

GCC编译C/C++程序（一步完成）

[< 上一页](#)[下一页 >](#)

通过前面章节的学习我们知道，GCC 编译器并未提供给用户可用鼠标点击的界面窗口，要想调用 GCC 编译器编译 C 或者 C++ 程序，只能通过执行相应的 gcc 或者 g++ 指令。本节将重点给大家讲解如何编写 gcc 或者 g++ 指令来编译 C、C++ 程序。

注意，在前面的讲解中我们一直提到“编译”C、C++ 程序，其本意指的是将 C、C++ 代码转变为可执行程序（等同于 Windows 系统中以 .exe 为后缀的可执行文件）。但实际上，C 或者 C++ 程序从源代码生成可执行程序的过程，需经历 4 个过程，分别是预处理、编译、汇编和链接。有关这 4 个过程的具体含义，读者可阅读《[那些被编译器隐藏了的过程](#)》一节做详细了解，这里不再做重复赘述。

同样，使用 GCC 编译器编译 C 或者 C++ 程序，也必须要经历这 4 个过程。但考虑在实际使用中，用户可能并不关心程序的执行结果，只想快速得到最终的可执行程序，因此 gcc 和 g++ 都对此需求做了支持。

首先以运行 C 语言程序为例，给大家演示如何使用 gcc 快速获得对应的可执行程序。如下就是一段 C 语言程序：

```
01. //存储在 demo.c 文件中
02. #include <stdio.h>
03. int main() {
04.     puts("GCC教程: http://c.biancheng.net/gcc/");
05.     return 0;
06. }
```

如上所示，这是一个很简单的输出“Hello,World!”字符串的 C 语言程序，接下来打开命令行窗口（Terminal），编写如下 gcc 指令：

```
[root@bogon ~]# gcc demo.c
```

按下 Enter 回车键，由此 GCC 编译器就帮我们在当前目录下生成了对应的可执行文件，该文件的名称为 a.out，可以通过 ls 指令查看该文件是否存在：

```
[root@bogon ~]# ls
a.out  demo.c
```



#或许还有其他文件，这里不再一一列出

在此基础上，我们可以执行该文件，查看其执行结果，继续编写如下指令：

```
[root@bogon ~]# ./a.out
GCC教程: http://c.biancheng.net/gcc/
```

通过前面的学习我们知道，执行 C++ 程序和执行 C 语言程序不同的是，要么使用 g++ 指令，要么使用 gcc -xc++ -lstdc++ -shared-libgcc 指令。比如下面为一段简单的 C++ 程序：

```
01. //位于 demo.cpp 文件中
02. #include <iostream>
03. using namespace std;
04.
05. int main() {
06.     cout << "GCC教程: http://c.biancheng.net/gcc/" << endl;
07.     return 0;
08. }
```

运行此程序，可以编写如下指令并执行：

```
[root@bogon ~]# g++ demo.cpp #或者 gcc -xc++ -lstdc++ -shared-libgcc demo.cpp
```

同样，GCC 编译器会在当前目录下生成一个名为 a.out 的可执行文件（如果之前有同名文件，旧文件会被覆盖）。通过如下指令即可运行该文件：


```
[root@bogon ~]# ./a.out
GCC教程: http://c.biancheng.net/gcc/
```

注意，gcc 或者 g++ 指令还支持用户手动指定最终生成的可执行文件的文件名，例如修改前面执行 C、C++ 程序的 gcc 和 g++ 指令：

```
[root@bogon ~]# gcc demo.c -o demo.exe
[root@bogon ~]# g++ demo.cpp -o democpp.exe # 或者 gcc -xc++ -lstdc++ -shared-libgcc demo.cpp -o democpp.exe
```

其中 -o 选项用于指定要生成的文件名，例如 -o demo.exe 即表示将生成的文件名设为 demo.exe。

可以看到，GCC 编译器支持使用 gcc (g++) 指令“一步编译”指定的 C (C++) 程序。

注意，虽然我们仅编写了一条 gcc 或者 g++ 指令，但其底层依据是按照预处理、编译、汇编、接的过程将 C、C++ 程序转变为可执行程序。而本应在预处理阶段、编译阶段、汇编阶段生成

的中间文件，此执行方式默认是不会生成的，只会生成最终的 a.out 可执行文件（除非为 gcc 或者 g++ 额外添加 -save-temps 选项）。

对于初学者来说，可能需要深入探究 C、C++ 程序转变为可执行程序的全过程，查看该过程中产生的中间文件。如此，上面介绍的执行方式将不再使用，而要采用分步编译的方式。

所谓“分步编译”，即由用户手动调用 GCC 编译器完成对 C、C++ 源代码的预处理、编译、汇编以及链接过程，每个阶段都会生成对源代码加工后的文件。

那么，到底如何分步编译 C、C++ 程序呢？事实上，GCC 编译器除了提供 gcc 和 g++ 这 2 个指令之外，还提供有大量的指令选项，方便用户根据自己的需求自定义编译方式。在前面的学习过程中，我们已经使用了一些指令选项，比如编译 C++ 程序时 gcc 指令后跟的 -xc++、-lstdc++、-shared-libgcc，再比如手动指定可执行文件名称的 -o 选项。

表 1 罗列出了实际使用 gcc 或者 g++ 指令编译 C/C++ 程序时，常用的一些指令选项：

表 1 GCC常用的编译选项	
gcc/g++指令选项	功 能
-E（大写）	预处理指定的源文件，不进行编译。
-S（大写）	编译指定的源文件，但是不进行汇编。
-c	编译、汇编指定的源文件，但是不进行链接。
-o	指定生成文件的文件名。
-llibrary（-l library）	其中 library 表示要搜索的库文件的名称。该选项用于手动指定链接环节中程序可以调用的库文件。建议 -l 和库文件名之间不使用空格，比如 -lstdc++。
-ansi	对于 C 语言程序来说，其等价于 -std=c90；对于 C++ 程序来说，其等价于 -std=c++98。
-std=	手动指令编程语言所遵循的标准，例如 c89、c90、c++98、c++11 等。

注意，表 1 中仅列出了初学者常用的一些指令选项，事实上这仅是冰山一角，GCC 编译器提供有大量的指令选项，可满足我们在大部分场景下的编译需求。有关更多的编译指令，感兴趣的读者可自行查看 [GCC 手册](#)。

在表 1 的基础上，接下来将分章节对如何实现分步编译做详细的讲解，即如何将一个源代码程序经历预处理、编译、汇编以及链接这 4 个过程，最终生成对应的可执行程序。