

3.2 编译器gcc的使用

动态库的创建步骤

(1) 在头文件 (.h) 中声明动态库所导出的函数

(2) 在源文件 (.c) 中实现动态库所导出的函数

(3) 编译源文件, 生成与位置无关的目标文件 (.o)

(4) 创建动态库

```
app.c
src E: c
include E: h
lib *.so
```

```
gcc -fPIC *.c -I ../include -c
```

```
gcc -shared -o libtest0x00.so *.o
```

动态库的使用

(1) 方式1

gcc [源文件] -L [动态库路径] -l [动态库名] -I [头文件路径] -o [可执行文件]

```
gcc app.c -Llib -ltest0x00 -I include -o app
./app
```

```
gcc -fPIC *.c -I ../include
gcc -shared -o libtest0x00.so *.o
```

```
gcc -o app app.c -I include -L lib
-ltest0x00
```

(2) 方式2

gcc [源文件] -I [头文件路径] [库文件名 (libxxx.so)] -o [可执行文件]

```
gcc app.c -I include ./lib/libtest0x00.so -o app
```

./app

⇒ 改为绝对路径:

```
gcc app.c -o app -I include $(pwd)/lib/libtest0x00.so
```

解决找不到链接库的方法

(1) 更新LD_LIBRARY_PATH

```
export LD_LIBRARY_PATH =
```

```
export LD_LIBRARY_PATH=[自定义动态库路径]
```

只起到临时作用

LD_LIBRARY_PATH: 指定查找动态库的路径 (除了默认路径), 该路径在默认路径前进行查找。

(2) 配置环境变量

(3) 直接将动态库拷贝至/usr/lib等系统目录下 (可行但强烈不推荐)

(4) 更新/etc/ld.so.conf文件

```
etc/ldso.conf
ldconf.py
```


将动态库的绝对路径写入/etc/ld.so.conf文件，后使用ldconfig命令更新

将编译得到的二进制文件拷贝到别的主机，是否能够运行？

静态可以 动态不可以