简单编译

hello world 是如何输出的

```
// test.c
#include <stdio.h>
int main(){
    printf("hello world\n");
    return 0;
}
简单的hello world
```

```
***
#一步到位的编译指令
gcc test.c -o test
```

- 实际上 分为4各阶段
    1. 预编译(Preprocessing)
    2. 编译(Compilation)
    3. 汇编 (Assembly)
    4. 连接(Linking)。

预处理

```
gcc -E test.c -o test.i
#或
gcc -E test.c
```

- gcc的-E选项可以让编译器在预处理后停止
- 本例中是将头文件插入到源文件中

编译

预处理之后可直接对生成的test.i文件编译 生成汇编代码

```
gcc -S test.i -o test.s
```

- gcc的-S西选项表示在程序编译期间,生成汇编代码后停止.
  -o输出汇编代码文件

汇编

对于上一节生成的汇编代码文件test.s,gas编译器负责将其编译成目标文件

```
gcc -c test.s -o test.o
```

链接

gcc链接器是gas提供的,负责将程序的目标文件与所有的附加目标文件链接起来
附加的目标文件包括静态链接库和动态链接库.

对于上一小节中生成的test.o 将其与C标准库进行链接 生成可执行文件test

```
gcc test.o -o test
```

多文件编译

通常整个程序是有多个源文件组成的 相应的就生成了多个编译单元,使用gcc可以很好的管理这些编译单元.
假设一个由test1,c & test2.c两个源文件组成的程序,为了对们进行编译并最终生成可执行文件, 可以使用下面这条命令

gcc test1.c test2.c -o test

如果同时处理的文件不止一个，GCC仍然会按照预处理、编译和链接的过程依次进行。如果深究起来，上面这条命令大致相当于依次执行如下三条命令：

```
gcc -c test1.c -o test1.o
gcc -c test2.c -o test2.o
gcc test1.o test2.o -o test
```

检错

```
######################
gcc -pedantic illcode.c -o illcode
######################
gcc -Wall illcode.c -o illcode
######################
gcc -Werror test.c -o test.c
```

> 1. gcc的-pedantic选项能帮助程序员发现一些不符合标准的代码 /只有一部分 不能全部发现/
> 2. 除了-pedantic之外，GCC还有一些其它编译选项也能够产生有用的警告信息。这些选项大多以-W开头，其中最有价值的当数-Wall了，使用它能够使GCC产生尽可能多的警告信息。
> 3. gcc的-Werror选项时 gcc会在warning的地方停止编译

库文件链接(？)

开发软件时，完全不使用第三方函数库的情况是比较少见的，通常来讲都需要借助许多函数库的支持才能够完成相应的功能。从程序员的角度看，函数库实际上就是一些头文件（.h）和库文件（so、或lib、dll）的集合。。虽然Linux下的大多数函数都默认将头文件放到/usr/include/目录下，而库文件则放到/usr/lib/目录下； Windows所使用的库文件主要放在Visual Stido的目录下的include和lib，以及系统文件夹下。但也有的时候，我们要用的库不再这些目录下，所以GCC在编译时必须用自己的办法来查找所需要的头文件和库文件.

例如我们的程序test.c是在linux上使用c连接mysql，这个时候我们需要去mysql官网下载MySQL Connectors的C库，下载下来解压之后，有一个include文件夹，里面包含mysql connectors的头文件，还有一个lib文件夹，里面包含二进制so文件libmysqlclient.so

其中includle文件夹的路径是/usr/dev/mysql/include,lib文件夹是/usr/dev/mysql/lib

1. 编译成可执行文件
   首先我们要进行编译test.c为目标文件，这个时候需要执行

```
gcc -c -I /usr/dev/mysql/include test.c -o test.o
```

2. 链接

最后我们把所有目标文件链接成可执行文件:

```
gcc –L /usr/dev/mysql/lib –lmysqlclient test.o –o test
```

Linux下的库文件分为两大类分别是动态链接库（通常以.so结尾）和静态链接库（通常以.a结尾），二者的区别仅在于程序执行时所需的代码是在运行时动态加载的，还是在编译时静态加载的。

3. 强制链接时使用静态链接库
   默认情况下，GCC在链接时优先使用动态链接库，只有当动态链接库不存在时才考虑使用静态链接库，如果需要的话可以在编译时加上-static选项，强制使用静态链接库。

在/usr/dev/mysql/lib目录下有链接时所需要的库文件libmysqlclient.so和libmysqlclient.a，为了让GCC在链接时只用到静态链接库，可以使用下面的命令:

```
gcc –L /usr/dev/mysql/lib –static –lmysqlclient test.o –o test
```