**[Linux编译工具：gcc入门](https://link.zhihu.com/?target=https%3A//www.cnblogs.com/QG-whz/p/5456720.html" \t "_blank)**

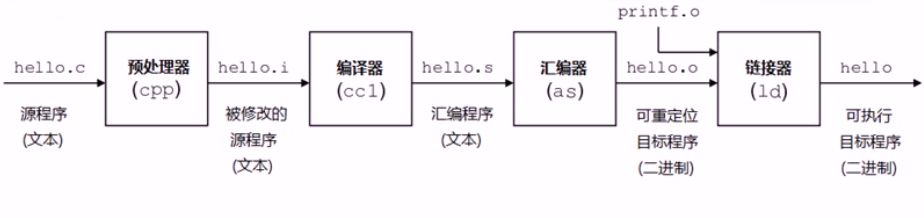
**1. 什么是gcc**  
gcc的全称是GNU Compiler Collection，它是一个能够编译多种语言的编译器。最开始gcc是作为C语言的编译器（GNU C Compiler），现在除了c语言，还支持C++、java、Pascal等语言。gcc支持多种硬件平台。

**2. gcc的特点**

* 略

**3. gcc编译程序的过程**  
gcc编译程序主要经过四个过程：

* 预处理（Pre-Processing）
* 编译 （Compiling）
* 汇编 （Assembling）
* 链接 （Linking）



gcc实际上是个工具链，在编译程序的过程中调用不同的工具。  
1、预编译要干的事情：如删除注释、替换宏*（比如定义#define pi 3.1415926，预处理会将程序里所有的pi换成3.1415926）*、递归展开头文件*（比如#include指令使另外一个文件被编译：预处理器先删除这条指令，并用包含文件的内容替换。这样一个文件被包含10次，那就实际被编译10次。）*、处理以#开头的预编译指令等，在预编译阶段不做任何有效信息的类型检查。（摆好位置待处理）

2、编译阶段，gcc调用不同语言的编译器，词法分析、语法分析和语义分析、代码的优化、编译、汇总所有的符号，例如c语言调用编译器ccl转成汇编语言。（查错、优化）

3、汇编阶段，gcc调用汇编器进行汇编，将汇编指令转换为特定平台下的机器语言、构建\*.o文件（obj文件）组成格式。（汇编语言转换机器语言）

4、链接器

将所有的目标文件和依赖的库文件进行汇总，得到最终的可执行程序。

**4. gcc常用选项**  
来看一下gcc常用选项

-c

　 只激活预处理,编译,和汇编,也就是他只把程序做成obj文件

　　例子用法:

　　gcc -c hello.c

　　他将生成.o的obj文件

-S

　 只激活预处理和编译，就是指把文件编译成为汇编代码。

　　例子用法

　　gcc -S hello.c

　　他将生成.s的汇编代码，你可以用文本编辑器察看

-E

　 只激活预处理,这个不生成文件,你需要把它重定向到一个输出文件里面.

　　例子用法:

　　gcc -E hello.c > pianoapan.txt

　　gcc -E hello.c | more

　　慢慢看吧,一个hello word 也要与处理成800行的代码

-o

　　指定目标名称,缺省的时候,gcc 编译出来的文件是a.out

　　例子用法

　　 gcc -o hello.asm -S hello.c

-pipe

　　使用管道代替编译中临时文件,在使用非gnu汇编工具的时候,可能有些问题

　　 gcc -pipe -o hello.exe hello.c-g 生成调试信息。GNU 调试器可利用该信息。

-C

在预处理的时候,不删除注释信息,一般和-E使用,有时候分析程序，用这个很方便的

-m486 针对 486 进行代码优化。

-O0 不进行优化处理。

-O 或 -O1 优化生成代码。

-O2 进一步优化。

-O3 比 -O2 更进一步优化，包括 inline 函数。

-w 不生成任何警告信息。

-Wall 生成所有警告信息。

参考文献 https://zhuanlan.zhihu.com/p/347611674

现在我们有源文件hello.c，下面是一些gcc的使用示例：

gcc -E hello.c -o hello.i 对hello.c文件进行预处理，生成hello.i 文件

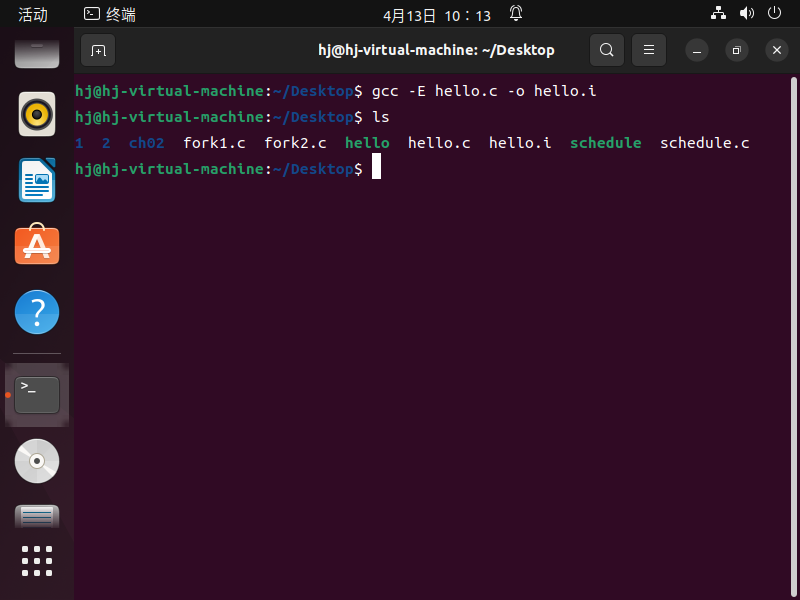
gcc -S hello.i -o hello.s 对预处理文件进行编译，生成了汇编文件

gcc -c hello.s -o hello.o 对汇编文件进行编译，生成了目标文件

gcc hello.o -o hello 对目标文件进行链接，生成可执行文件

gcc hello.c -o hello 直接编译链接成可执行目标文件

gcc -c hello.c 或 gcc -c hello.c -o hello.o 编译生成obj目标文件

  
使用gcc时可以加上-Wall选项。下面这个例子如果不加上-Wall选项，编译器不会报出任何错误或警告，但是程序的结果却不是预期的：

//bad.c

#include<stdio.h>

int main()

{

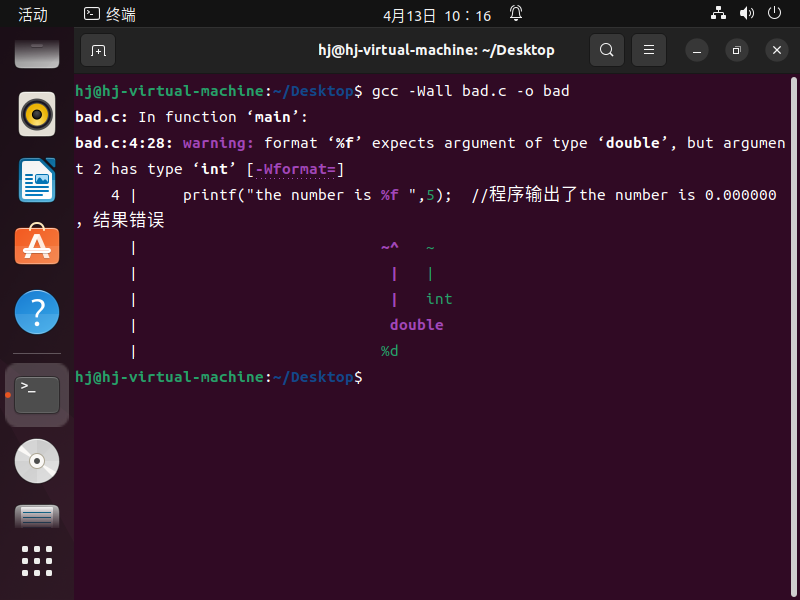
printf("the number is %f ",5); //程序输出了the number is 0.000000，结果错误

return 0;

}

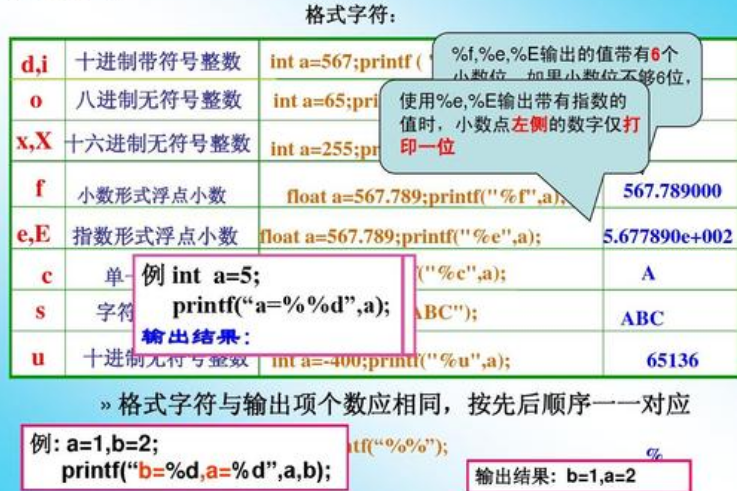
使用-Wall选项：

gcc -Wall bad.c -o bad

  
gcc将输出警告信息：

格式 '%f' 需要类型为“double”的参数，但参数 2 的类型为 'int'

warning: format ‘%f’ expects argument of type ‘double’, but argument 2 has type ‘int’ [-Wformat=]  
printf("the number is %f\n",5);

****

**5. gcc编译多个文件**

// hello.c

#include<stdio.h>

#include"hello.h"

void printHello()

{

printf("hello world!\n");

}

//main.c

#include<stdio.h>

#include"hello.h"

int main()

{

printHello();

return 0;

}

//hello.h

//仅包含函数声明

#ifndef \_HELLO\_

#define \_HELLO\_

void printHello();

#endif

编译这三个文件，可以一次编译：

gcc hello.c main.c -o main 生成可执行文件main

也可以独立编译：

gcc -Wall -c main.c -o main.o  
gcc -Wall -c hello.c -o hello.o  
gcc -Wall main.o hello.o -o main

独立编译的好处是，当其中某个模块发送改变时，只需要编译该模块就行，不必重新编译所有文件，这样可以节省编译时间。

**6. 使用外部库**  
在使用C语言和其他语言进行程序设计的时候，我们需要头文件来提供对常数的定义和对系统及库函数调用的声明。库文件是一些预先编译好的函数集合，那些函数都是按照可重用原则编写的。它们通常由一组互相关联的可重用原则编写的，它们通常由一组互相关联的用来完成某项常见工作的函数构成。使用库的优点在于：

* 模块化的开发
* 可重用性
* 可维护性

库又可以分为静态库与动态库：

* 静态库（.a）：程序在编译链接的时候把库的代码链接到可执行文件中。程序运行的时候将不再需要静态库。静态库比较占用磁盘空间，而且程序不可以共享静态库。运行时也是比较占内存的，因为每个程序都包含了一份静态库。
* 动态库（.so或.sa）：程序在运行的时候才去链接共享库的代码，多个程序共享使用库的代码，这样就减少了程序的体积。

一般头文件或库文件的位置在：

* /usr/include及其子目录底下的include文件夹
* /usr/local/include及其子目录底下的include文件夹
* /usr/lib
* /usr/local/lib
* /lib