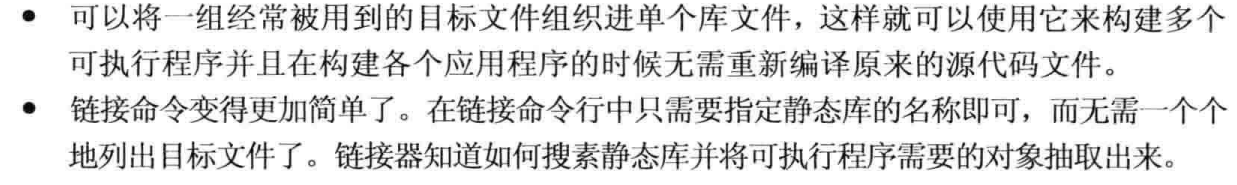
第四十一章

1. 共享库基础：
2. 目标库：

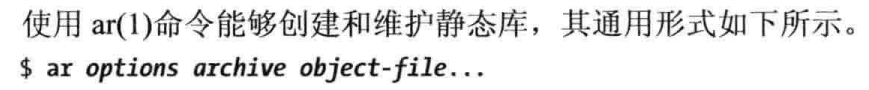
分为静态库和动态库。

1. 静态库：

也成为归档文件。它的优点是：

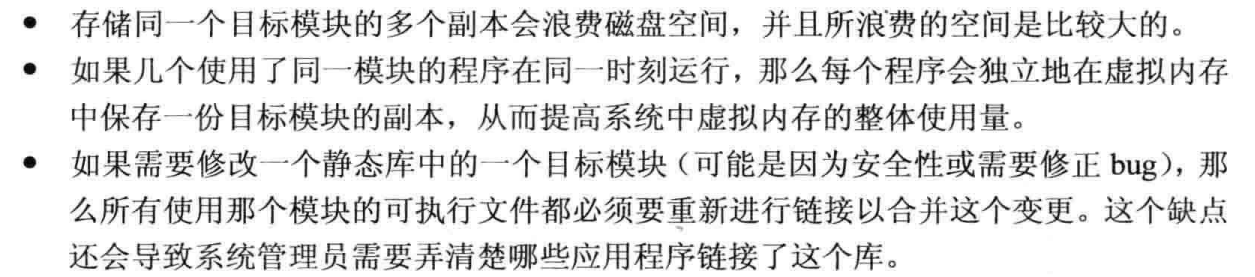


创建和维护静态库：

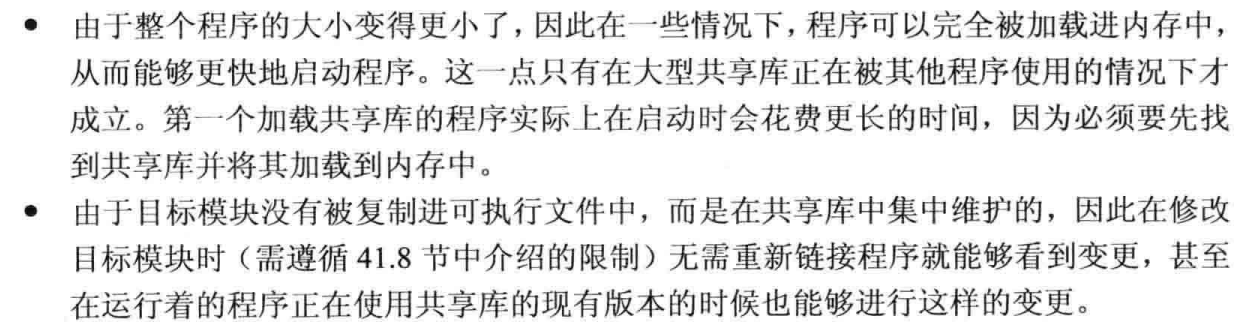


1. 共享库概述：

静态库的缺点：

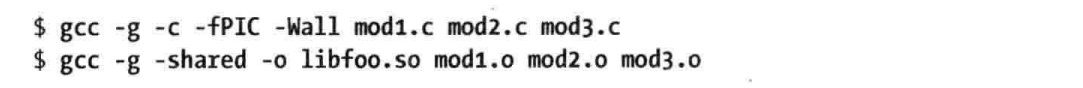


共享库优点：



1. 创建和使用共享库：

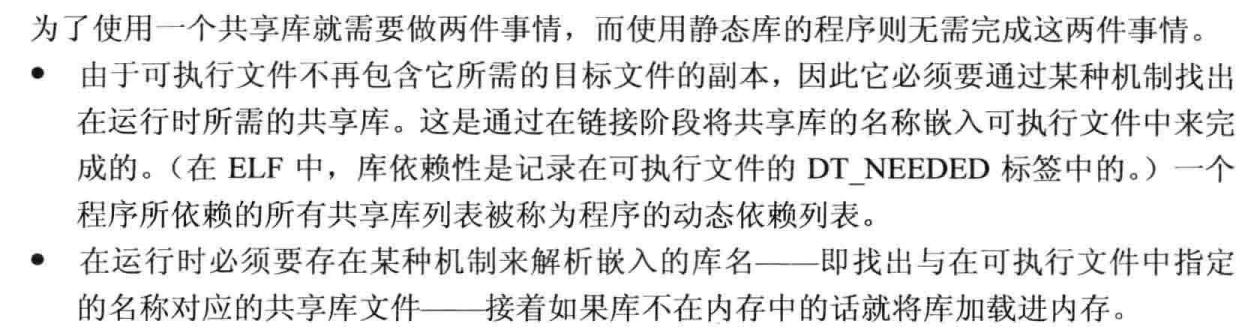
创建一个共享库：



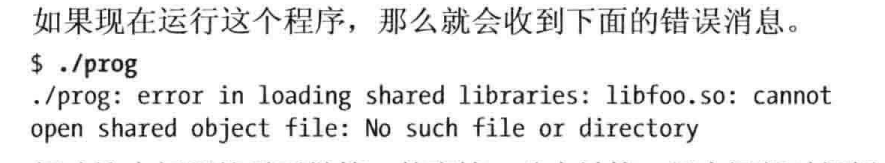
位置独立代码：

-fPIC 编译参数是生成位置独立代码。

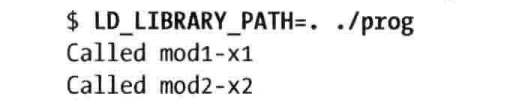
使用一个共享库：

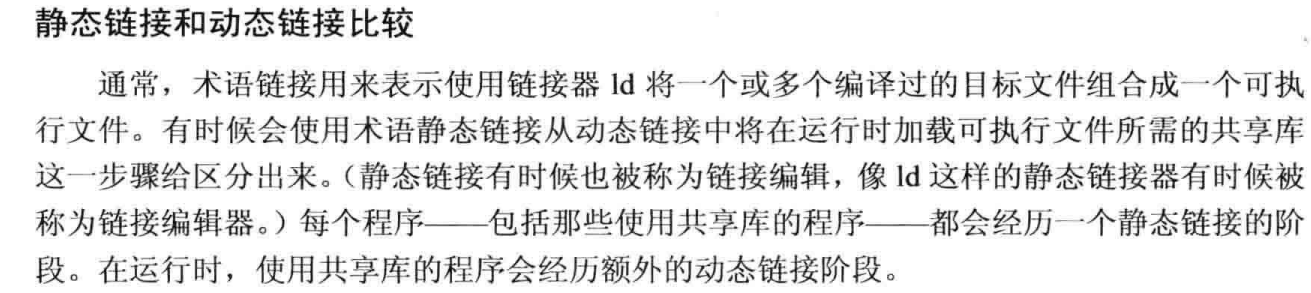




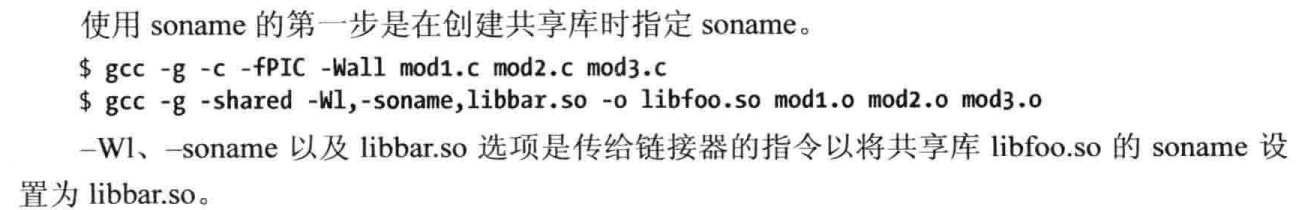


LD\_LIBRARY\_PATH环境变量：

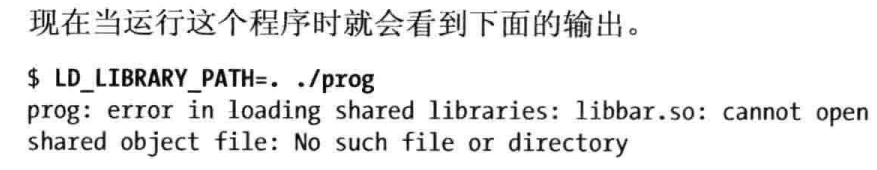


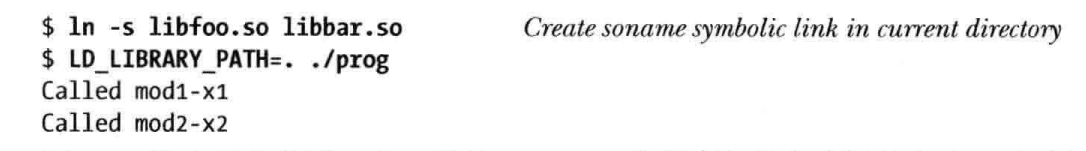


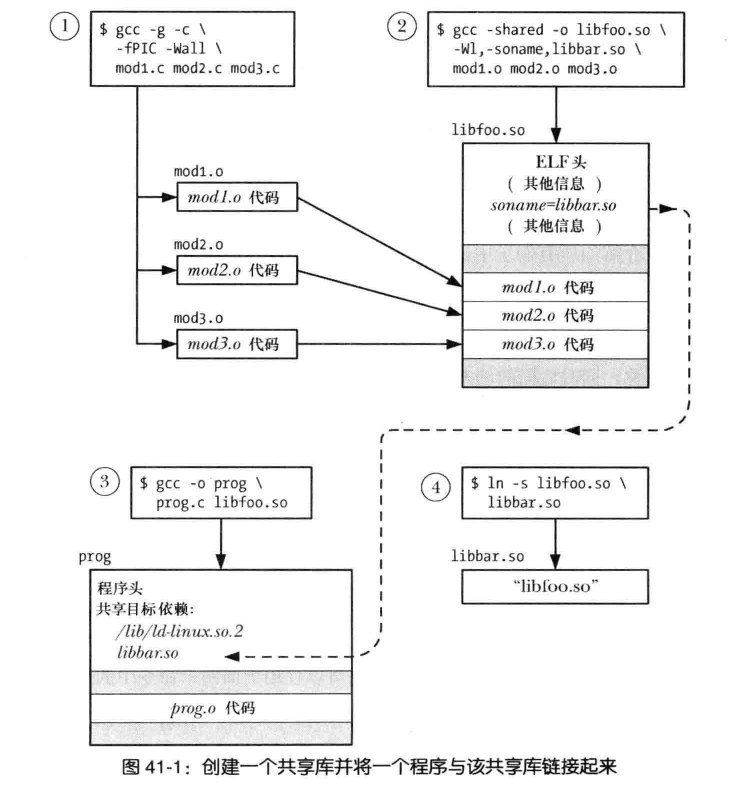
共享库soname：

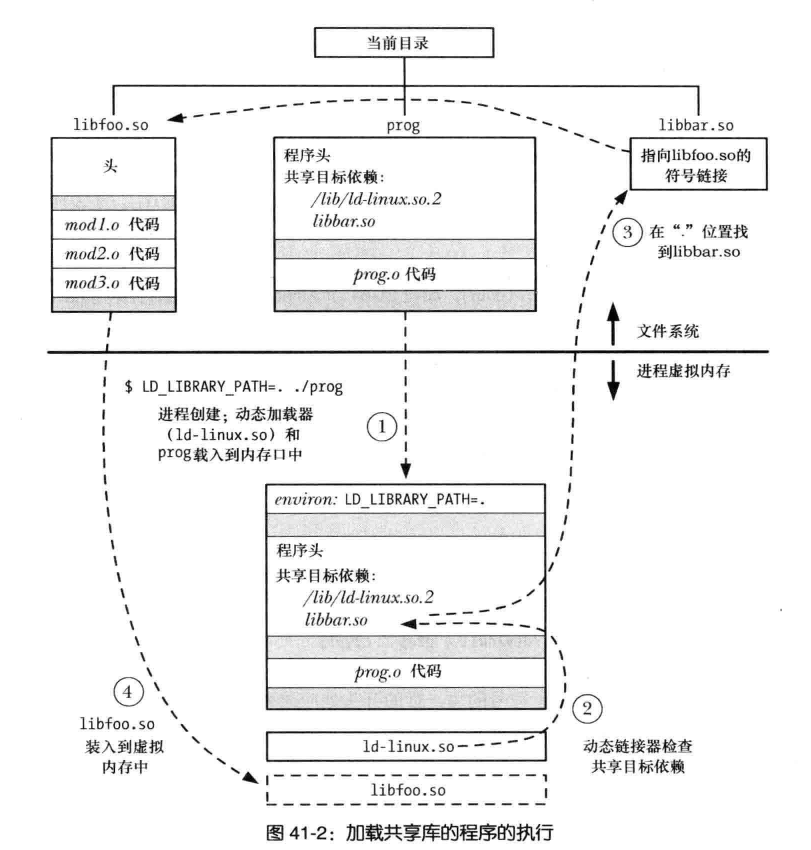






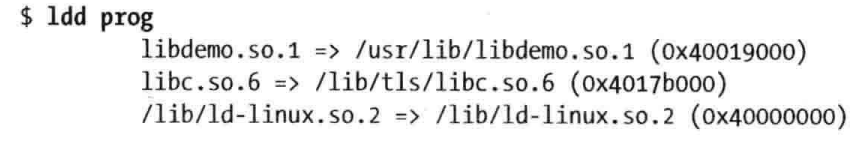


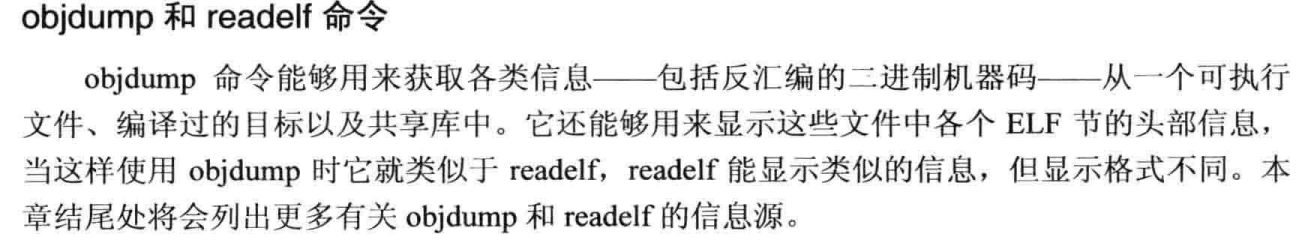




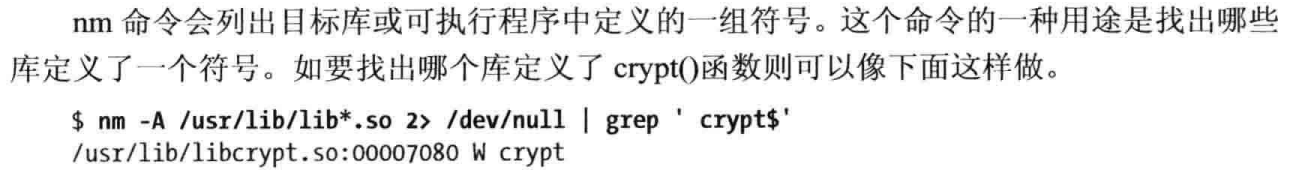
1. 使用共享库有用的工具：

Ldd命令：显示一个程序运行时所需要的共享库：





Nm命令：

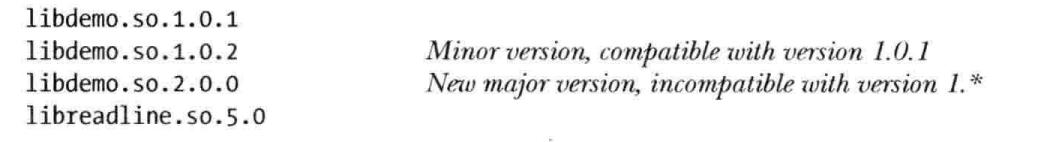


1. 共享库版本和命名规则：

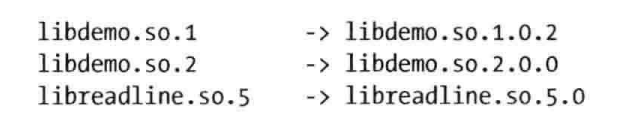
主要版本标识符：一个数字。

次要版本标识符：两个点分割的数字。

真实命名：

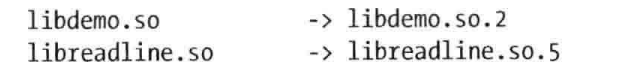


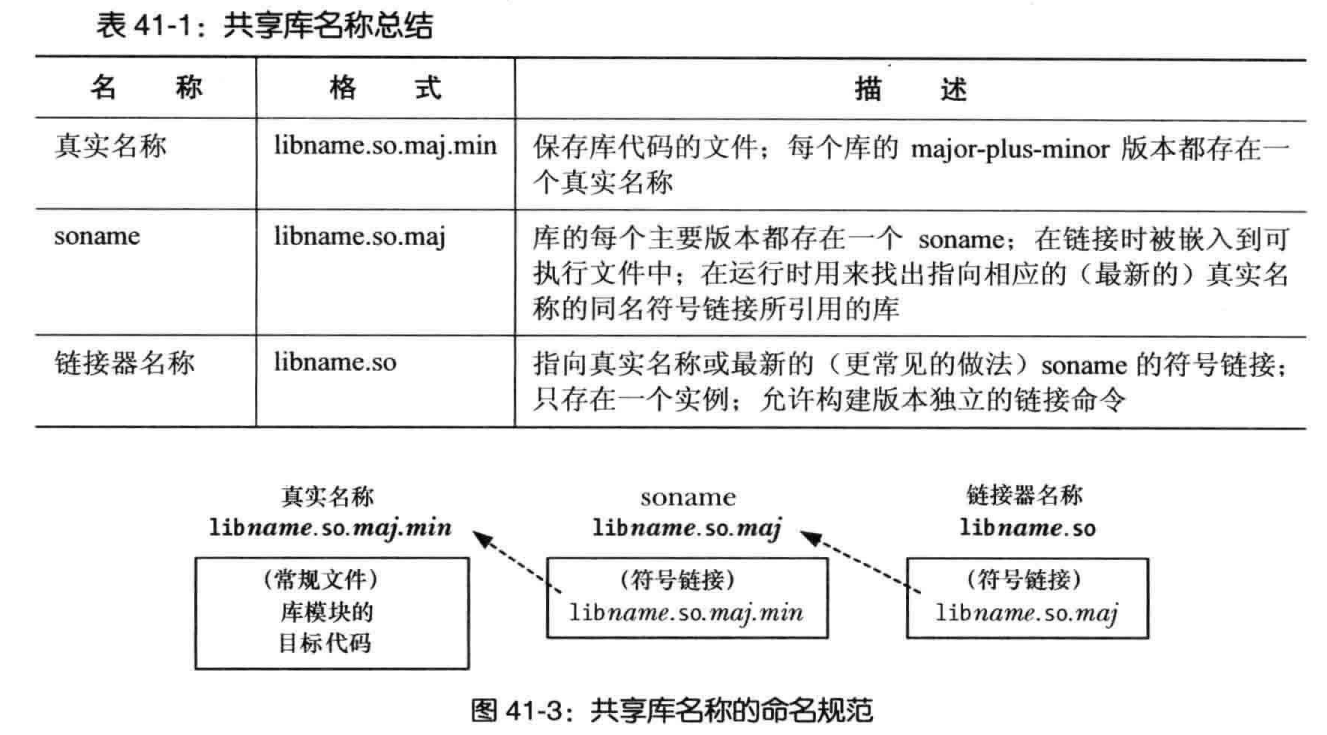
Soname：

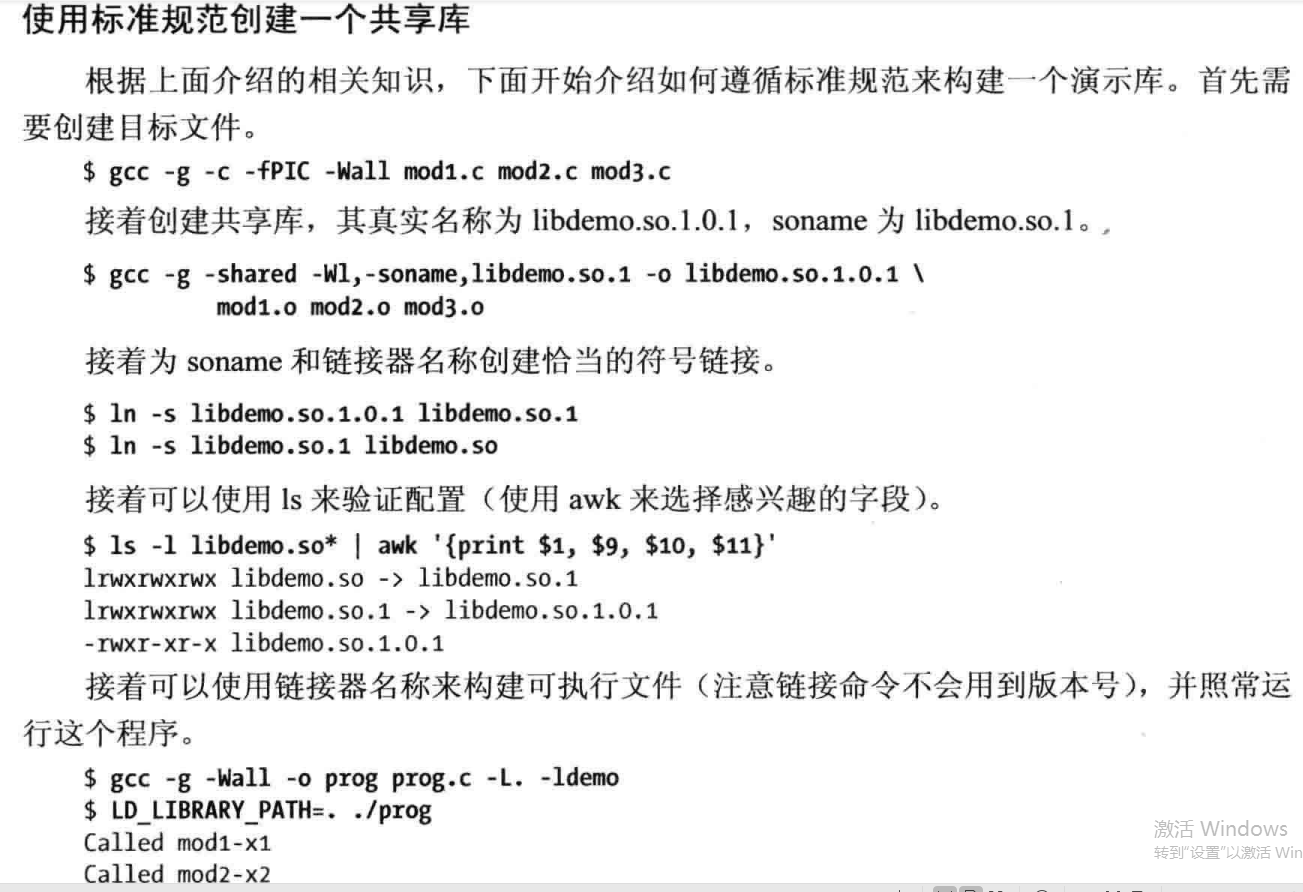


链接器名称：

只包含库名。

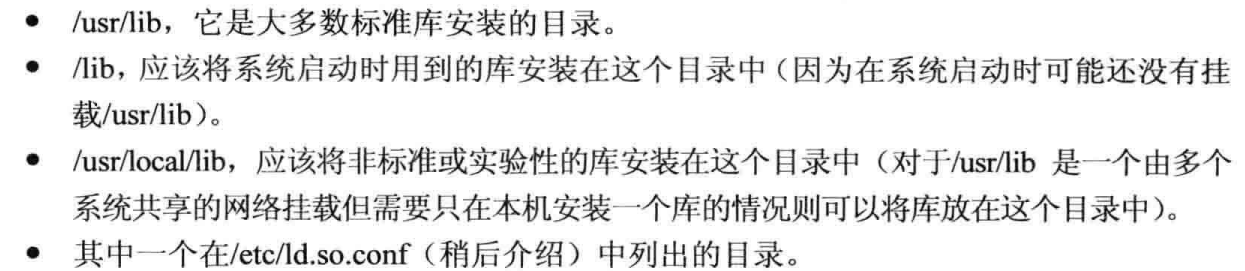




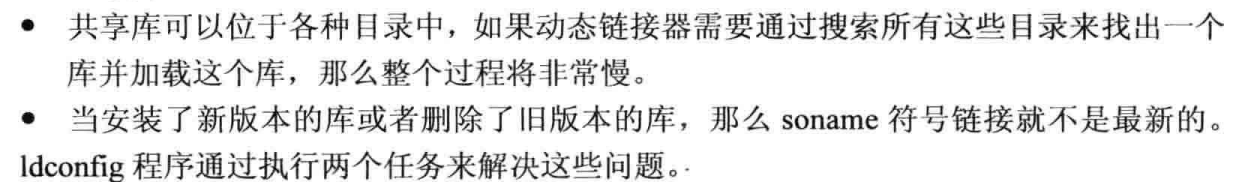


1. 安装共享库：

标准库目录包括：



Ldconfig：解决了共享库的两个潜在问题：

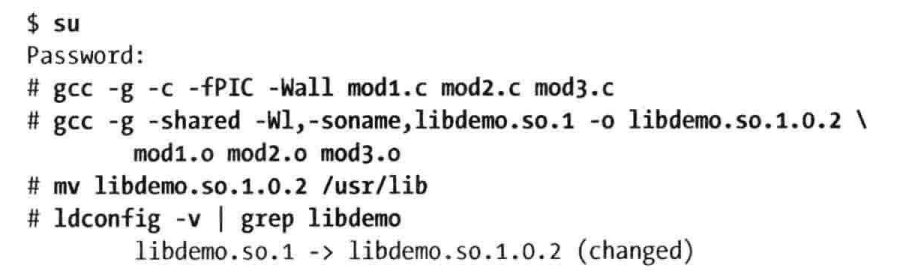


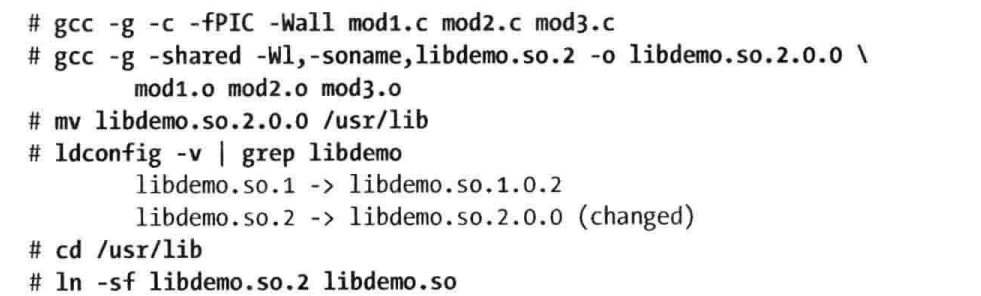
具体做法查看书本。

1. 兼容与不兼容库比较：

兼容的情况：

1. 公共方法和变量的语义保持不变。
2. 添加API不会影响。
3. 分配的结构和返回的结构保持不变。
4. 升级共享库：





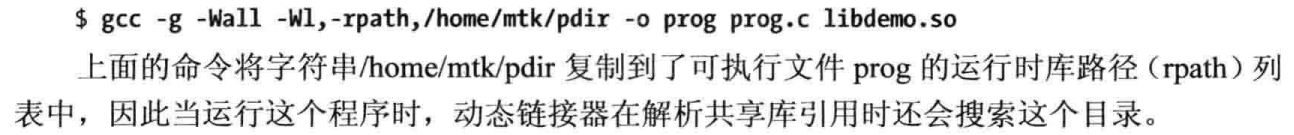
1. 在目标文件中指定库搜索目录：

目前为止介绍了两种通知动态链接器共享库位置的方法：

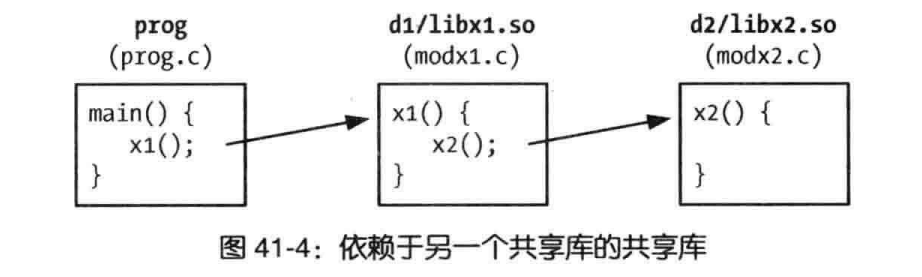
1. 使用LD\_LIBRARY\_PATH变量。
2. 安装到标准库目录中。

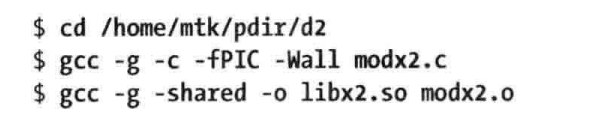
第三种方法：

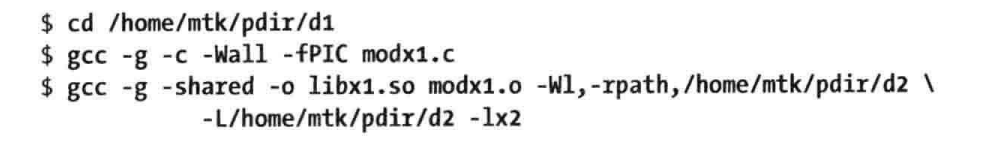
使用rpath:

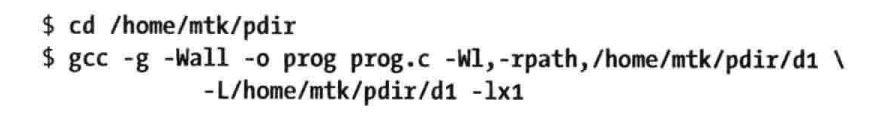


在构建共享库的时候也可以使用：









注意，上面的-rpath和-L的区别是什么？

我试了一下，如果在命令行中没有-L，直接报错，而没有-rpath不会报错，但是在运行的时候会报错。所以，-rpath是把路径复制到可执行文件的运行时的库路径，而-L是根据此路径搜索库。

ELF DT\_RPATH和DT\_RUNPATH

这两种的区别是相对于LD\_LIBRARY\_PATH的优先级。

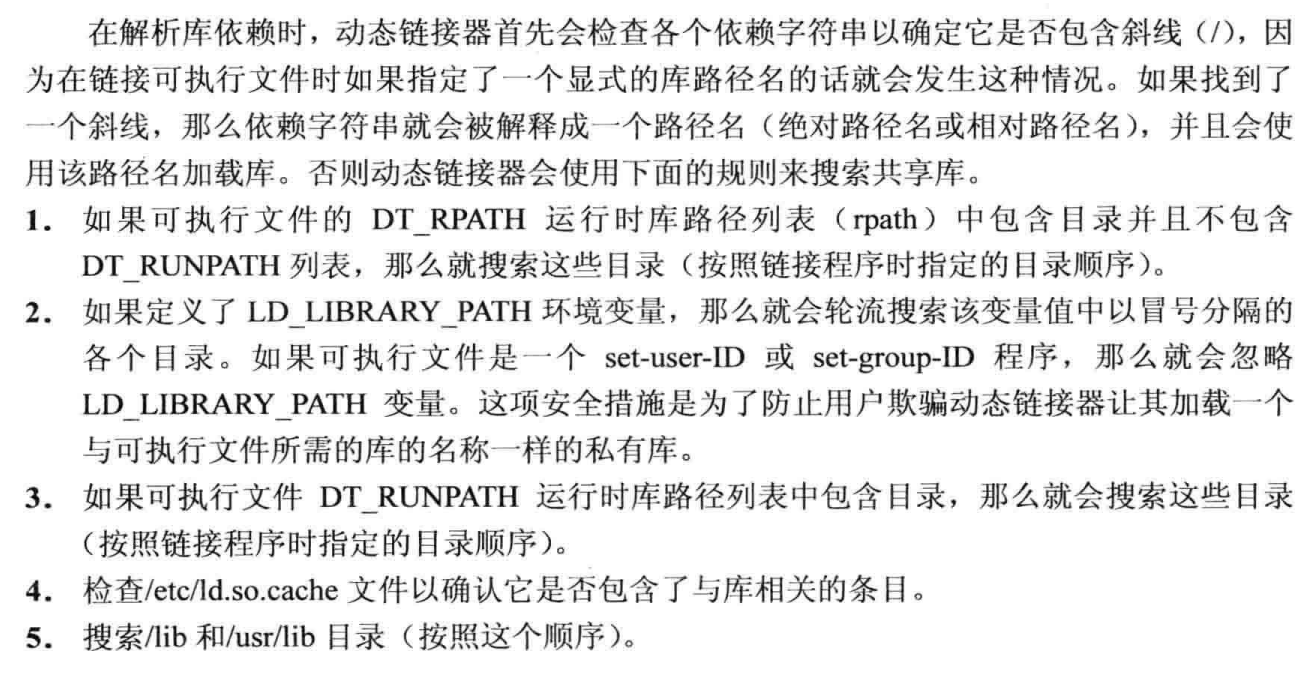
前者优先级更高，后者低。



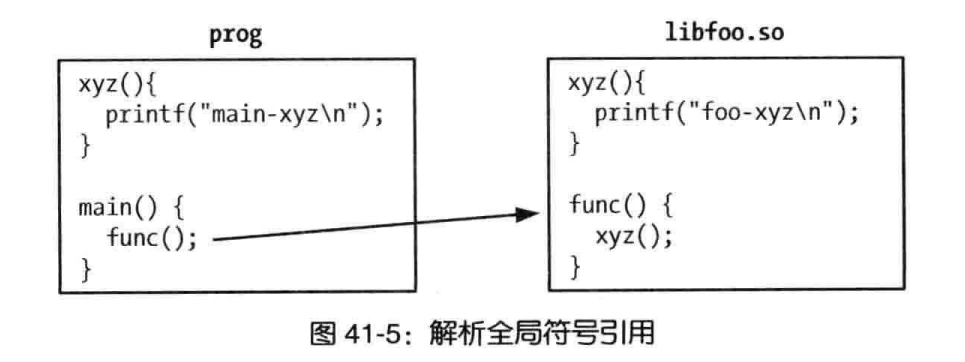
在rpath里使用$ORIGIN

能使共享库放在应用程序隔壁。

1. 在运行时找出共享库：

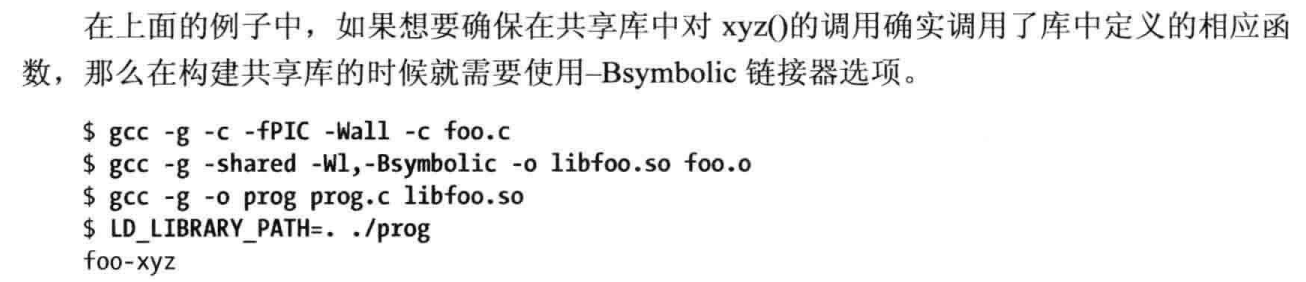


1. 运行时的符号解析：





注意，我在静态库里是无法编译成功的。



1. 使用静态库代替共享库：

在某些时候使用静态库比共享库要好。

强制使用库的静态版本：

