

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VS2019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VS2019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

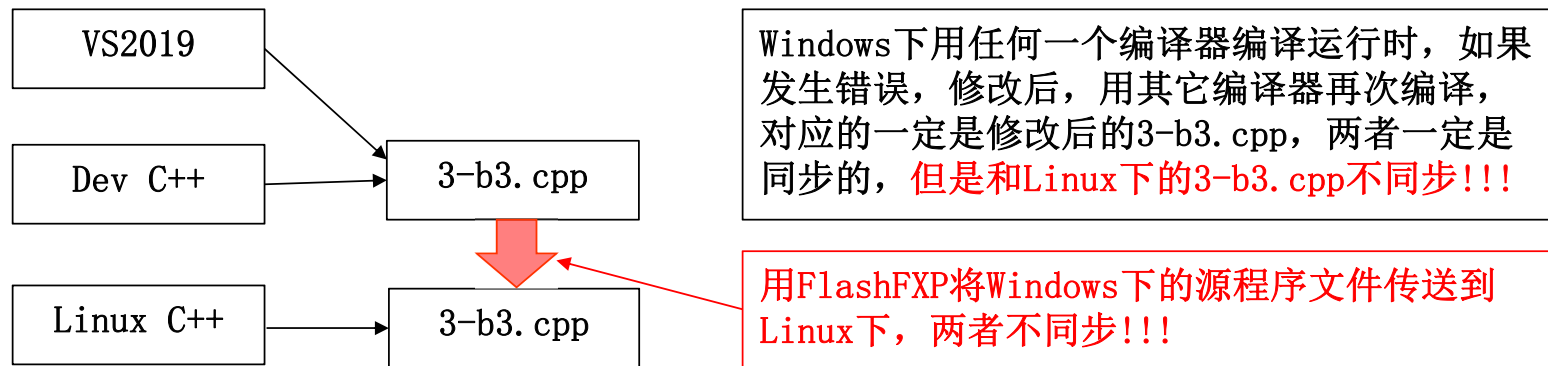


★ 用多个编译器完成同一个作业时，希望共用一个源程序文件，目的是避免同一个作业维护多个源程序文件所带来的冲突及错误

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）

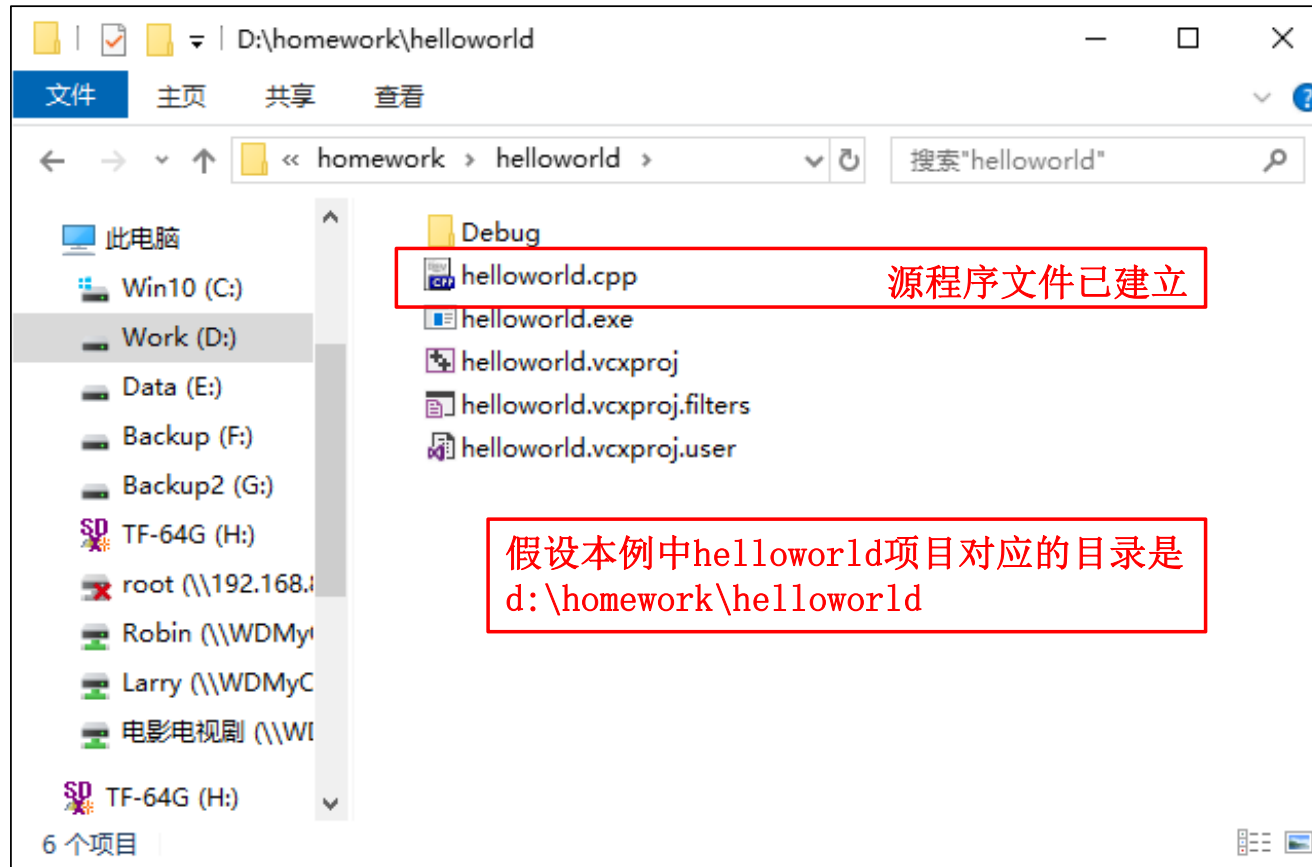
前提：Windows下的两个编译器对应同一个 3-b3.cpp，已验证完毕，现需要进行Linux下的验证

方法：用FlashFXP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 VS2019 + 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VS2019中已经建立了一个“helloworld”项目，对应的 helloworld.cpp 已经在VS2019下建立，并在 VS2019 和 Dev C++ 中已验证通过



第2步：用SecureCRT远程登录Linux服务器



```
校园网-10.60.102.252(高程-10022-u1234567) x
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Mar  7 03:01:45 2021 from 111.187.76.37

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 <= 256 (MB)
每个程序占用CPU时间 <= 600 (sec)
用户可用磁盘空间 <= 500 (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)

[u1234567@10074201 ~]$
```



第3步：启动FlashFXP，将 d:\homework\helloworld 文件夹下的 helloworld.cpp 复制到 Linux 的用户文件夹下（本例中是/home/u1234567）

FlashFXP - Linux-Server

本地浏览器

名称 大小 修改时间

名称	大小	修改时间
上级目录		
Debug		2019/9/3 17:07:16
helloworld.cpp	110	2019/9/3 17:04:50
helloworld.exe	1.36 MB	2019/9/3 18:00:21
helloworld.vcxproj	5 KB	2019/9/3 17:03:50
helloworld.vcxproj.filters	955	2019/9/3 17:03:50
helloworld.vcxproj.user	168	2019/9/3 17:00:25

1、Windows切换到指定目录

名称 大小 修改时间 属性

名称	大小	修改时间	属性
上级目录			
.cache	18	2019/9/3 19:59:54	drwxr-xr-x
.config	18	2019/9/3 19:59:54	drwxr-xr-x
.bash_logout	18	2018/10/31 1:07:12	-rw-r--r--
.bash_profile	193	2018/10/31 1:07:12	-rw-r--r--
.bashrc	231	2018/10/31 1:07:12	-rw-r--r--
helloworld.cpp	110	2019/9/3 17:04:51	-rw-r--r--

2、Linux切换到指定目录

3、从Windows端将需要的文件拖曳到Linux端

5 个文件, 1 个文件夹, 共计 6 项, 已选定 1 项 (110 字节)

本地浏览器

名称 目标 大小 备注

4 个文件, 2 个文件夹, 共计 6 项, 已选定 1 项 (110 字节)

Linux-Server

4、日志区：显示传输相关信息

```
[20:33:14] [R] SSH 连接打开
[20:33:15] [R] 已建立连接对象: OpenSSH_7.4 (SFTP v3)
[20:33:15] [R] SFTP 连接就绪
[20:33:15] [R] 目录更改进度: /home/u1234567/
[20:33:15] [R] 获取目录列表中.....
[20:33:15] [R] 列表完成: 710 字节 耗时 0.03 秒 (0.7 KB/s)
[20:33:37] [R] 正在上传: /home/u1234567/helloworld.cpp
[20:33:37] [R] 上传: helloworld.cpp 110 字节 耗时 0.03 秒 (0.1 KB/s)
[20:33:37] [R] 获取目录列表中.....
[20:33:37] [R] 列表完成: 822 字节 耗时 0.03 秒 (0.8 KB/s)
[20:33:37] [R] 传输队列已完成
[20:33:37] [R] 已传输 1 个文件 (110 字节) 耗时 0.06 秒 (0.1 KB/s)
空闲. (00:07)
```



第4步：在Linux下编译刚才传输过来的helloworld.cpp文件并运行

```
✓ 校园网-10.60.102.252(高程-10022-u1234567) x
无操作超时退出时间      = 900 (sec)

[u1234567@10074201 ~]$ ls -l
总用量 4
-rw-r--r-- 1 u1234567 stu 109 3月  7 03:31 helloworld.cpp

[u1234567@10074201 ~]$
[u1234567@10074201 ~]$ c++ -Wall -o helloworld helloworld.cpp
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$
[u1234567@10074201 ~]$ ls -l
总用量 20
-rwxr-xr-x 1 u1234567 stu 13408 3月  7 03:33 helloworld
-rw-r--r-- 1 u1234567 stu  109 3月  7 03:31 helloworld.cpp

[u1234567@10074201 ~]$
[u1234567@10074201 ~]$ ./helloworld
Hello, world!
[u1234567@10074201 ~]$
```

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

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

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



基本使用方法:

★ 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

★ 如果需要修改源程序文件, 可以在VS2019/DevC++下修改并保存后, 再次用FlashFXP传输过来, 并再次编译

- 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)



说明:

★ 通过步骤1-4, 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FlashFXP再次上传并编译、运行

