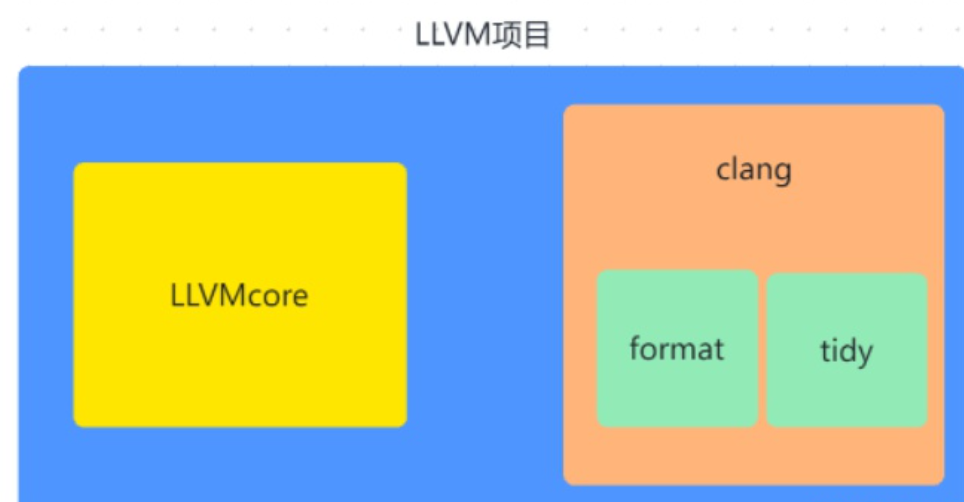
## VisualStudio2022下clang使用教程

### LLVM 17.03包地址

LLVM中集成了clang-tidy和clang-format这两种工具，所以只需要安装LLVM即可，不需要对clang-tidy和clang-format分别去安装。（安装时请检查本机的操作新系统位数）



安装地址：

1. 官网地址：<https://github.com/llvm/llvm-project/releases/tag/llvmorg-17.0.3>
2. 离线地址：见tools文件夹

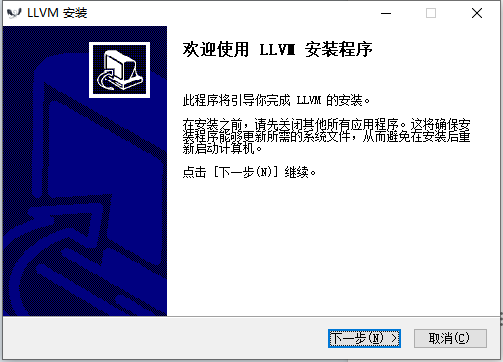
### LLVM 17.0安装步骤

以本机为例，本机为64位的系统。

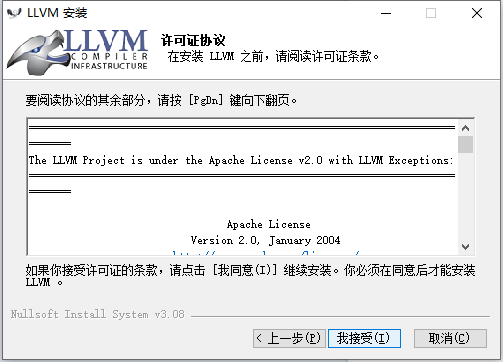


①双击LLVM-17.0.3-win64.exe

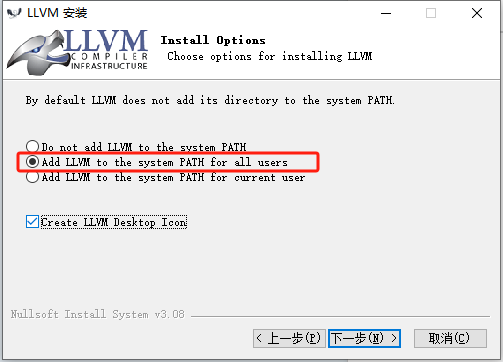
②点击下一步



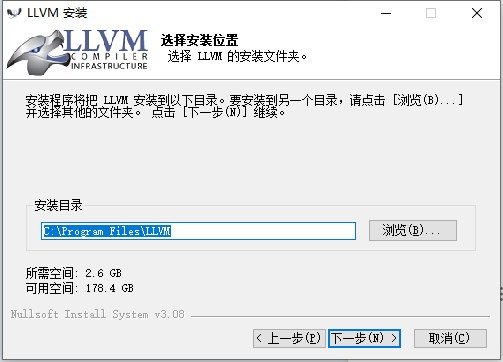
③点击我接受



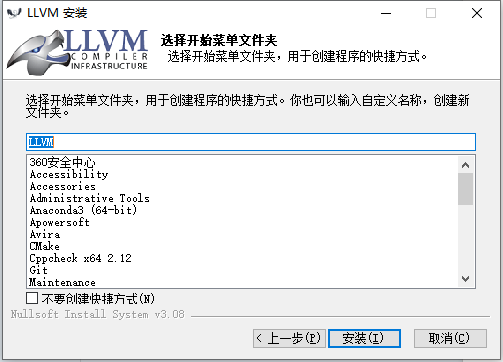
④加入环境变量，并点击下一步，在桌面创不创建无所谓



⑤选择安装路径后，点击下一步

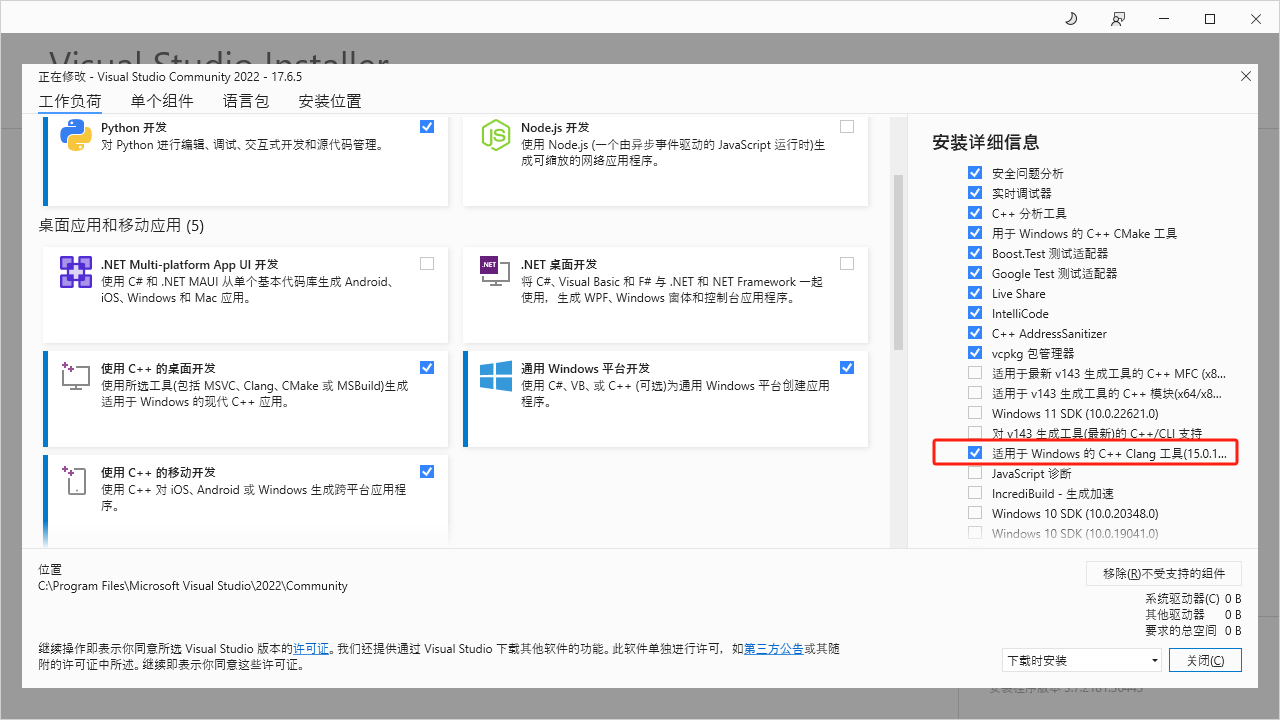


⑥点击安装后，等待安装完成即可。

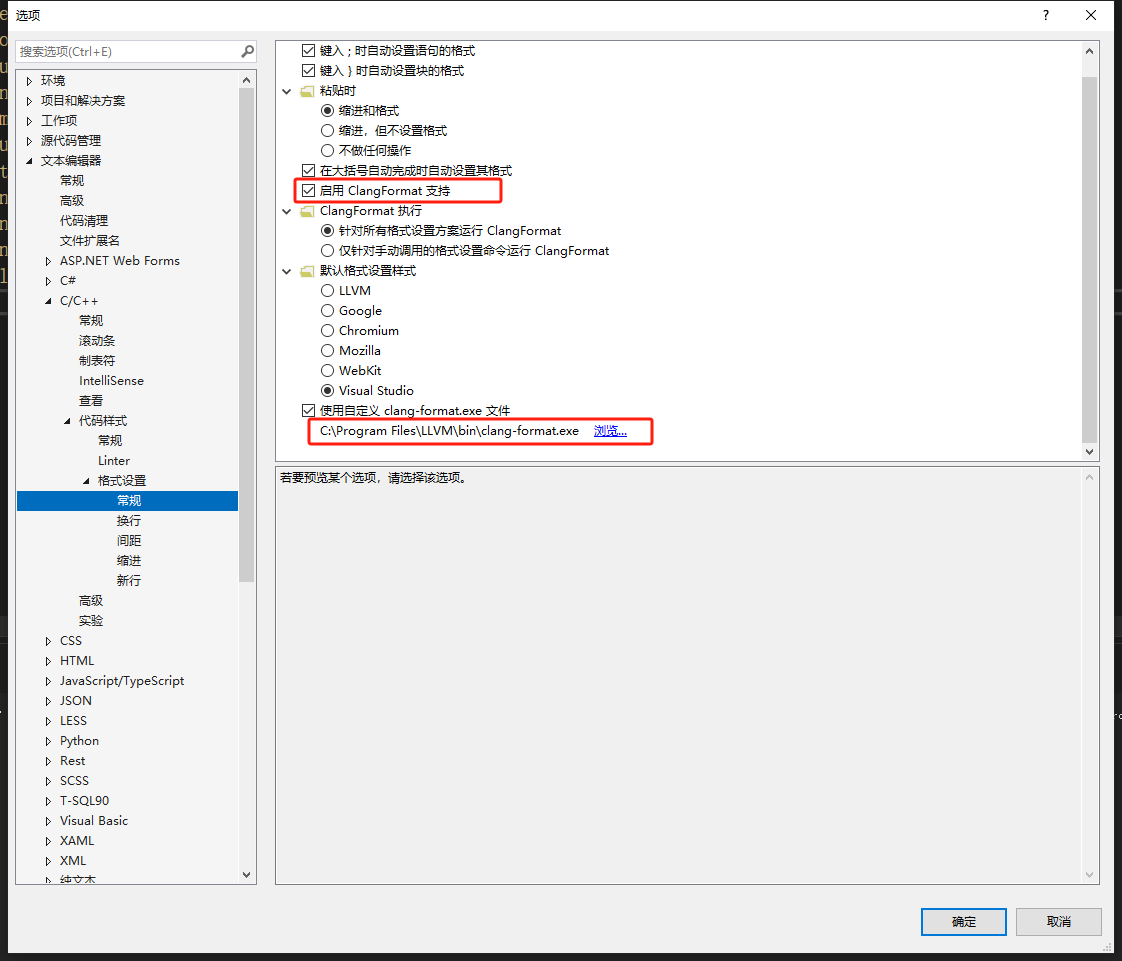


### Visual studio 2022使用format工具

检查一下是否已经安装了clang，如果未安装，请安装一下Visual studio 2022默认适用于windows的clang工具包。



①打开visual studio 2022，进入工具>>选项>>文本编辑器>>C/C++>>代码样式>>格式设置>>常规。选择已安装的17.03版本的clang-format.exe。



②拷贝.clang-format规则文件到项目根目录

相关规则如《format配置说明》文档所示

③对每一个文件执行ctrl+k+D，即可完成格式化操作。

④若在使用.clang-format文件过程中，发现有规则需要增删改的情况，请上报至各室主任或项目经理，进行统一评估审批，不得擅自修改配置文件（因为修改后，你的代码风格和别人不能保持统一，同样失去了格式化的一部分含义）。

### Visual studio 2022使用tidy工具

Tidy工具是属于代码静态检查工具。只有本地最终通过该工具检查无误后的代码才允许上库。

#### 4.1 cmake工程使用tidy操作步骤

对于cmake工程，无法从项目属性中将tidy工具启用，有两种方法，一是通过在cmake中进行配置，使得编译时对代码进行检查，而是通过脚本对代码进行批量检查，这里介绍cmake的情况，脚本方法请自行从官网下载。

Tidy的相关检查的清单列表如下链接所示：

<https://clang.llvm.org/extra/clang-tidy/checks/list.html>

①打开CmakeSettings.json（如果打开是gui界面，请在右上角点击JSON），添加

"enableClangTidyCodeAnalysis": true,

"enableMicrosoftCodeAnalysis": true,

②在cmakelist中增加以下配置。

set(CMAKE\_EXPORT\_COMPILE\_COMMANDS ON)

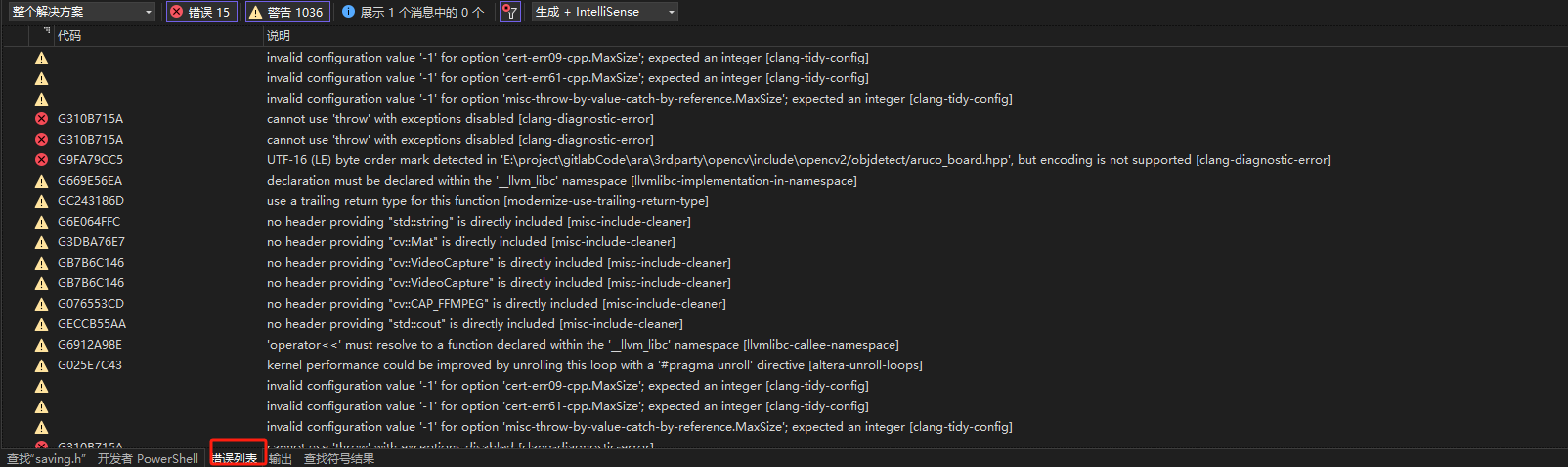
find\_program(CLANG\_TIDY\_BIN NAMES "clang-tidy")

if(CLANG\_TIDY\_BIN)

set(CMAKE\_CXX\_CLANG\_TIDY "${CLANG\_TIDY\_BIN}" "-checks=\*")

endif()

③执行编译，在错误列表中可看到检查的输出



#### 4.1 cmake工程使用tidy操作步骤（以脚本的方式使用）

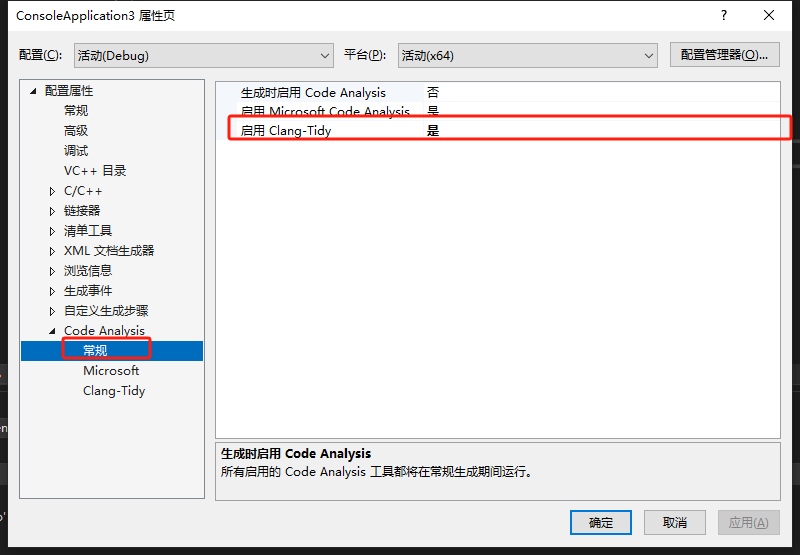
1、在cmakelist中加入set(CMAKE\_EXPORT\_COMPILE\_COMMANDS ON)

2、将脚本run-clang-tidy.py拷到compile\_commands.json所在的目录

3、在cmd中执行python run-clang-tidy.py -config-file=”[.clang-tidy的路径]”

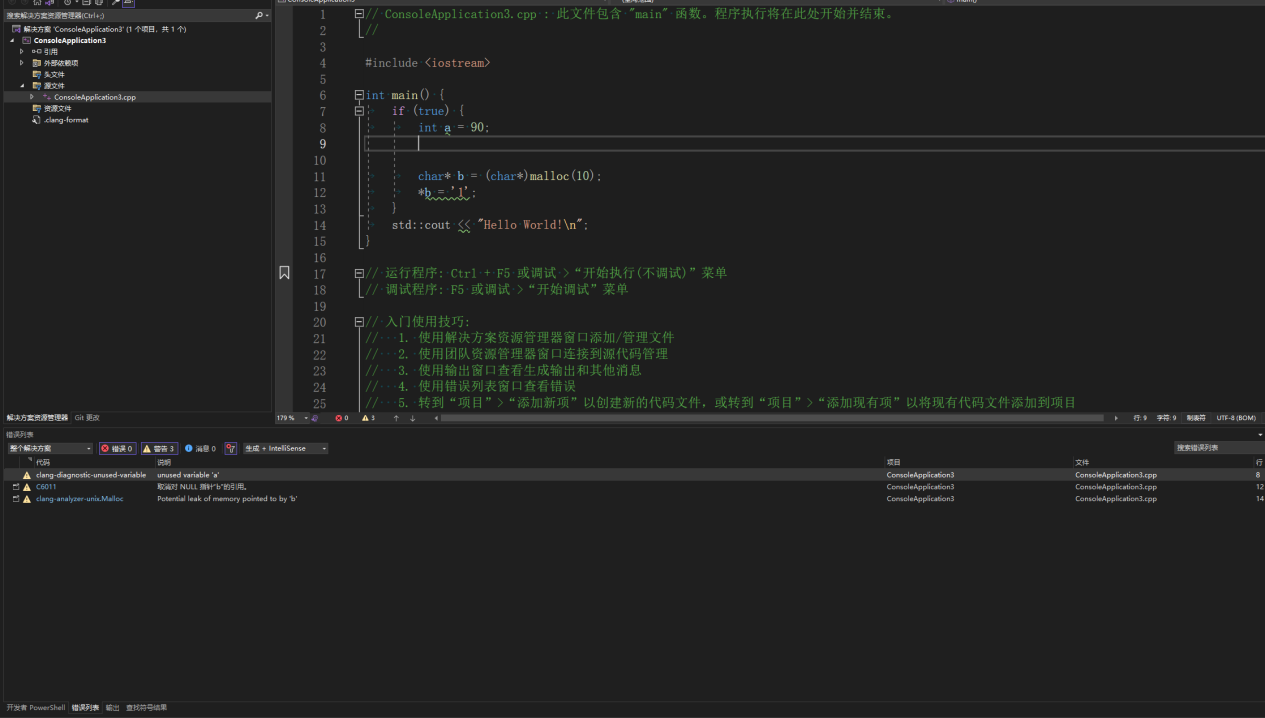
#### 4.1 sln工程使用tidy操作步骤

①右键点击项目>>属性>>常规，启动clang-tidy



②将.clang-tidy文件放到需要检查的工程目录下

③执行编译，告警出现在错误列表中



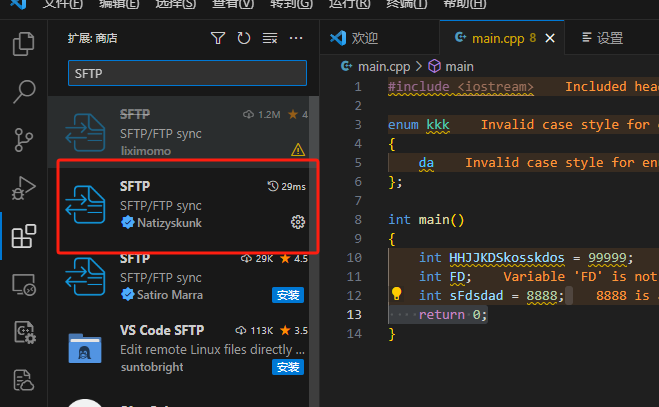
### 交叉编译(建议方法一)

##### 方法一：

本方法需要借助SFTP，原理是通过SFTP将远程的代码同步到本端，利用本端的工具和插件进行检查，在在本端修改后的代码可以通过SFTP再同步到远端。

步骤如下：

1. 本端vscode安装SFTP



1. 安装完成后配置SFTP：在本端新建一个空白目录，使用vscode打开该空白目录，然后在vscode中使用ctrl+shift+p打开命令界面，输入SFTP：config

出现以下内容

{

    "name": "远端IP",

    "host": "远端IP",

    "protocol": "sftp",

    "port": 22,

    "username": "远端用户名",

    "password": "远端密码",

    "remotePath": "需要检查的代码路径",

    "uploadOnSave": false,

    "useTempFile": false,

    "openSsh": false

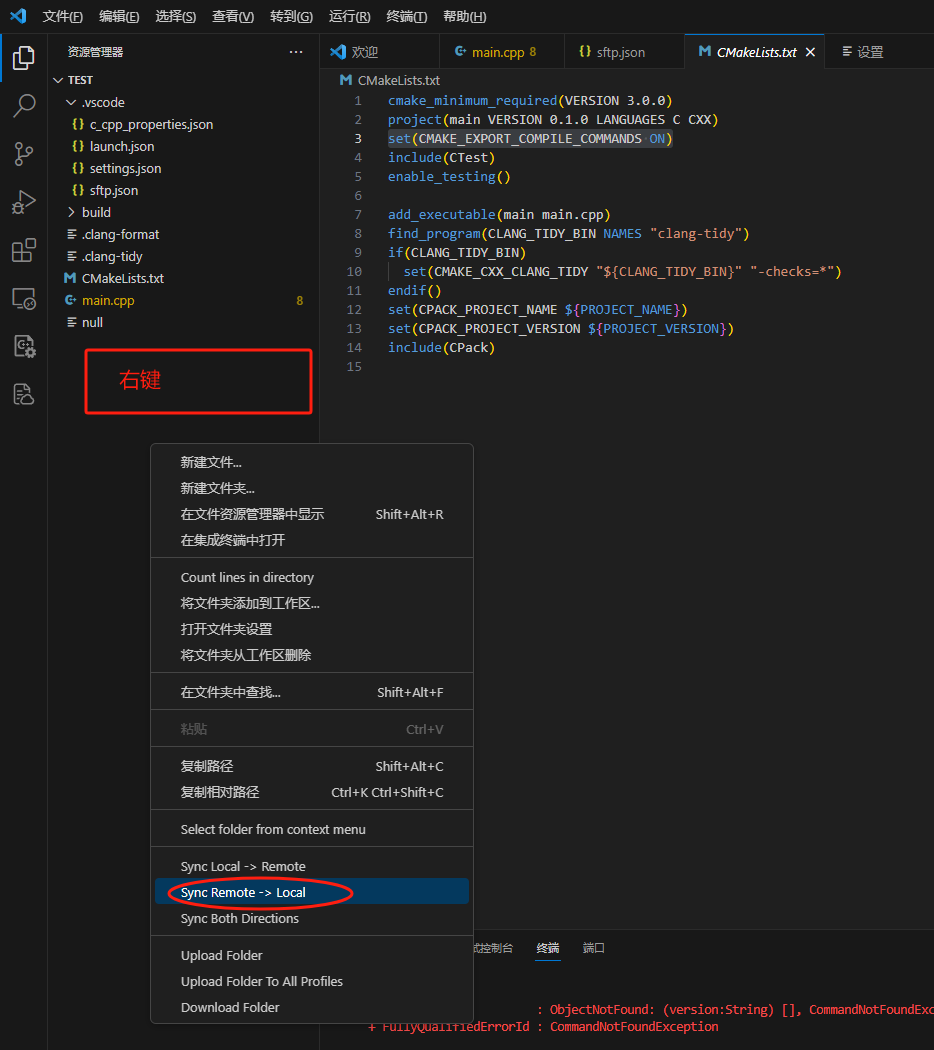
}

1. 配置SFTP完成后，需要有一个前提，就是需要检查的代码路径中需要包含compile\_commands.json文件。至于怎么生成该文件，我们可以通过在cmakelist中加入：

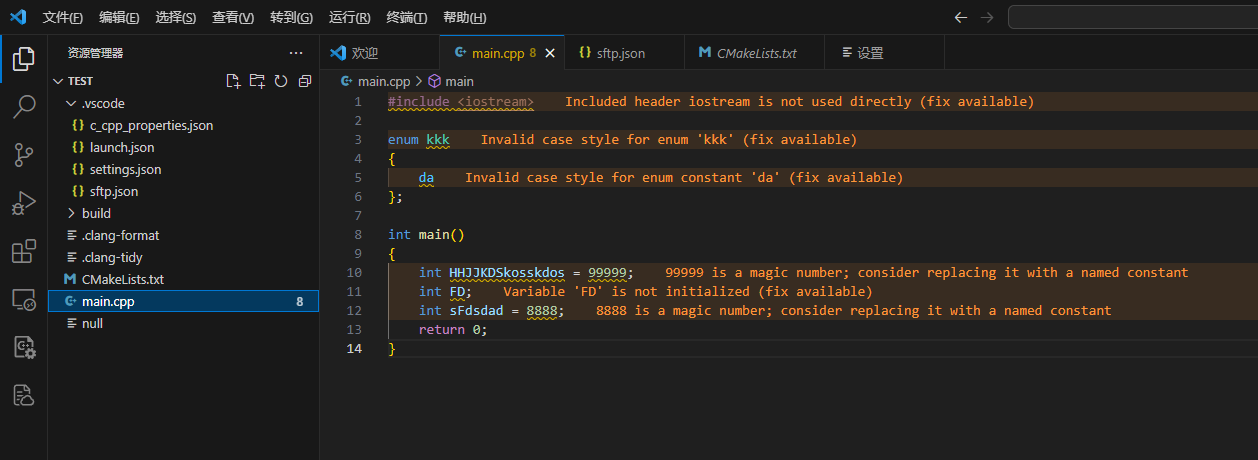
set(CMAKE\_EXPORT\_COMPILE\_COMMANDS ON)

或者如果是makefile文件，通过bear生成compile\_commands.json文件。

compile\_commands.json文件准备就绪后，按如下进行操作。



然后打开代码，就可以看到clang对远端代码进行检查的提示了。



##### 方法二：

（机器存在加密的请自行解决）

环境：

|  |  |
| --- | --- |
| 环境 | 工具 |
| 本端（windows） | vscode（下载C/C++、CMake、CMake Tools、remote插件） |
| 远端linux | 如下 |

Linux需要安装如下：

1. 安装clangd-linux-17.03

从gitlab仓库上的tool里边取https://gl.xtkfpt.online/codecheckcfg/codecheckcfg

1. 添加clangd到环境变量

vim /etc/profile

export PATH

source /etc/profile

1. 升级cmake到3.20以上（安装cmake-3.26.0）

Cmake包在<https://gl.xtkfpt.online/codecheckcfg/codecheckcfg>中的tools取

cd cmake-3.26.0

sudo ./bootstrap

sudo make

sudo make install

# 一步到位

sudo ./bootstrap && sudo make && sudo make install

1. 安装ninja

sudo apt install ninja-build

如果安装慢或者失败，换源试试

换源方法

编辑/etc/apt/sources.list文件, 在文件最前面添加以下条目(操作前请做好相应备份)：或者备份后，把里边的都删除再加上以下内容即可：

# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-updates main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-updates main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-backports main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-backports main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-security main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-security main restricted universe multiverse

# 预发布软件源，不建议启用

# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-proposed main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-proposed main restricted universe multiverse

换完之后执行下面两条命令

sudo apt-get update

sudo apt-get upgrade

更新完成后再执行以下命令安装ninja

sudo apt install ninja-build

1. 安装LLVM，包在<https://gl.xtkfpt.online/codecheckcfg/codecheckcfg>中的tools取

（1）cd llvm-project-llvmorg-17.0.3

（2）mkdir build

（3）cd build

（4）cmake -G "Ninja" -DLLVM\_ENABLE\_PROJECTS="clang;clang-tools-extra;compiler-rt;" -DLLVM\_TARGETS\_TO\_BUILD=X86 -DCMAKE\_BUILD\_TYPE="Release" -DLLVM\_INCLUDE\_TESTS=OFF -DCMAKE\_INSTALL\_PREFIX="/usr/local/llvm" ../llvm

（5）ninja && ninja install

（6）同理，命令执行完以后，立刻输入echo $?命令查看上一条命令的执行情况，如果返回值为0，则说明build命令执行成功。

输入指令clang -v验证是否安装成功

### 怎么屏蔽相关文件

##### 屏蔽文件夹

直接在该文件夹中新建一个.clang-tidy，文件中输入以下内容即可

Checks: '-\*'

##### 屏蔽**文件**

屏蔽文件不适用于工程的形式，目前只能通过命令行的形式去屏蔽，自行查看以下说明：

$ clang-tidy --help

USAGE: clang-tidy [options] <source0> [... <sourceN>]

OPTIONS:

Generic Options:

  --help                         - Display available options (--help-hidden for more)

  --help-list                    - Display list of available options (--help-list-hidden for more)

  --version                      - Display the version of this program

clang-tidy options:

  --checks=<string>              -

                                   Comma-separated list of globs with optional '-'

                                   prefix. Globs are processed in order of

                                   appearance in the list. Globs without '-'

                                   prefix add checks with matching names to the

                                   set, globs with the '-' prefix remove checks

                                   with matching names from the set of enabled

                                   checks. This option's value is appended to the

                                   value of the 'Checks' option in .clang-tidy

                                   file, if any.

                                   翻译：

                                   用逗号分隔的glob列表，带有可选的'-'前缀。glob按照出现在列表中的顺

                                   序处理。没有“-”前缀则添加检  查，带有'-'前缀的则删除相应检查。选

                                   项的值会被附加到.clang-tidy配置文件中的'Checks'选项,如果有的话。

  --config=<string>              -

                                   Specifies a configuration in YAML/JSON format:

                                     -config="{Checks: '\*',

                                               CheckOptions: {x, y}}"

                                   When the value is empty, clang-tidy will

                                   attempt to find a file named .clang-tidy for

                                   each source file in its parent directories.

                                   翻译：

                                   指定配置的格式为YAML/JSON ：

                                   -config="{Checks: '\*',CheckOptions: {x, y}}"  当值为空时，

                                   clang-tidy将尝试在父目录中的每个源文件中找到一个名为.clang-tidy的

                                   文件。

  --config-file=<string>         -

                                  Specify the path of .clang-tidy or custom config file:

                                    e.g. --config-file=/some/path/myTidyConfigFile

                                  This option internally works exactly the same way as

                                    --config option after reading specified config file.

                                  Use either --config-file or --config, not both.

                                  翻译：

                                  指定.clang-tidy或自定义配置文件的路径:

                                  例如,--config-file=/some/path/myTidyConfigFile

                                  此选项在内部的工作方式与读取指定的配置文件后选择--config选项一样。

                                  使用--config-file或--config中的任意一个，而不是两者都使用。

  --dump-config                  -

                                   Dumps configuration in the YAML format to

                                   stdout. This option can be used along with a

                                   file name (and '--' if the file is outside of a

                                   project with configured compilation database).

                                   The configuration used for this file will be

                                   printed.

                                   Use along with -checks=\* to include

                                   configuration of all checks.

                                   翻译：

                                   将YAML格式的配置转储到stdout。该选项可与文件名(如果文件位于

                                   配置了编译数据库的项目外面)。用于此文件的配置将被打印出来。

                                   与-checks=\*一起使用来包含所有检查的配置。

  --enable-check-profile         -

                                   Enable per-check timing profiles, and print a

                                   report to stderr.

                                   翻译：

                                   启用每次检查计时配置文件，并打印报告到stderr。

  --explain-config               -

                                   For each enabled check explains, where it is

                                   enabled, i.e. in clang-tidy binary, command

                                   line or a specific configuration file.

                                   翻译：

                                   对于每个启用的检查说明它的位置启用，也就是说，在clang-tidy的二进

                                   制文件中，命令行或特定的配置文件。

  --export-fixes=<filename>      -

                                   YAML file to store suggested fixes in. The

                                   stored fixes can be applied to the input source

                                   code with clang-apply-replacements.

                                   翻译：

                                   YAML文件来存储建议的修复。这些 存储的修正可以应用到输入的源代码中

  --extra-arg=<string>           - Additional argument to append to the compiler command line.

                                   Can be used several times.

                                   翻译：

                                   要追加到编译器命令行的附加参数。可以多次使用。

  --extra-arg-before=<string>    - Additional argument to prepend to the compiler command line.

                                   Can be used several times.

                                    翻译：

                                   要追加到编译器命令行的附加参数。可以多次使用。

  --fix                          -

                                   Apply suggested fixes. Without -fix-errors

                                   clang-tidy will bail out if any compilation

                                   errors were found.

                                   翻译：

                                   应用建议的修复。如果没有 -fix-errors clang-tidy会在发现任何编译

                                   错误时跳过。

  --fix-errors                   -

                                   Apply suggested fixes even if compilation

                                   errors were found. If compiler errors have

                                   attached fix-its, clang-tidy will apply them as

                                   well.

                                   翻译：

                                   即使发现编译错误，也要应用建议的修复方法。如果编译器错

                                   误有 附带的修复程序，clang-tidy也会应用。

  --fix-notes                    -

                                   If a warning has no fix, but a single fix can

                                   be found through an associated diagnostic note,

                                   apply the fix.

                                   Specifying this flag will implicitly enable the

                                   '--fix' flag.

                                   翻译：

                                   如果一个警告没有修复，但通过相关的诊断说明可以找到一个单一的修复，

                                   应用该修复。 指定这个标志将隐含地启用 '--fix'标志。

  --format-style=<string>        -

                                   Style for formatting code around applied fixes:

                                     - 'none' (default) turns off formatting

                                     - 'file' (literally 'file', not a placeholder)

                                       uses .clang-format file in the closest parent

                                       directory

                                     - '{ <json> }' specifies options inline, e.g.

                                       -format-style='{BasedOnStyle: llvm, IndentWidth: 8}'

                                     - 'llvm', 'google', 'webkit', 'mozilla'

                                   See clang-format documentation for the up-to-date

                                   information about formatting styles and options.

                                   This option overrides the 'FormatStyle` option in

                                   .clang-tidy file, if any.

  --header-filter=<string>       -

                                   Regular expression matching the names of the

                                   headers to output diagnostics from. Diagnostics

                                   from the main file of each translation unit are

                                   always displayed.

                                   Can be used together with -line-filter.

                                   This option overrides the 'HeaderFilterRegex'

                                   option in .clang-tidy file, if any.

                                   翻译：

                                   匹配头文件名称的正则表达式，用于输出诊断程序。每个翻译单元的主文件

                                   中的诊断总是显示出来。

                                   可与-line-filter一起使用。这个选项覆盖'HeaderFilterRegex'

                                   选项在.clang-tidy文件中，如果有的话。

  --line-filter=<string>         -

                                   List of files with line ranges to filter the

                                   warnings. Can be used together with

                                   -header-filter. The format of the list is a

                                   JSON array of objects:

                                     [

                                       {"name":"file1.cpp","lines":[[1,3],[5,7]]},

                                       {"name":"file2.h"}

                                     ]

                                    翻译：

                                    带有行数范围的文件列表，用于过滤警告。可以和 -header-filter一起

                                    用。该列表的格式是一个JSON数组的对象。

  --list-checks                  -

                                   List all enabled checks and exit. Use with

                                   -checks=\* to list all available checks.

                                   翻译：

                                   列出所有已启用的检查并退出。使用-checks=\*来列出所有可用的检查。

  -load=<plugin>                 -

                                   Load the dynamic object ``plugin``. This

                                   object should register new static analyzer

                                   or clang-tidy passes. Once loaded, the

                                   object will add new command line options

                                   to run various analyses. To see the new

                                   complete list of passes, use the

                                   :option:`--list-checks` and

                                   :option:`-load` options together.

                                   翻译：

                                   加载动态对象“插件”。这个对象应该注册新的静态分析器或clang-tidy通

                                   道。一旦加载，对象将添加新的命令行选项来运行各种分析。要查看新的完

                                   整的通过列表， 可以同时使用

                                   :option:`--list-checks`和

                                   :option:-load'选项。

  -p <string>                    - Build path

  --quiet                        -

                                   Run clang-tidy in quiet mode. This suppresses

                                   printing statistics about ignored warnings and

                                   warnings treated as errors if the respective

                                   options are specified.

                                   翻译：

                                   在安静模式下运行clang-tidy。如果指定了相应的选项，则禁止打印关于

                                   被忽略的警告和被视为错误的警告的统计信息。

  --store-check-profile=<prefix> -

                                   By default reports are printed in tabulated

                                   format to stderr. When this option is passed,

                                   these per-TU profiles are instead stored as JSON.

                                   翻译：

                                   默认情况下，报告以表格格式打印到stderr。当传递此选项时，这些每个

                                   TU的概要文件将被存储为JSON。

  --system-headers               - Display the errors from system headers.

  --use-color                    -

                                  Use colors in diagnostics. If not set, colors

                                  will be used if the terminal connected to

                                  standard output supports colors.

                                  This option overrides the 'UseColor' option in

                                  .clang-tidy file, if any.

                                  翻译：

                                  在诊断中使用颜色。如果没有设置，如果连接到标准输出的终端支持颜色，

                                  将使用颜色。

                                  这个选项覆盖clang-tidy文件中的'UseColor'选项(如果有的话)。

  --verify-config                -

                                   Check the config files to ensure each check and

                                   option is recognized.

                                   翻译：

                                   检查配置文件，确保每个检查和选项都被识别。

  --vfsoverlay=<filename>        -

                                   Overlay the virtual filesystem described by file

                                   over the real file system.

                                    翻译：

                                    将文件描述的虚拟文件系统覆盖在真实的文件系统上。

  --warnings-as-errors=<string>  -

                                   Upgrades warnings to errors. Same format as

                                   '-checks'.

                                   This option's value is appended to the value of

                                   the 'WarningsAsErrors' option in .clang-tidy

                                   file, if any.

                                   翻译：

                                   将警告升级为错误。格式与'-checks'相同。该选项的值被附加到

                                   .clang-tidy文件中的'WarningsAsErrors'选项的值(如果有的话)。

-p <build-path> is used to read a compile command database.

        For example, it can be a CMake build directory in which a file named

        compile\_commands.json exists (use -DCMAKE\_EXPORT\_COMPILE\_COMMANDS=ON

        CMake option to get this output). When no build path is specified,

        a search for compile\_commands.json will be attempted through all

        parent paths of the first input file . See:

        https://clang.llvm.org/docs/HowToSetupToolingForLLVM.html for an

        example of setting up Clang Tooling on a source tree.

        翻译：

        -p <build-path>用于读取编译命令数据库。

        例如，它可以是一个CMake构建目录，其中包含一个名为compile\_commands.json 的文件。

        (使用-DCMAKE\_EXPORT\_COMPILE\_COMMANDS=ON CMake选项来获得此输出)。

        当没有指定构建路径时，将尝试通过第一个输入文件的所有父路径搜索compile\_commands.json。

        有关在源代码树上设置Clang Tooling 的示例，

        请参见:https://clang.1lvm.org/docs/HowToSetupToolingForLLVM.html。

<source0> ... specify the paths of source files. These paths are

        looked up in the compile command database. If the path of a file is

        absolute, it needs to point into CMake's source tree. If the path is

        relative, the current working directory needs to be in the CMake

        source tree and the file must be in a subdirectory of the current

        working directory. "./" prefixes in the relative files will be

        automatically removed, but the rest of a relative path must be a

        suffix of a path in the compile command database.

        翻译:

        <source0>...指定源文件的路径。这些路径在编译命令数据库中查找。如果文件的路径是绝对的，它需要指向CMake的源树。如果路径是相对的，则当前工作目录需要在CMake源代码树中，而文件必须在当前工作目录的一个子路径。相对文件中的"./"前缀将被自动删除，但相对路径的其余部分必须是编译命令数据库中路径的后缀。

Configuration files:

  clang-tidy attempts to read configuration for each source file from a

  .clang-tidy file located in the closest parent directory of the source

  file. If InheritParentConfig is true in a config file, the configuration file

  in the parent directory (if any exists) will be taken and current config file

  will be applied on top of the parent one. If any configuration options have

  a corresponding command-line option, command-line option takes precedence.

  The effective configuration can be inspected using -dump-config:

翻译：

Clang-tidy尝试从位于源文件最近的父目录中读取每个源文件的配置的.Clang-tidy。如果配置文件中的InheritParentconfig为true，则父目录中的配置文件(如果存在的话)将被取走，并将当前配置文件应用到父目录之上。如果任何配置选项都有相应的命令行选项，则命令行选项优先。有效的配置可以使用-dump-config命令检查:

    $ clang-tidy -dump-config

    ---

    Checks:              '-\*,some-check'

    WarningsAsErrors:    ''

    HeaderFilterRegex:   ''

    FormatStyle:         none

    InheritParentConfig: true

    User:                user

    CheckOptions:

      some-check.SomeOption: 'some value'