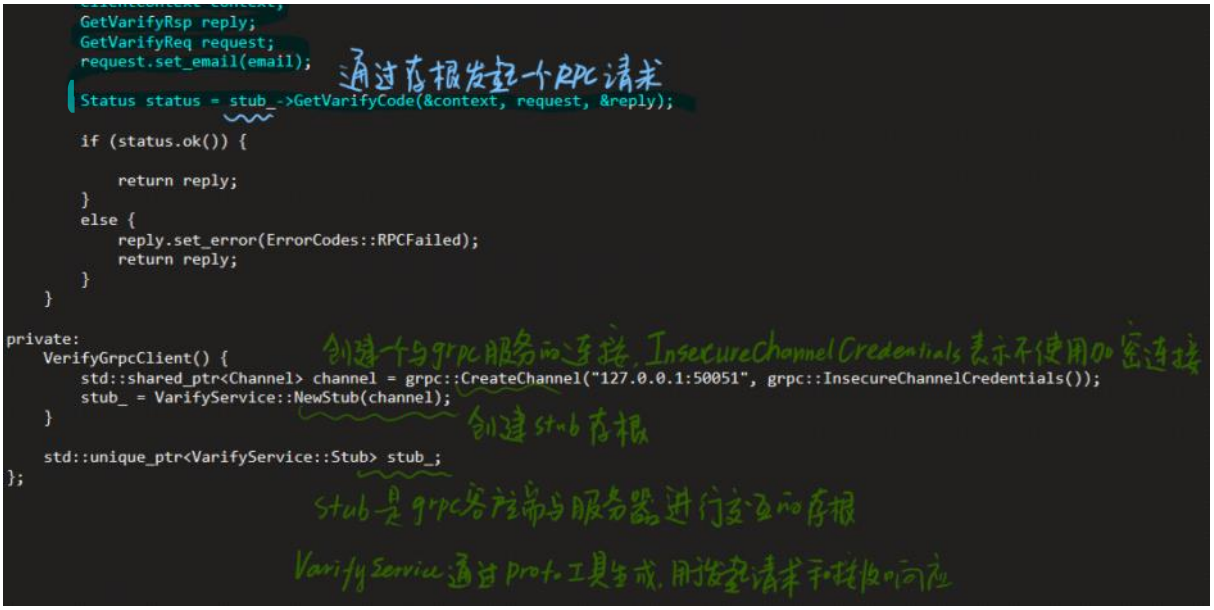
可以看到已经没有了 upd.lib 取而代之的是多个 upd_XXXX.lib

absl_flags.lib 也是，现在absl_flags.lib已经不被支持了，取而代之的是很多的 absl_flags_XXX.lib 库

还有 absl_leak_check_disable.lib 这个库现在好像不再使用了，我也没找到这个库。
这些东西我都在 premake 脚本中更改了，不再使用的库不能包含，否则会引起链接错误。

》》》》GRPC 代码解析





》》》》unique_ptr ??

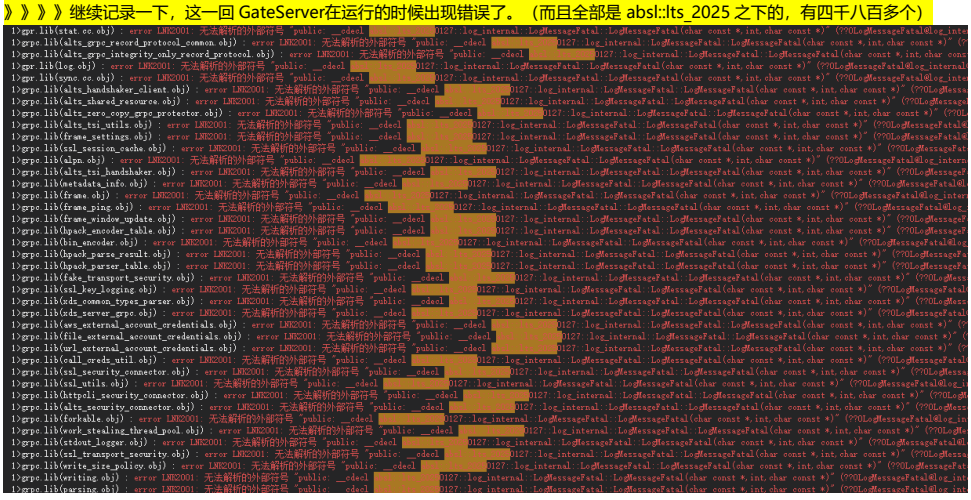
unique_ptr 是 C++11 引入的智能指针之一，属于标准库中的 <memory> 头文件。

它是一个拥有独占所有权的指针，意味着它管理的资源（通常是堆上的对象）只能有一个 unique_ptr 拥有。

当 unique_ptr 被销毁时，它会自动释放资源，从而避免了手动管理内存的麻烦，减少了内存泄漏的风险。

特性：	**独占所有权**：unique_ptr 保证它指向的对象在任何时候只有一个 unique_ptr 拥有。你不能将它们直接复制给另一个 unique_ptr，这样做会引发编译错误。如果想要将所有权转移给另一个 unique_ptr，必须使用 std::move。
	
	std::unique_ptr<int> p1 = std::make_unique<int>(10);
	std::unique_ptr<int> p2 = std::move(p1); // 只能通过 std::move 转移所有权

》》》》继续记录一下，这一回 GateServer在运行的时候出现错误了。（而且全部是 absl::lts_2025 之下的，有四千八百多个）



其实全都是因为最新的 grpc 需要很多其他的库，而 UP 给的 文本名单可能没有包含这些最新的库，我们需要添加一下。（查看 premake 脚本）

实际上需要添加挺多的库，而且这个错误没什么人遇到，我想了半天才想到。

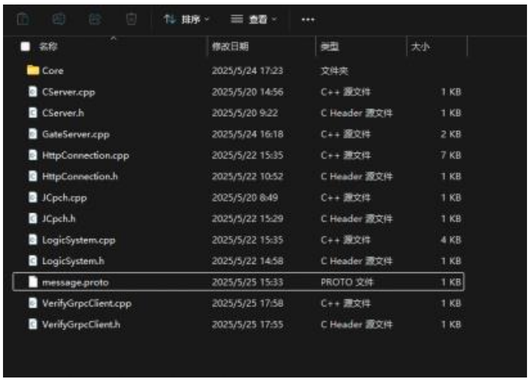
》》》》现在只剩下 11 个

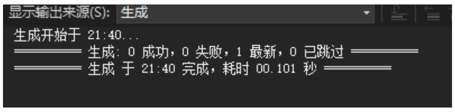
那我重新生成一下 proto 好了

我直接在项目src 这个文件夹这里创建一个 message.proto， 并在这里运行指令：

```
E:\VS\JustinChat\VisualStudioProj\GateServer\GateServer\vendor\grpc\visual_pro\third_party\protobuf\Debug\protoc.exe  -I="." --grpc_out="." --plugin=protoc-gen-grpc="E:\VS\JustinChat\VisualStudioProj\GateServer\GateServer\vendor\grpc\visual_pro\Debug\grpc_cpp_plugin.exe" "message.proto"

E:\VS\JustinChat\VisualStudioProj\GateServer\GateServer\vendor\grpc\visual_pro\third_party\protobuf\Debug\protoc.exe --cpp_out=. "message.proto"
```





》》》》