<https://www.cnblogs.com/xj626852095/p/3648246.html>

gcc是C编译器；g++是C++编译器；linux下cc一般是一个符号连接，指向gcc；gcc和g++都是GUN（组织）的编译器。而CC则一般是makefile里面的一个名字，即宏定义，嘿，因为Linux/Unix都是大小写敏感的系统，这点一定要注意。

cc是Unix系统的C Compiler，而gcc则是GNU Compiler Collection，GNU编译器套装。gcc原名为Gun C语言编译器，因为它原本只能处理C语言，但gcc很快地扩展，包含很多编译器（C、C++、Objective-C、Ada、Fortran、Java）。因此，它们是不一样的，一个是古老的C编译器，一个是GNU编译器集合，gcc里面的C编译器比cc强大多了，因此没必要用cc。

下载不到cc的原因在于：cc来自于昂贵的Unix系统，cc是商业软件。

Linux下的cc是gcc符号连接，可以通过$ls –l /usr/bin/cc来简单察看，该变量是make程序的内建变量，默认指向gcc。cc符号链接和变量存在的意义在于源码的移植性，可以方便的用gcc来编译老的用cc编译的Unix软件，甚至连makefile都不用改在，而且也便于Linux程序在Unix下编译。

误区一：gcc只能编译C代码，g++只能编译C++代码。

两者都可以，但请注意：

（1）后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是C++程序，注意，虽然C++是C的超集，但是两者对语法的要求是有区别的。C++的语法规则更加严谨一些。

（2）编译阶段，g++会调用gcc，对于C++代码，两者是等价的，但是因为gcc命令不能自动和C++程序使用的库联接，所以通常用g++来完成链接，为了统一起见，干脆编译/链接统统用g++了，这就给人一种错觉，好像cpp程序只能用g++似的。

误区二：gcc不会定义\_\_cplusplus宏，而g++会

实际上，这个宏只是标志着编译器将会把代码按C还是C++语法来解释，如上所述，如果后缀为.c，并且采用gcc编译器，则该宏就是未定义的，否则，就是已定义。

误区三：编译只能用gcc，链接只能用g++

严格来说，这句话不算错误，但是它混淆了概念，应该这样说：编译可以用gcc/g++，而链接可以用g++或者gcc -lstdc++。因为gcc命令不能自动和C++程序使用的库联接，所以通常使用g++来完成联接。但在编译阶段，g++会自动调用gcc，二者等价。

C++的编译器肯定可以编译C的代码，注意除了C++对C的语法扩充之外，编译和链接C和C++的标准库通常也不一样呢，用gcc而非g++也编译了C++的程序就证明了这一点。

注：符号链接是一种特殊类型的文件，它的内容只是一个字符串。它可能指向一个存在的文件也可能什么都不指向。当您在命令行或程序里提到符号链接的时候，您实际上进入了它指向的文件，前提是这个文件是存在的。

用法：gcc [选项] 文件...

选项：

-pass-exit-codes 在某一阶段退出时返回最高的错误码

--help 显示此帮助说明

--target-help 显示目标机器特定的命令行选项

(使用‘-v --help’显示子进程的命令行参数)

-dumpspecs 显示所有内建 spec 字符串

-dumpversion 显示编译器的版本号

-dumpmachine 显示编译器的目标处理器

-print-search-dirs 显示编译器的搜索路径

-print-libgcc-file-name 显示编译器伴随库的名称

-print-file-name=<库> 显示 <库> 的完整路径

-print-prog-name=<程序> 显示编译器组件 <程序> 的完整路径

-print-multi-directory 显示不同版本 libgcc 的根目录

-print-multi-lib 显示命令行选项和多个版本库搜索路径间的映射

-print-multi-os-directory 显示操作系统库的相对路径

-Wa,<选项> 将逗号分隔的 <选项> 传递给汇编器

-Wp,<选项> 将逗号分隔的 <选项> 传递给预处理器

-Wl,<选项> 将逗号分隔的 <选项> 传递给链接器

-Xassembler <参数> 将 <参数> 传递给汇编器

-Xpreprocessor <参数> 将 <参数> 传递给预处理器

-Xlinker <参数> 将 <参数> 传递给链接器

-combine 将多个源文件一次性传递给汇编器

-save-temps 不删除中间文件

-pipe 使用管道代替临时文件

-time 为每个子进程计时

-specs=<文件> 用 <文件> 的内容覆盖内建的 specs 文件

-std=<标准> 指定输入源文件遵循的标准

--sysroot=<目录> 将 <目录> 作为头文件和库文件的根目录

-B <目录> 将 <目录> 添加到编译器的搜索路径中

-b <机器> 为 gcc 指定目标机器(如果有安装)

-V <版本> 运行指定版本的 gcc(如果有安装)

-v 显示编译器调用的程序

-### 与 -v 类似，但选项被引号括住，并且不执行命令

-E 仅作预处理，不进行编译、汇编和链接

-S 编译到汇编语言，不进行汇编和链接

-c 编译、汇编到目标代码，不进行链接

-o <文件> 输出到 <文件>

-x <语言> 指定其后输入文件的语言

允许的语言包括：c c++ assembler none

‘none’意味着恢复默认行为，即根据文件的扩展名猜测

源文件的语言

以 -g、-f、-m、-O、-W 或 --param 开头的选项将由 gcc 自动传递给其调用的

不同子进程。若要向这些进程传递其他选项，必须使用 -W<字母> 选项。