在Linux中将多源程序文件编译为一个可执行程序

★ 以VS2017中的ex_without_h项目为例(不包含头文件)
(包含ex1.cpp和ex2.cpp)

192.168.80.230(RHEL 7.4) ×

```
[root@RHEL74-X64 ex_without_h]# ls -l
总用量 13
drwxr-xr-x 2 root root    0 11月 23 21:30 Debug
-rwxr-xr-x 1 root root  118 11月 23 21:30 ex1.cpp
-rwxr-xr-x 1 root root   98 11月 23 21:30 ex2.cpp
-rwxr-xr-x 1 root root 5861 11月 23 21:30 ex_without_h.vcxproj
-rwxr-xr-x 1 root root 1031 11月 23 21:30 ex_without_h.vcxproj.filters
[root@RHEL74-X64 ex_without_h]#
[root@RHEL74-X64 ex_without_h]#
[root@RHEL74-X64 ex_without_h]# c++ -o ex_without_h ex1.cpp ex2.cpp
[root@RHEL74-X64 ex_without_h]#
[root@RHEL74-X64 ex_without_h]# ./ex_without_h
15
[root@RHEL74-X64 ex_without_h]#
[root@RHEL74-X64 ex_without_h]#
```

1、用mount或映射方法使ex1.cpp和ex2.cpp在Linux下可见
2、用 c++ -o 可执行文件名 源程序名1 ... 源程序名n
将n个源程序文件名编译为一个可执行文件
3、运行可执行文件，验证是否正确

在Linux中将多源程序文件编译为一个可执行程序

★ 以VS2017中的ex_with_h项目为例(包含头文件)
(包含ex.h、ex1.cpp和ex2.cpp)

192.168.80.230(RHEL 7.4) ×

```
[root@RHEL74-X64 ex_with_h]# ls -l
总用量 14
drwxr-xr-x 2 root root    0 11月 23 21:56 Debug
-rwxr-xr-x 1 root root  113 11月 23 21:42 ex3.cpp
-rwxr-xr-x 1 root root   98 11月 23 21:56 ex4.cpp
-rwxr-xr-x 1 root root   40 11月 23 21:56 ex.h
-rwxr-xr-x 1 root root 5923 11月 23 21:56 ex_with_h.vcxproj
-rwxr-xr-x 1 root root 1146 11月 23 21:56 ex_with_h.vcxproj.filters
[root@RHEL74-X64 ex_with_h]#
[root@RHEL74-X64 ex_with_h]# c++ -o ex_with_h ex3.cpp ex4.cpp
[root@RHEL74-X64 ex_with_h]#
[root@RHEL74-X64 ex_with_h]# ./ex_with_h
15
[root@RHEL74-X64 ex_with_h]#
[root@RHEL74-X64 ex_with_h]#
```

1、用mount或映射方法使ex.h、ex1.cpp和ex2.cpp在Linux下可见
2、用 c++ -o 可执行文件名 源程序名1 ... 源程序名n
将n个源程序文件名编译为一个可执行文件
(注：头文件不会出现在编译命令中，因此无论是否包含头文件，Linux下的编译命令相同)
3、运行可执行文件，验证是否正确