GCC创建和使用静态链接库（.a文件）

Linux 下的静态链接库是以 .a 结尾的二进制文件，它作为程序的一个模块，在链接期间被组合到程序中。和静态链接库相对的是动态链接库（ .so 文件），它在程序运行阶段被加载进内存。

制作链接库的目的是希望别人使用我们已经实现的功能，但又不希望别人看到我们的源代码，这对商业机构是非常友好的。

Linux 下静态链接库文件的命名规则为：

```
libxxx.a
```

xxx 表示库的名字。例如，libc.a、libm.a、libieee.a、libgcc.a 都是 Linux 系统自带的静态库。

GCC 生成静态链接库

1) 首先使用 gcc 命令把源文件编译为目标文件，也即 .o 文件：

```
gcc -c 源文件列表
```

-c 选项表示只编译，不链接，我们已在《GCC -c选项》中进行了讲解。

2) 然后使用 ar 命令将 .o 文件打包成静态链接库，具体格式为：

```
ar rcs + 静态库文件的名字 + 目标文件列表
```

ar 是 Linux 的一个备份压缩命令，它可以将多个文件打包成一个备份文件（也叫归档文件），也可以从备份文件中提取成员文件。ar 命令最常见的用法是将目标文件打包为静态链接库。

对参数的说明：

- 参数 r 用来替换库中已有的目标文件，或者加入新的目标文件。
- 参数 c 表示创建一个库。不管库否存在，都将创建。
- 参数 s 用来创建目标文件索引，这在创建较大的库时能提高速度。

例如，下面的写法表示将目标文件 a.o、b.o 和 c.o 打包成一个静态库文件 libdemo.a：

```
ar rcs libdemo.a a.o b.o c.o
```

实例演示

在用户主目录（home 目录）下创建一个文件夹 test，将 test 作为整个项目的基础目录。在 test 目录中再创建四个源文件，分别是 add.c、sub.c、div.c 和 test.h。

add.c 实现两个数相加，代码展示如下：

```
#include “test.h”
int add(int a,int b)
{
    return a + b;
}
```

sub.c 实现两个数相减，代码展示如下：

```
#include “test.h”
int sub(int a,int b)
{
    return a - b;
}
```

div.c 实现两个函数相除，代码展示如下：

```
#include “test.h”
int div(int a,int b)
{
    return a / b;
}
```

还有一个 test.h 头文件，用来声明三个函数，代码展示如下：

```
#ifndef __TEST_H_
#define __TEST_H_

int add(int a,int b);
int sub(int a,int b);
int div(int a,int b);

#endif
```

接下来，我们就将以上代码制作成静态链接库。

首先将所有源文件都编译成目标文件：

```
gcc -c *.c
```

`*.c` 表示所有以 `.c` 结尾的文件，也即所有的源文件。执行完该命令，会发现 test 目录中多了三个目标文件，分别是 add.o、sub.o 和 div.o。

然后把所有目标文件打包成静态库文件：

```
ar rcs libtest.a *.o
```

`*.o` 表示所有以 `.o` 结尾的文件，也即所有的目标文件。执行完该命令，发现 test 目录中多了一个静态库文件 libtest.a，大功告成。

下面是完整的生成命令：

```
[c.biancheng.net ~]$ cd test
[c.biancheng.net test]$ gcc -c *.c
[c.biancheng.net test]$ ar rcs libtest.a *.o
```

GCC 使用静态链接库

使用静态链接库时，除了需要库文件本身，还需要对应的头文件：库文件包含了真正的函数代码，也即函数定义部分；头文件包含了函数的调用方法，也即函数声明部分。

为了使用上面生成的静态链接库 libtest.a，我们需要启用一个新的项目。在用户主目录（home 目录）中再创建一个文件夹 math，将 math 作为新项目的基础目录。

在比较规范的项目目录中，lib 文件夹一般用来存放库文件，include 文件夹一般用来存放头文件，src 文件夹一般用来存放源文件，bin 文件夹一般用来存放可执行文件。为了规范，我们将前面生成的 libtest.a 放到 math 目录下的 lib 文件夹，将 test.h 放到 math 目录下的 include 文件夹。

在 math 目录下再创建一个 src 文件夹，在 src 中再创建一个 main.c 源文件。

此时 math 目录中文件结构如下所示：

```
|-- include
|   `-- test.h
|-- lib
|   `-- libtest.a
`-- src
    `-- main.c
```

在 main.c 中，可以像下面这样使用 libtest.a 中的函数：

```
01.  #include <stdio.h>
```

```
#include "test.h"  //必须引入头文件

int main(void)
{
    int m, n;
    printf("Input two numbers: ");
    scanf("%d %d", &m, &n);
    printf("%d+%d=%d\n", m, n, add(m, n));
    printf("%d-%d=%d\n", m, n, sub(m, n));
    printf("%d÷%d=%d\n", m, n, div(m, n));

    return 0;
}
```

在编译 main.c 的时候，我们需要使用 `-I`（大写的字母 `i`）选项指明头文件的包含路径，使用 `-L` 选项指明静态库的包含路径，使用 `-l`（小写字母 `L`）选项指明静态库的名字。所以，main.c 的完整编译命令为：

```
gcc src/main.c -I include/ -L lib/ -l test -o math.out
```

注意，使用 `-l` 选项指明静态库的名字时，既不需要 `lib` 前缀，也不需要 `.a` 后缀，只能写 test，GCC 会自动加上前缀和后缀。

打开 math 目录，发现多了一个 math.out 可执行文件，使用 `./math.out` 命令就可以运行 math.out 进行数学计算。

完整的使用命令如下所示：

```
[c.biancheng.net ~]$ cd math
[c.biancheng.net math]$ gcc src/main.c -I include/ -L lib/ -l test -o math.out
[c.biancheng.net math]$ ./math.out
Input two numbers: 27 9↙
27+9=36
27-9=18
27÷9=3
```