**一、什么是库（Library）？**

​**​库（Library）​**​ 是一组预先编译好的函数、类、变量或资源的集合，用于实现特定功能（如数学计算、网络通信、文件操作等）。它的核心作用是​**​代码复用​**​，避免开发者重复造轮子，同时提高开发效率和程序的可维护性。

在 Linux 系统中，库主要分为两种类型：

​​静态库（Static Library）​​：静态库在程序编译时会被连接到目标代码中，程序运行时将不再需要该静态库，因此体积较大

​​动态库（Shared Library/Dynamic Library）​​：动态库在程序编译时并不会被连接到目标代码中，而是在程序运行时才被载入，因此在程序运行时还需要动态库存在，因此代码体积较小。

|  |  |
| --- | --- |
| 动态库 | 静态库 |
| 编译得到程序体积更小，动态升级 | 可以独立运行（运行时不依赖库） |

**二、Linux 下制作库**

**2.1 静态库（.a 文件）的制作**

静态库的本质是目标文件（.o）的归档包，通过 ar 工具打包生成。

​**​示例**

1.将c文件编译目标文件 (add.c -> add.o)

gcc -c add.c

2.由.o文件创建静态库 (add.o -> libadd.a)

ar crs libadd.a add.o

-r 替换

-c 不管库是否存在都建创建

-s 生成索引（加速链接时查找符号）

**2.2 动态库（.so 文件）的制作**

动态库需要编译时生成位置无关代码（PIC，Position-Independent Code），并通过 gcc 的 -shared 选项链接。

**​示例**

1.将c文件编译目标文件 (add.c -> add.o)

gcc -c add.c

2.由.o文件创建动态库文件(add.o -> libadd.so)

gcc -shared -fPIC -o libadd.so add.o

**三、Linux 下使用库**

**1. 使用静态库**

步骤1：编写测试程序调用静态库中的函数（例如：test.c）

步骤2：编译程序

gcc test.c -o test -L. -ladd --static

// -L 指定库路径

// -l 指定库名

// --static 静态编译

注意：使用 -l 选项指定库名, -L 选项指定库路径

库名：为之前编译的libadd.a或libadd.so, 去掉前缀和后缀的名字

库路径：默认路径/usr/lib 或 /lib，在则不需要指定库路径，不在则需要指定

步骤3：运行程序

./test

静态库已嵌入可执行文件，直接运行即可

**2. 使用动态库**

步骤1：编写测试程序调用动态库中的函数(例如：test.c)

步骤2：编译程序（与静态相同）

gcc test.c -o test -L. -ladd

步骤3：运行程序

./test //运行时 需要库，找不到报错

注意：由于采用的是动态库，运行时需要加载库，因此需要配置

一般有两种方案：**临时、永久**

**方案1：临时指定路径（当前终端有效）**

export LD\_LIBRARY\_PATH=.:$LD\_LIBRARY\_PATH #将当前目录加入路径

**方案2：永久**

简单做法： 直接拷贝库文件到 /usr/lib 或 /lib中

更好的做法： 修改系统配置文件

echo "/xx/libpath" | sudo tee /etc/ld.so.conf.d/my.conf

sudo ldconfig # 刷新动态库缓存

**四、常用工具**

​**​nm​**​：查看库中的符号（函数/变量）。示例：nm -D libadd.so（-D 表示查看动态符号）。

​**​ldd​**​：查看可执行文件依赖的动态库。示例：ldd test。

​**​ar​**​：静态库管理工具（查看、删除、添加目标文件）。示例：ar t libadd.a（查看静态库包含的目标文件）。