gcc指令一次处理多个文件

< 上一页 下一页 >

通过前面几节的学习，读者已经了解了如何使用 gcc（g++）指令调用 GCC 编译器编译（包括预处理、编译、汇编和链接）C 或者 C++ 源代码，例如：

```
[root@bogon demo]# ls
demo1.c  demo2.c
[root@bogon demo]# cat demo1.c
#include<stdio.h>
int main(){
    printf("GCC:http://c.biancheng.net/gcc/");
    return 0;
}
[root@bogon demo]# cat demo2.c
#include<stdio.h>
int main(){
    printf("Python:http://c.biancheng.net/python/");
    return 0;
}
[root@bogon demo]# gcc -c demo1.c -o demo1.o
[root@bogon demo]# gcc -c demo2.c -o demo2.o
[root@bogon demo]# ls
demo1.c  demo1.o  demo2.c  demo2.o
```

如上所示，我们创建了 2 个 C 语言源程序文件，分别为 demo1.c 和 demo2.c，并在此基础上分别对它们执行 gcc -c 指令，使得 GCC 编译器先后对 demo1.c、demo2.c 进行了编译，并生成了各自的目标文件。

实际上，一条 gcc（g++）指令往往可以一次性处理多个文件。仍以编译 demo1.c 和 demo2.c 为例，可以执行如下指令：

```
[root@bogon demo]# gcc -c demo1.c demo2.c
[root@bogon demo]# ls
demo1.c  demo1.o  demo2.c  demo2.o
```

可以看到，demo1.c 和 demo2.c 的编译过程可以共用一条 gcc -c 指令，其默认情况下会分别生成 demo1.o 和 demo2.o 目标文件。

需要注意的是，此方法无法使用 -o 选项分别将编译 demo1.c 和 demo2.c 的目标代码输出到指定文件，也就是说如下这行代码是错误的：

```
[root@bogon demo]# gcc -c demo1.c demo2.c -o demo1.o demo2.o
gcc: demo2.o: No such file or directory
gcc: cannot specify -o with -c or -S with multiple files
```

显然，gcc 指令并没有我们想象的那么聪明。

不仅如此，以下这些操作都可以共用一条 gcc 指令：

- 将多个 C（C++）源文件加工为汇编文件或者目标文件；
- 将多个 C（C++）源文件或者预处理文件加工为汇编文件或者目标文件；
- 将多个 C（C++）源文件、预处理文件或者汇编文件加工为目标文件；
- 同一项目中，不同的源文件、预处理文件、汇编文件以及目标文件，可以使用一条 gcc 指令，最终生成一个可执行文件。

> 注意，多个 C（C++）源文件也可以使用一条 gcc -E 指令完成预处理操作，但由于该指令默认情况下只会将预处理结果输出到屏幕上，因此预处理操作虽然可以完成，但无法生成各自对应的预处理文件。

仍以 demo1.c 和 demo2.c 源程序文件为例：

```
[root@bogon demo]# gcc -E demo1.c -o demo1.i
[root@bogon demo]# ls
demo1.c  demo1.i  demo2.c
[root@bogon demo]# gcc -c demo1.i demo2.c
[root@bogon demo]# ls
demo1.c  demo1.i  demo1.o  demo2.c  demo2.o
```

可以看到，首先单独将 demo1.c 源文件做预处理操作，并生成 demo1.i 文件。在此基础上，我们仅使用一条 gcc -c 指令，同时将 demo1.i 和 demo2.c 各自编译为 demo1.o 和 demo2.o 目标文件。

GCC编译多文件项目

在一个 C（或者 C++）项目中，往往在存储多个源文件，如果仍按照之前“先单独编译各个源文件，再将它们链接起来”的方法编译该项目，需要编写大量的编译指令，事倍功半。事实上，利用 gcc 指令可以同时处理多个文件的特性，可以大大提高我们的工作效率。

举个例子，如下是一个拥有 2 个源文件的 C 语言项目：

```
[root@bogon demo]# ls
main.c  myfun.c
[root@bogon demo]# cat main.c
#include <stdio.h>
int main(){
    display();
    return 0;
}
[root@bogon demo]# cat myfun.c
#include <stdio.h>
void display(){
    printf("GCC:http://c.biancheng.net/gcc/");
}
[root@bogon demo]#
```

可以看到，该项目中仅包含 2 个源文件，其中 myfun.c 文件用于存储一些功能函数，以方便直接在 main.c 文件中调用。

对于此项目，我们可以这样编译：

```
[root@bogon demo]# ls
main.c  myfun.c
[root@bogon demo]# gcc -c myfun.c main.c
[root@bogon demo]# ls
main.c  main.o  myfun.c  myfun.o
[root@bogon demo]# gcc myfun.o main.o -o main.exe
[root@bogon demo]# ls
main.c  main.exe  main.o  myfun.c  myfun.o
[root@bogon demo]# ./main.exe
GCC:http://c.biancheng.net/gcc/
```

甚至于，gcc 指令还可以直接编译并链接它们：

```
[root@bogon demo]# gcc myfun.c main.c -o main.exe
[root@bogon demo]# ls
main.c  main.exe  myfun.c
[root@bogon demo]# ./main.exe
GCC:http://c.biancheng.net/gcc/
```

以上 2 种方式已然可以满足大部分场景的需要。但值得一提的是，如果一个项目中有十几个甚至十个源文件，即便共用一条 gcc 指令编译（并链接），编写各个文件的名称也是一件麻烦事。

为了解决这个问题，我们可以进入该项目目录，用 *.c 表示所有的源文件，即执行如下指令：

```
[root@bogon demo]# ls
main.c  myfun.c
[root@bogon demo]# gcc *.c -o main.exe
[root@bogon demo]# ls
main.c  main.exe  myfun.c
[root@bogon demo]# ./main.exe
GCC:http://c.biancheng.net/gcc/
```

由此，大大节省了手动输入各源文件名称的时间。

< 上一页 下一页 >

所有教程

算法 Python爬虫 C语言入门 C语言编译器 C语言项目案例 数据结构
多线程 链接库 C++ STL C++11 socket GCC GDB Makefile
OpenCV Qt教程 Unity 3D UE4 游戏引擎 Python Python并发编程
TensorFlow Django NumPy Linux Shell Java教程 设计模式
Java Swing Servlet教程 JSP教程 JSTL Struts2 Maven Nexus
Spring Spring MVC Spring Boot Spring Cloud Hibernate Mybatis
MySQL教程 MySQL函数 NoSQL Redis常用命令手册 HBase MongoDB
Go语言 C# MATLAB JavaScript Bootstrap HTML CSS PHP
汇编语言 TCP/IP vi命令 Android教程 区块链 Docker 大数据
云计算 推荐阅读 编程笔记 资源下载 VIP视频 一对一答疑 关于我们