Linux GCC常用命令

1 简介

GCC 的意思也只是 GNU C Compiler 而已。经过了这么多年的发展，GCC 已经不仅仅能支持 C 语言；它现在还支持 Ada 语言、C++ 语言、Java 语言、Objective C 语言、Pascal 语言、COBOL语言，以及支持函数式编程和逻辑编程的 Mercury 语言，等等。而 GCC 也不再单只是 GNU C 语言编译器的意思了，而是变成了 GNU Compiler Collection 也即是 GNU 编译器家族的意思了。另一方面，说到 GCC 对于操作系统平台及硬件平台支持，概括起来就是一句话：无所不在。

2 简单编译

示例程序如下：

```
//test.c
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
```

```
}
```

这个程序，一步到位的编译指令是:

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

实质上，上述编译过程是分为四个阶段进行的，即预处理(也称预编译，Preprocessing)、编译(Compilation)、汇编 (Assembly)和连接(Linking)。

2.1预处理

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

可以输出test.i文件中存放着test.c经预处理之后的代码。打开test.i文件，看一看，就明白了。后面那条指令，是直接在命令行窗口中输出预处理后的代码.

gcc的-E选项，可以让编译器在预处理后停止，并输出预处理结果。在本例中，预处理结果就是将stdio.h 文件中的内容插入到test.c中了。

2.2编译为汇编代码(Compilation)

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选项，表示在程序编译期间，在生成汇编代码后，停止，-o输出汇编代码文件。

2.3汇编(Assembly)

对于上一小节中生成的汇编代码文件test.s，gas汇编器负责将其编译为目标文件，如下：

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

2.4连接(Linking)

gcc连接器是gas提供的，负责将程序的目标文件与所需的所有附加的目标文件连接起来，最

终生成可执行文件。附加的目标文件包括静态连接库和动态连接库。
对于上一小节中生成的test.o，将其与 C 标准输入输出库进行连接，最终生成程序test

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

在命令行窗口中，执行./test, 让它说HelloWorld吧！

3 多个程序文件的编译

通常整个程序是由多个源文件组成的，相应地也就形成了多个编译单元，使用GCC能够很好地管理这些编译单元。假设有一个由test1.c和 test2.c两个源文件组成的程序，为了对它们进行编译，并最终生成可执行程序test，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
```

4 检错

```
gcc -pedantic illcode.c -o illcode
```

`-pedantic` 编译选项并不能保证被编译程序与ANSI/ISO C标准的完全兼容，它仅仅只能用来帮助Linux程序员离这个目标越来越近。或者换句话说，-pedantic选项能够帮助程序员发现一些不符合 ANSI/ISO C标准的代码，但不是全部，事实上只有ANSI/ISO C语言标准中要求进行编译器诊断的那些情况，才有可能被GCC发现并提出警告。

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。

```
gcc -Wall illcode.c -o illcode
```

GCC给出的警告信息虽然从严格意义上说不能算作错误，但却很可能成为错误的栖身之所。一个优秀的Linux程序员应该尽量避免产生警告信息，使自己的代码始终保持标准、健壮的特性。所以将警告信息当成编码错误来对待，是一种值得赞扬的行为！所以，在编译程序时带上-Werror选项，那么GCC会在所有产生警告的地方停止编译，迫使程序员对自己的代码进行修改，如下：

```
gcc -Werror test.c -o test
```

5 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`

5.1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
```

5.2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。

5.3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
```

静态库链接时搜索路径顺序:

```
1. ld会去找GCC命令中的参数-L
2. 再找gcc的环境变量LIBRARY_PATH
3. 再找内定目录 /lib /usr/lib /usr/local/lib 这是当初compile gcc时写在程序内的
```

动态链接时、执行时搜索路径顺序:

```
1. 编译目标代码时指定的动态库搜索路径
2. 环境变量LD_LIBRARY_PATH指定的动态库搜索路径
3. 配置文件/etc/ld.so.conf中指定的动态库搜索路径
4. 默认的动态库搜索路径/lib
5. 默认的动态库搜索路径/usr/lib
```

有关环境变量:
LIBRARY_PATH环境变量: 指定程序静态链接库文件搜索路径
LD_LIBRARY_PATH环境变量: 指定程序动态链接库文件搜索路径