

Linux系统及Shell编程

东北林业大学

卢洋

3.2

编译器gcc的使用

Linux库的创建与使用

1. 什么是库
2. 静态库的创建和使用
3. 动态库的创建和使用

静态库的创建步骤

1. 在头文件（.h）中声明静态库所导出的函数
2. 在源文件（.c）中实现静态库所导出的函数
3. 编译源文件，生成目标文件（.o）
4. 通过命令ar将目标文件加入到静态库中
5. 将静态库拷贝到系统默认的存放库的路径，或指定的路径下

1. 在头文件 (`.h`) 中声明静态库所导出的函数

2. 在源文件 (`.c`) 中实现静态库所导出的函数

还是用之前的写好的程序就可以

① `app.c`

② `other1.h`

③ `other1.c`

④ `other2.h`

⑤ `other2.c`

3. 编译源文件，生成目标文件 (.o)

- `gcc -o other1.o -c other1.c`
- `gcc -o other2.o -c other2.c`

4. 通过命令`ar`将目标文件加入到静态库中

- `ar rcs libother1.a other1.o`
- `ar rcs libother2.a other2.o`

5. 将静态库拷贝到系统默认的存放库的路径，或指定的路径下

静态库的使用

1. 编译文件

- `-L`: 指示编译器，装载的函数库。该函数库位于系统默认的路径（包含 `/usr/lib`、`/usr/lib64`），或通过 `-L` 选项指定库所在的路径
- `gcc -o app app.c -L./ -lother1 -lother2`

2. 运行

`./app`

项目的结构

- `include`: 存放头文件
- `src`: 存放源代码
- `Lib`: 存放库文件

```
.  
├── include  
├── lib  
├── main.c  
└── src
```


静态库的使用

1. 编译文件

- `-L`: 指示编译器，装载的函数库。该函数库位于系统默认的路径（包含 `/usr/lib`、`/usr/lib64` 以及 `./`），或通过 `-L` 选项指定库所在的路径
- `gcc -o app app.c -Iinclude -Llib -lother1 -lother2`

2. 运行

`./app`

使用系统默认库

1. 路径

- 库: `/usr/local/lib`
- 头文件: `/usr/local/include`

2. 编译

- `gcc -o app app.c -lother1 -lother2`

3. 运行

`./app`

将编译得到的二进制文件拷贝到别的主机，
是否能够运行？