CMake 基础使用(version >= 3.12)

前置知识

多文件编译

1. 单文件编译虽然简单，但是不利于模块化和项目的可读性
2. 只要改动一点就需要重新编译，如果工程项目十分庞大，所花费的时间也非常久

针对这种情况，提出了多文件编译的概念

```
g++ -c hello.c -o hello.o
g++ -c main.c -o main.o
g++ main.o hello.o -o a.out
```

-c 指定生成对应的object文件, 各个 `` `object file` `` 通过符号声明互相引用

动态链接库

动态链接库的是为了节省程序内存占用，多个程序共享同一个动态链接库，而不是每个程序都单独的加载一个动态链接库。 相当于在程序中生成一个插桩函数，程序执行过程中，当可执行文件被加载时会读取指定目录下的dll文件，加载到内存的指定目录，并且替换相应的插桩函数，指向的地址为加载后的地址。这个过程称为"重定向"。

动态库回去运行时查找，必须在同目录或者其他位置找到库文件才能执行 静态库不需要动态查找，但是生成的可执行文件会变大

静态链接库

静态链接库相当于直接把代码插入到可执行文件当中，这样做的后果就是会增大可执行文件的体积，但是只需要一个文件就能运行.

_

库查找

windows会现在当前目录下查找库，如果查找不到则会去PATH环境变量当中查找 Linux则会在RPath(*run-time search path*)当中查找，然后再去/usr/lib当中查找

为什么需要声明

1. 因为没有声明，函数重载，函数参数类型隐式转换等特性就没法儿支持 比如void add(float); 如果在代码中传入的是一个整数，那么编译器就知道把参数转换为成浮点数传入函数。
2. 让编译器知道使用的`add()`是一个函数而非实例化一个对象,这里的add有可能是一个class
   - `vector<MyClass> s` 这里的vector小于 MyClass 这个变量，大于s这个变量。编译器并不知道vector是个模板类

头文件

c++必须使用声明来保证

MakeFile是什么?

构建程序时，我们通常要将源码分成不同的模块进行构建，因为这样可以

1. 提高编译的速度（增量编译Incremental compiler）
2. 方便开发

安装CMake

Ubuntu

```
sudo apt install cmake
```

如果在windows上也可以使用`pip`安装

```
pip install  cmake
```

CMAKE 基础使用

创建库

在cmake中使用`add Libary`命令创建一个库

生成静态库

```
add_libary(hellolib STATIC hello.cpp)
```

动态库

```
add_libary(hellolib SHARED hello.cpp)
```

连接库

```
target_link_libraries(a.out PUBLIC hellolib)
```