**Gcc的编译流程分为了四个步骤:**



[坡后村](https://me.csdn.net/xiaohouye) 2016-08-01 13:11:53  50123  收藏 7

分类专栏： [GCC](https://blog.csdn.net/xiaohouye/category_6297104.html)

Gcc的编译流程分为了四个步骤:

    1.预处理，生成预编译文件（.文件）：

        gcc –E hello.c –o hello.i  
    2.编译，生成汇编代码（.s文件）：

        gcc –S hello.i –o hello.s  
    3.汇编，生成目标文件（.o文件）：  
        gcc –c hello.s –o hello.o  
    4.链接，生成可执行文件：  
        gcc hello.o –o hello

    在成功编译之后，就进入了链接阶段。在这里涉及到一个重要的概念：函数库。

读者可以重新查看这个小程序，在这个程序中并没有定义”printf”的函数实现，且在预编译中包含进的”stdio.h”中也只有该函数的声明，而没有定义函数的实现，那么，是在哪里实现”printf”函数的呢？最后的答案是：系统把这些函数实现都被做到名为libc.so.6的库文件中去了，在没有特别指定时，Gcc会到系统默认的搜索路径”/usr/lib”下进行查找，也就是链接到libc.so.6库函数中去，这样就能实现函数”printf”了，而这也就是链接的作用。

    函数库一般分为静态库和动态库两种。静态库是指编译链接时，把库文件的代码全部加入到可执行文件中，因此生成的文件比较大，但在运行时也就不再需要库文件了。其后缀名一般为”.a”。动态库与之相反，在编译链接时并没有把库文件的代码加入到可执行文件中，而是在程序执行时由运行时链接文件加载库，这样可以节省系统的开销。动态库一般后缀名为”.so”，如前面所述的libc.so.6就是动态库。Gcc在编译时默认使用动态库。

    整个过程如果想一步到位：

       gcc hello.c -o hello

即可

    gcc简介  
    Linux系统下的gcc（GNU C Compiler）是GNU推出的功能强大、性能优越的多平台编译器，是GNU的代表作品之一。gcc是可以在多种硬体平台上编译出可执行程序的超级编译器，其执行效率与一般的编译器相比平均效率要高20%~30%。gcc编译器能将C、C++语言源程序、汇程式化序和目标程序编译、连接成可执行文件，如果没有给出可执行文件的名字，gcc将生成一个名为a.out的文件。在Linux系统中，可执行文件没有统一的后缀，系统从文件的属性来区分可执行文件和不可执行文件。而gcc则通过后缀来区别输入文件的类别，下面我们来介绍gcc所遵循的部分约定规则。  
    .c为后缀的文件，C语言源代码文件；  
    .a为后缀的文件，是由目标文件构成的档案库文件；  
    .C，.cc或.cxx 为后缀的文件，是C++源代码文件；  
    .h为后缀的文件，是程序所包含的头文件；  
    .i 为后缀的文件，是已经预处理过的C源代码文件；  
    .ii为后缀的文件，是已经预处理过的C++源代码文件；  
    .m为后缀的文件，是Objective-C源代码文件；  
    .o为后缀的文件，是编译后的目标文件；  
    .s为后缀的文件，是汇编语言源代码文件；  
    .S为后缀的文件，是经过预编译的汇编语言源代码文件。  
  
    gcc的执行过程  
    虽然我们称gcc是C语言的编译器，但使用gcc由C语言源代码文件生成可执行文件的过程不仅仅是编译的过程，而是要经历四个相互关联的步骤∶预处理(也称预编译，Preprocessing)、编译(Compilation)、汇编(Assembly)和连接(Linking)。命令gcc首先调用cpp进行预处理，在预处理过程中，对源代码文件中的文件包含(include)、预编译语句(如宏定义define等)进行分析。接着调用cc1进行编译，这个阶段根据输入文件生成以.o为后缀的目标文件。汇编过程是针对汇编语言的步骤，调用as进行工作，一般来讲，.S为后缀的汇编语言源代码文件和汇编、.s为后缀的汇编语言文件经过预编译和汇编之后都生成以.o为后缀的目标文件。当所有的目标文件都生成之后，gcc就调用ld来完成最后的关键性工作，这个阶段就是连接。在连接阶段，所有的目标文件被安排在可执行程序中的恰当的位置，同时，该程序所调用到的库函数也从各自所在的档案库中连到合适的地方。  
  
    gcc的基本用法和选项  
    在使用gcc编译器的时候，我们必须给出一系列必要的调用参数和文件名称。gcc编译器的调用参数大约有100多个，其中多数参数我们可能根本就用不到，这里只介绍其中最基本、最常用的参数。  
    gcc最基本的用法是∶gcc [options] [filenames]  
    其中options就是编译器所需要的参数，filenames给出相关的文件名称。  
    -c，只编译，不连接成为可执行文件，编译器只是由输入的.c等源代码文件生成.o为后缀的目标文件，通常用于编译不包含主程序的子程序文件。  
    -o output\_filename，确定输出文件的名称为output\_filename，同时这个名称不能和源文件同名。如果不给出这个选项，gcc就给出预设的可执行文件a.out。  
    -g，产生符号调试工具(GNU的gdb)所必要的符号资讯，要想对源代码进行调试，我们就必须加入这个选项。  
    -O，对程序进行优化编译、连接，采用这个选项，整个源代码会在编译、连接过程中进行优化处理，这样产生的可执行文件的执行效率可以提高，但是，编译、连接的速度就相应地要慢一些。  
    -O2，比-O更好的优化编译、连接，当然整个编译、连接过程会更慢。  
    -Idirname，将dirname所指出的目录加入到程序头文件目录列表中，是在预编译过程中使用的参数。  
    C程序中的头文件包含两种情况∶  
    A)#include  
    B)#include “myinc.h”  
    其中，A类使用尖括号(< >)，B类使用双引号(“ ”)。对于A类，预处理程序cpp在系统预设包含文件目录(如/usr/include)中搜寻相应的文件，而对于B类，cpp在当前目录中搜寻头文件，这个选项的作用是告诉cpp，如果在当前目录中没有找到需要的文件，就到指定的dirname目录中去寻找。在程序设计中，如果我们需要的这种包含文件分别分布在不同的目录中，就需要逐个使用-I选项给出搜索路径。  
    -Ldirname，将dirname所指出的目录加入到程序函数档案库文件的目录列表中，是在连接过程中使用的参数。在预设状态下，连接程序ld在系统的预设路径中(如/usr/lib)寻找所需要的档案库文件，这个选项告诉连接程序，首先到-L指定的目录中去寻找，然后到系统预设路径中寻找，如果函数库存放在多个目录下，就需要依次使用这个选项，给出相应的存放目录。-lname，在连接时，装载名字为“libname.a”的函数库，该函数库位于系统预设的目录或者由-L选项确定的目录下。例如，-lm表示连接名为“libm.a”的数学函数库。上面我们简要介绍了gcc编译器最常用的功能和主要参数选项，更为详尽的资料可以参看Linux系统的联机帮助。  
假定我们有一个程序名为test.c的C语言源代码文件，要生成一个可执行文件，最简单的  
办法就是∶  
    gcc test.c  
    这时，预编译、编译连接一次完成，生成一个系统预设的名为a.out的可执行文件，对于稍为复杂的情况，比如有多个源代码文件、需要连接档案库或者有其他比较特别的要求，就要给定适当的调用选项参数。再看一个简单的例子。整个源代码程序由两个文件testmain.c 和testsub.c组成，程序中使用了系统提供的数学库，同时希望给出的可执行文件为test，这时的编译命令可以是∶  
    gcc testmain.c testsub.c -lm -o test  
    其中，-lm表示连接系统的数学库libm.a，这个过程可以用图12-1框图描述。