

注意：本文档讲述的**不是**独立使用 Linux C++ 编译C++程序的方法，
而是将已经使用VS2022编写并调试完成的C++程序再用
Linux C++ 编译一遍



- ★ 用多个编译器完成同一个作业时，希望共用一个源程序文件，目的是避免同一个作业维护多个源程序文件所带来的冲突及错误
- ★ 单独使用Linux C++编写C++程序的方法请自行摸索
- ★ 为了统一，**强制要求**每个作业首先用VS2022完成，在调试通过的基础上用Dev C++及Linux C++再次编译，从而体验同一程序在不同编译器中可能出现的差异

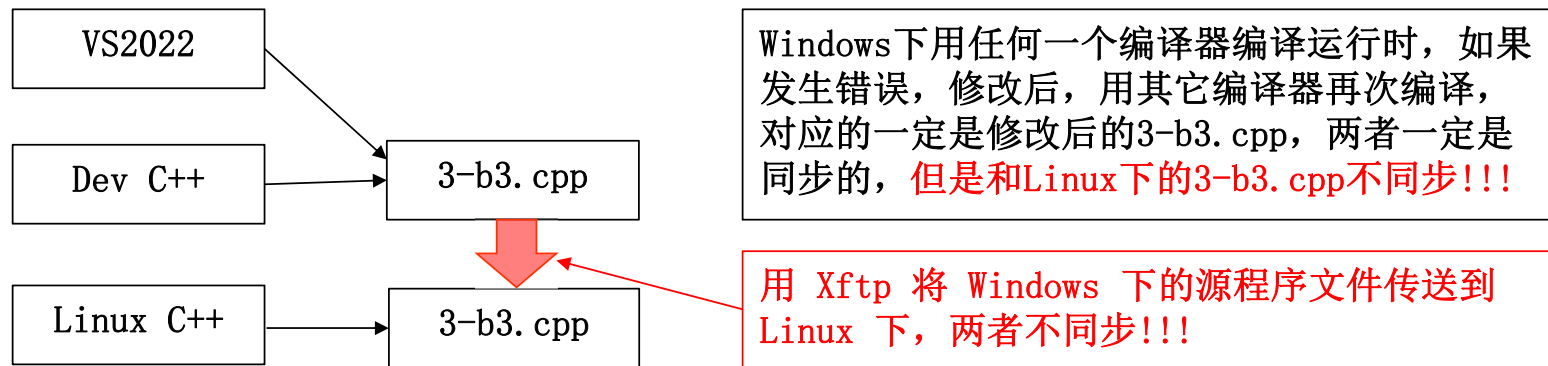


★ 用多个编译器完成同一个作业时，希望共用一个源程序文件，目的是避免同一个作业维护多个源程序文件所带来的冲突及错误

例：完成作业3-b3（对应源代码为 3-b3.cpp，要求能同时适应三个编译器）

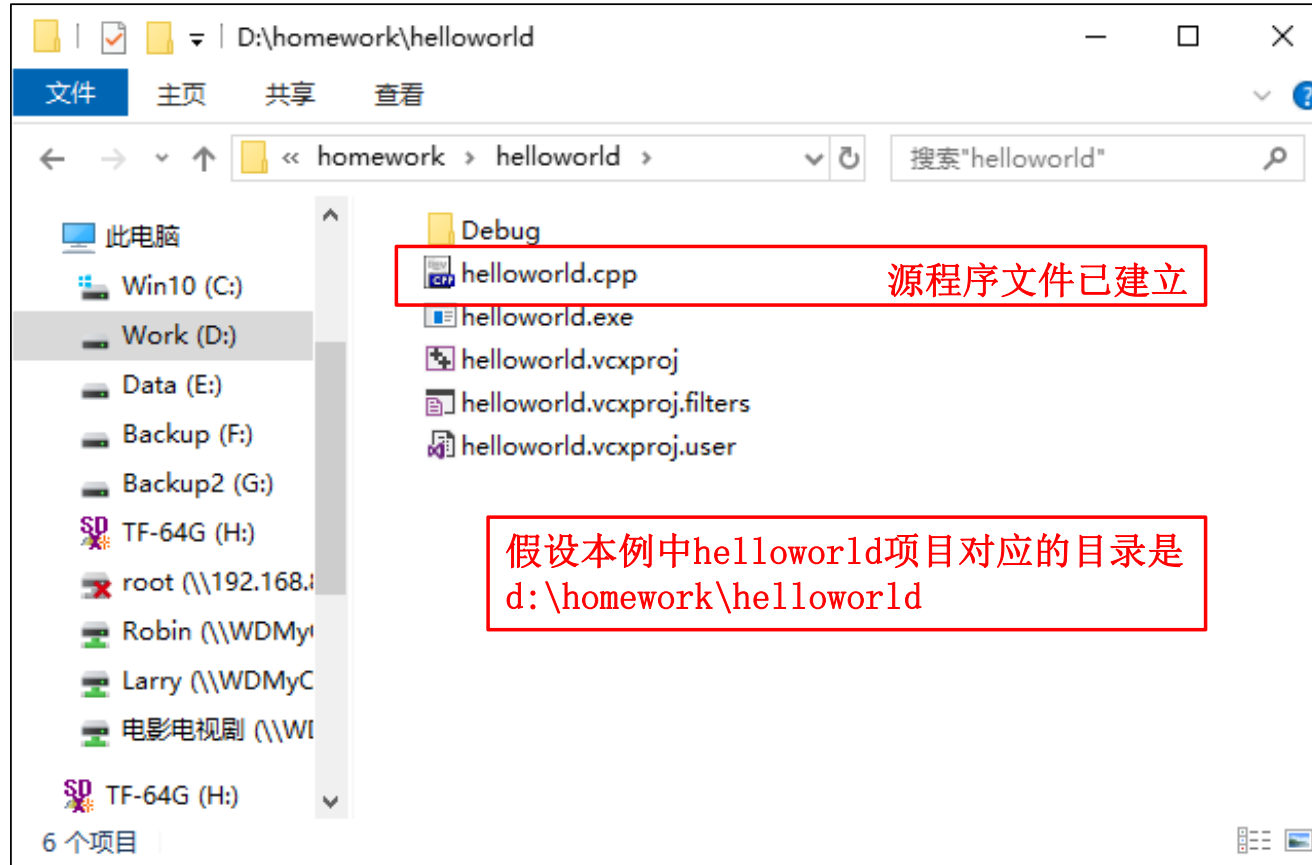
前提：Windows下的两个编译器对应同一个 3-b3.cpp，已验证完毕，现需要进行 Linux 下的验证

方法：用 Xftp 将 Windows 已验证好的 3-b3.cpp 上传到 Linux 服务器上，如果出现错误，则在 Windows 下修改，用 VS2022 + DevC++ 编译器验证正确后，再次上传至 Linux 服务器验证





第1步：假设在VS2022中已经建立了一个“helloworld”项目，对应的 helloworld.cpp 已经在VS2022下建立，并在 VS2022 和 Dev C++ 中已验证通过



第2步：用 Xshell 登录 Linux 服务器，再用 Xftp 将 helloworld.cpp 传到 Linux 服务器上

具体方法见本文档的前序文档即可





第3步：在Linux下编译刚才传输过来的helloworld.cpp文件并运行

```
1 oop服务器 x +
[D:\~]$

Connecting to 10.80.42.230:22...
Connection established.
To escape to local shell, press Ctrl+Alt+].

WARNING! The remote SSH server rejected X11 forwarding request.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Sep  7 18:08:09 2024 from 10.10.108.117

用户使用限制提示:
同时运行的程序/进程数量 <= 64
登录shell <= 3
每个程序打开文件数量 <= 64
每个程序可使用内存 <= 512 (MB)
每个程序占用CPU时间 <= 600 (sec)
用户可用磁盘空间 <= 1000M (MB) -- 可用 show-disk 查看当前已用的磁盘空间
无操作超时退出时间 = 900 (sec)

[ui1234567@oop ~]$ ls -l
总用量 4
-rw-r--r-- 1 ui1234567 stu 110 9月  7 18:01 helloworld.cpp
[ui1234567@oop ~]$
[ui1234567@oop ~]$ c++ -Wall -o helloworld helloworld.cpp
[ui1234567@oop ~]$
[ui1234567@oop ~]$ ls -l
总用量 28
-rwxr-xr-x 1 ui1234567 stu 80912 9月  7 18:10 helloworld
-rw-r--r-- 1 ui1234567 stu  110 9月  7 18:01 helloworld.cpp
[ui1234567@oop ~]$
[ui1234567@oop ~]$ ./helloworld
Hello, World!
[ui1234567@oop ~]$
[ui1234567@oop ~]$
```

①用 ls -l 确认helloworld.cpp 文件已传输过来

②将helloworld.cpp编译为可执行文件helloworld

③用 ls -l 确认可执行文件 helloworld 已生成

④用 ./helloworld 运行观察运行结果是否符合预期

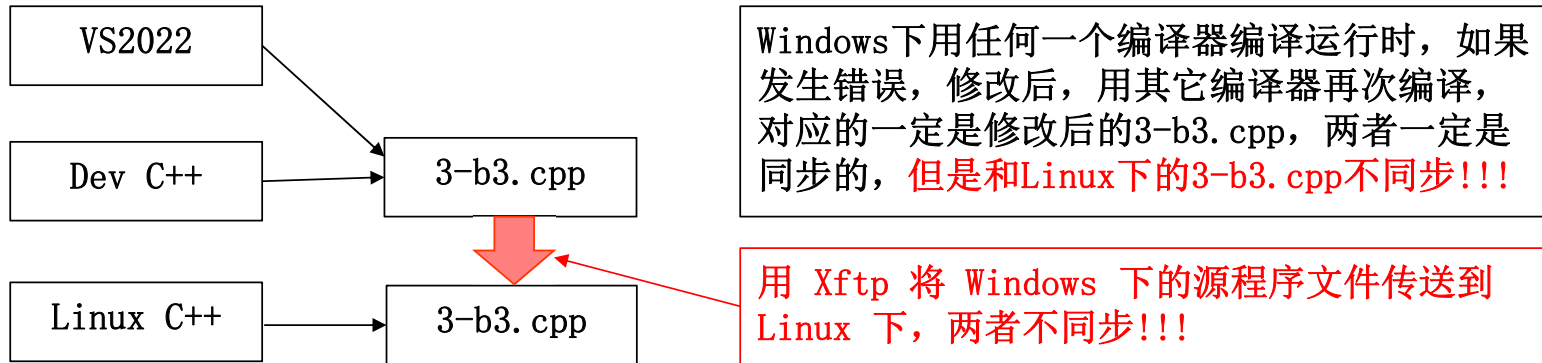
注意：1、四个红框代表从上到下四个步骤，依次进行
2、所有命令中均是字母l，不是数字1



说明:

★ 通过步骤1-3, 一个程序在Linux下编译并验证完成

★ **再次提醒**, 如果发生源程序修改, 一定要用 Xftp 再次上传并编译、运行





Linux下编译C/C++程序的基本方法:

★ 编译单源程序文件 (*.c) 的基本方法:

`gcc -Wall -o 可执行文件名 源程序文件名`

- 以helloworld.c为例, 编译命令为: `gcc -Wall -o helloworld helloworld.c`
- 不能写成 `gcc -Wall -o helloworld.c helloworld` (即不可以先源程序名再可执行文件名, 否则后果自行体会)

★ 编译单源程序文件 (*.cpp) 的基本方法:

`c++ -Wall -o 可执行文件名 源程序文件名`

- 以helloworld.cpp为例, 编译命令为: `c++ -Wall -o helloworld helloworld.cpp`
- 不能写成 `c++ -Wall -o helloworld.cpp helloworld` (即不可以先源程序名再可执行文件名, 否则后果自行体会)

★ 编译多源程序文件的基本方法:

`c++ -Wall -o 可执行文件名 源程序文件名1 ... 源程序文件名n`

- 上学期的4-b16为例 (求一元二次方程的根)

16、 题目及要求同 4-b14, 要求 main 函数在 4-b16-main.cpp 中, 其余 4 个函数分别放在 4-b16-sub1.cpp、4-b16-sub2.cpp、4-b16-sub3.cpp、4-b16-sub4.cpp 中, 四个函数的声明放在 4-b16.h 中, 以上六个文件共同生成可执行文件 (需要在 4-b16-main.cpp 中加入 `#include "4-b16.h"`)

编译命令为: `c++ -Wall -o 4-b16 4-b16-main.cpp 4-b16-sub1.cpp 4-b16-sub2.cpp 4-b16-sub3.cpp 4-b16-sub4.cpp`

- ◆ 五个cpp文件名在命令行中出现的顺序无限制
- ◆ 头文件(4-b16.h)不能出现在编译命令中

★ 编译若有错误, 则下面会出现错误提示, 如果正确, 则无提示

★ 如需修改源文件, 可以在VS2022/DevC++下修改并保存后再次用Xftp传输, 并再次编译(只要能保持同步, 其它方法也可以)

- Linux下编辑源程序文件 (*.cpp/*.c/*.h) 的方法本课程不做要求
- 学有余力的同学可自学vi/vim的简单操作 (掌握查找、替换、插入、删除、修改、存盘/不存盘退出的基本命令即可)



Linux下C/C++编译器与VS2022/DevC++在数据类型上的差异:

- 1、纯64位编译器，不能编译32位应用程序
- 2、因为是纯64位编译器，因此指针变量占8字节
- 3、long型和long long型都是8字节

```
[u1234567@oop ~]$ cat test.cpp
#include <iostream>
using namespace std;

int main()
{
    cout << sizeof(void *) << endl;
    cout << sizeof(long) << endl;
    cout << sizeof(long long) << endl;

    return 0;
}

[u1234567@oop ~]$ c++ -Wall -o test test.cpp
[u1234567@oop ~]$ 
[u1234567@oop ~]$ ./test
8
8
8
[u1234567@oop ~]$ 
[u1234567@oop ~]$
```