静态库和共享库的制作和使用

一、基础：gcc的编译过程

1、编译工具链：

1）预处理器（cpp）：头文件展开、宏替换、去掉注释，命令：gcc **–E** hello.c –o hello.i

2）编译器（gcc）：c文件变成汇编文件，gcc **–S** hello.i –o hello.s

3）汇编器（as）：汇编文件变成二进制文件，命令：gcc **–c** hello.s –o hello.o

4）链接器（ld）：将函数库中相应的代码组合到目标文件中，命令：gcc hello.o –o hello

**在这四步编译过程中，如果直接执行后面的过程则会默认调用前面的过程。**也就是说，gcc hello.c –o hello会自动调用前三步。如果直接执行gcc hello.c则生成的可执行文件名称默认为a.out。

2、gcc参数说明：

-I（大写i）：后面跟头文件目录，在旧版本中-I和目录直接不能有空格，新版本中加不加空格都行。

-D：后面跟宏名，在编译的所有.c文件中加入指定的宏定义。

-O：优化程序。分为四个等级：-O0、-O1、-O2、-O3，把程序中冗余代码作出优化。

-o：指定生成文件的名字。

-Wall：输出警告信息。比如定义了一个变量但没有使用等警告信息。

-g：生成包含调试信息的可执行程序，在gdb调试时使用。

例：gcc hello.c –o hello –I./include –D DEBUG –O3 –Wall –g

二、静态库的制作和使用

1、命名规则：lib+库的名字+.a 例：libMytest.a

2、制作步骤：

1）生成对应的.o文件：gcc **–c** \*.c

2）将生成的.o文件打包成静态库：ar rcs + 静态库的名字（libMytest.a）+ 生成的所有的.o文件

3、发布和使用静态库：

1）发布静态库和头文件。

2）使用时包含头文件并链接静态库即可。-L：后面跟链接库的目录；-l（小写L）：后面跟链接库的名称（Mytest）。

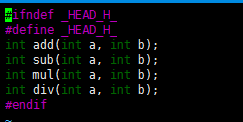
**\*在链接时并不会将整个静态库组合到可执行文件中，而是以静态库中的.o文件为单位，用到哪个组合那个。**

4、实例：

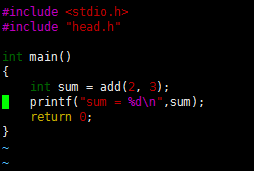
1）准备工作：

include目录存放头文件head.h；lib目录存放生成的库文件；src目录存放源文件add.c、sub.c、mul.c、div.c；main.c是测试程序。

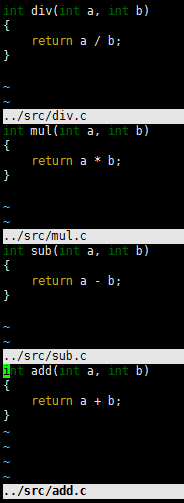
//head.h



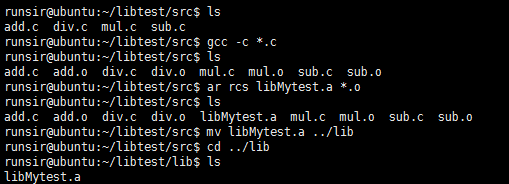
//main.c



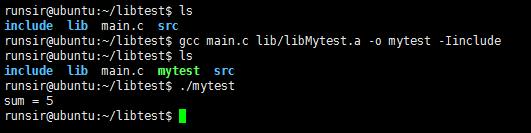
//src文件



2）制作静态库并将制作好的静态库放到lib目录下：



3）测试：



**gcc main.c lib/libMytest.a -o mytest –Iinclude**

**= gcc main.c -o mytest -Iinclude -L lib -l Mytest**

5、优缺点：

1）优点：

①静态库中相应的代码会组合到可执行程序中，所以发布程序的时候不需要提供对应的库。

②加载库的速度快。

2）缺点：

①应用程序的体积很大。

②库发生改变，应用程序需要重新编译。

三、共享库（动态库）的制作和使用

1、命名规则：lib+库的名字+.so 例：libMytest.so

2、制作步骤：

1）生成与位置无关的代码（.o文件）：gcc **–fPIC** –c \*.c（生产的代码中没有绝对路径，只有相对路径）

2）将生成的.o文件打包成共享库：gcc **–shared** -o libMytest.so \*.o

3、发布和使用共享库：

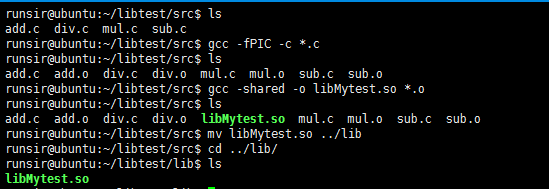
1）发布共享库和头文件。

2）使用时包含头文件并链接共享库。

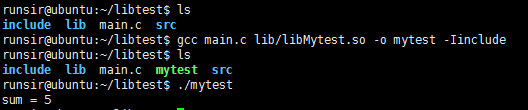
4、实例：

1）准备工作（同静态库）

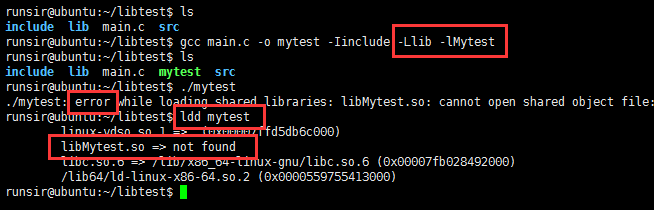
2）制作共享库并将制作好的共享库放到lib目录下：



3）测试：



**注：此时使用gcc main.c -o mytest -Iinclude -Llib –lMytest生成的文件将无法执行，可以用ldd命令查看可执行文件在运行中所依赖的共享库。**



**解决方案：可执行程序在运行时是通过动态链接器来加载动态库的**。

①可以将自己所编写的动态库放到/lib目录下，这样动态链接器就能够找到该动态库了。极其不推荐这样做。

②配置**LD\_LIBRARY\_PATH**环境变量：export LD\_LIBRARY\_PATH=自己所编写的动态库的路径，动态链接器会根据该环境变量找到动态库。也可以将该环境变量配置到~/.bashrc文件中达到永久配置效果。

③gcc编译时-L参数只在编译时有效，所以在运行时应用程序是不知道该目录的，但是可以在编译时加上**-Wl,--rpath=your\_lib\_dir1**参数来指明运行应用程序时动态库的位置。--rpath是ld编译选项，所以前面要加上-Wl（小写L）标明。

④将自己编写的动态库的路径添加到动态链接器的配置文件（/etc/ld.so.conf）中。

**注：动态库的路径应该写绝对路径，否则改变应用程序**

5、优缺点：

1）优点：

①应用程序的体积小。

②库发生改变，应用程序不需要重新编译。

2）缺点：

①发布程序的时候需要提供对应的库。

②加载库的速度慢。