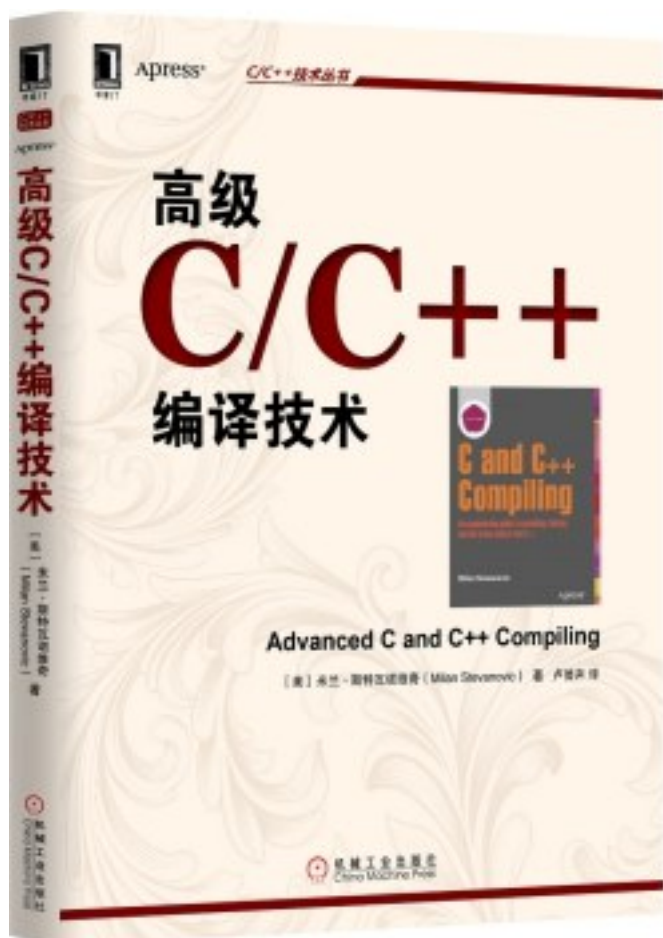


高级C/C++编译技术



[高级C/C++编译技术_下载链接1](#)

著者:Milan Stevanovic

出版者:机械工业出版社

出版时间:2015-4

装帧:

isbn:9787111496182

米兰·斯特瓦诺维奇编著的这本《高级C/C++编译技术》从多个角度全面、系统地讲解多任务操作系统中编译、链接、装载与库的内幕和技术细节，为深入理解和掌握系统底层技术提供详实参考和实践指南。与纯粹讲解

理论与技术细节的书不同。本书一方面对基本的理论进行了阐述，另一方面则聚焦于C/C++使用静态库和动态库的一些注意事项，并举例说明如何解决实际的链接与装载问题。此外，本书尽量使用通俗易懂的语言来阐述这些知识，并补充了大量示例，避免让读者整天纠结于枯燥的理论。

本书共14章，其中第1章至第4章对多任务操作系统、程序生命周期以及代码重用等重要概念进行介绍，为后续内容做铺垫；第5章介绍静态库的使用方法及其设计技巧；第6章至第11章介绍动态库的相关概念，包括不同平台中动态库的技术细节，比如库文件定位、引用解析与符号处理等，以及一些动态库设计的基本方法与原则和不同平台下应当注意的设计细节；第12章至第14章主要给出一些实践方面的总结，便于读者快速查找相关的概念，而且还总结了一些特定平台的二进制文件分析工具。

作者介绍:

目录: 译者序

前言

第1章 多任务操作系统基础 1

1.1 一些有用的抽象概念 1

1.2 存储器层次结构与缓存策略 2

1.3 虚拟内存 3

1.4 虚拟地址 5

1.5 进程的内存划分方案 5

1.6 二进制文件、编译器、链接器与装载器的作用 6

1.7 小结 7

第2章 程序生命周期阶段基础 8

2.1 基本假设 8

2.2 编写代码 9

2.3 编译阶段 11

2.3.1 基本概念 11

2.3.2 相关概念 11

2.3.3 编译的各个阶段 12

2.3.4 目标文件属性 23

2.3.5 编译过程的局限性 24

2.4 链接 26

2.4.1 链接阶段 26

2.4.2 链接器视角 31

2.5 可执行文件属性 33

2.5.1 各种节的类型 34

2.5.2 各种符号类型 36

第3章 加载程序执行阶段 37

3.1 shell的重要性 37

3.2 内核的作用 39

3.3 装载器的作用 39

3.3.1 装载器视角下的二进制文件（节与段） 39

3.3.2 程序加载阶段 40

3.4 程序执行入口点 43

3.4.1 装载器查找入口点 43

3.4.2 `_start()`函数的作用 43

3.4.3 `__libc_start_main()`函数的作用 44

3.4.4 栈和调用惯例 44

第4章 重用概念的作用 46

- 4.1 静态库 46
- 4.2 动态库 48
 - 4.2.1 动态库和共享库 49
 - 4.2.2 动态链接详解 51
 - 4.2.3 Windows平台中动态链接的特点 54
 - 4.2.4 动态库的特点 56
 - 4.2.5 应用程序二进制接口 (ABI) 56
- 4.3 静态库和动态库对比 57
 - 4.3.1 导入选择条件的差异 57
 - 4.3.2 部署难题 59
- 4.4 一些有用的类比 61
- 4.5 结论：二进制复用概念所产生的影响 63
- 第5章 使用静态库 64
 - 5.1 创建静态库 64
 - 5.1.1 创建Linux静态库 64
 - 5.1.2 创建Windows静态库 65
 - 5.2 使用静态库 65
 - 5.3 静态库设计技巧 66
 - 5.3.1 丢失符号可见性和唯一性的可能性 66
 - 5.3.2 静态库使用禁忌 67
 - 5.3.3 静态库链接的具体规则 68
 - 5.3.4 将静态库转换成动态库 68
 - 5.3.5 静态库在64位Linux平台上的问题 68
- 第6章 设计动态链接库：基础篇 70
 - 6.1 创建动态链接库 70
 - 6.1.1 在Linux中创建动态库 70
 - 6.1.2 在Windows中创建动态链接库 72
 - 6.2 设计动态库 75
 - 6.2.1 设计二进制接口 75
 - 6.2.2 设计应用程序的二进制接口 79
 - 6.2.3 控制动态库符号的可见性 82
 - 6.2.4 完成链接需要满足的条件 94
 - 6.3 动态链接模式 94
 - 6.3.1 加载时动态链接 95
 - 6.3.2 运行时动态链接 95
 - 6.3.3 比较两种动态链接模式 98
- 第7章 定位库文件 99
 - 7.1 典型用例场景 99
 - 7.1.1 开发用例场景 99
 - 7.1.2 用户运行时用例场景 100
 - 7.2 构建过程中库文件的定位规则 101
 - 7.2.1 Linux平台构建过程中的库文件定位规则 101
 - 7.2.2 Windows构建过程中的库文件定位规则 105
 - 7.3 运行时动态库文件的定位规则 109
 - 7.3.1 Linux运行时动态库文件的定位规则 110
 - 7.3.2 Windows运行时动态库文件的定位规则 114
 - 7.4 示例：Linux构建时与运行时的库文件定位 115
- 第8章 动态库的设计：进阶篇 119
 - 8.1 解析内存地址的必要性 119
 - 8.2 引用解析中的常见问题 120
 - 8.3 地址转换引发的问题 122
 - 8.3.1 情景1：客户二进制程序需要知道动态库符号地址 122
 - 8.3.2 情景2：被装载的库不需要知道其自身符号地址 123
 - 8.4 链接器-装载器协作 124

8.4.1 总体策略	125
8.4.2 具体技术	126
8.4.3 链接器重定位提示概述	127
8.5 链接器-装载器协作实现技术	128
8.5.1 装载时重定位 (LTR)	129
8.5.2 位置无关代码 (PIC)	129
第9章 动态链接时的重复符号处理	134
9.1 重复的符号定义	134
9.2 重复符号的默认处理	137
9.3 在动态库链接过程中处理重复符号	140
9.3.1 处理重复符号问题的一般策略	142
9.3.2 链接器解析动态库重复符号的模糊算法准则	143
9.4 特定重复名称案例分析	144
9.4.1 案例1: 客户二进制文件符号与动态库ABI函数冲突	144
9.4.2 案例2: 不同动态库的ABI符号冲突	147
9.4.3 案例3: 动态库ABI符号和另一个动态库局部符号冲突	151
9.4.4 案例4: 两个未导出的动态库符号冲突	153
9.5 小提示: 链接并不提供任何类型的命名空间继承	161
第10章 动态库的版本控制	162
10.1 主次版本号与向后兼容性	162
10.1.1 主版本号变更	162
10.1.2 次版本号变更	163
10.1.3 修订版本号	163
10.2 Linux动态库版本控制方案	163
10.2.1 基于soname的版本控制方案	163
10.2.2 基于符号的版本控制方案	169
10.3 Windows动态库版本控制	190
10.3.1 DLL版本信息	191
10.3.2 指定DLL版本信息	192
10.3.3 查询并获取DLL版本信息	193
第11章 动态库: 其他主题	202
11.1 插件	202
11.1.1 导出规则	203
11.1.2 一些流行的插件架构	204
11.2 提示和技巧	204
11.2.1 使用动态库的实际意义	204
11.2.2 其他主题	205
第12章 Linux工具集	211
12.1 快速查看工具	211
12.1.1 file实用程序	211
12.1.2 size实用程序	212
12.2 详细信息分析工具	212
12.2.1 ldd	212
12.2.2 nm	214
12.2.3 objdump	215
12.2.4 readelf	223
12.3 部署阶段工具	229
12.3.1 chrpath	229
12.3.2 patchelf	230
12.3.3 strip	231
12.3.4 ldconfig	231
12.4 运行时分析工具	232
12.4.1 strace	232
12.4.2 addr2line	233

- 12.4.3 gdb (GNU调试器) 233
- 12.5 静态库工具 234
- 第13章 平台实践 238
 - 13.1 链接过程调试 238
 - 13.2 确定二进制文件类型 239
 - 13.3 确定二进制文件入口点 240
 - 13.3.1 获取可执行文件入口点 240
 - 13.3.2 获取动态库入口点 240
 - 13.4 列出符号信息 241
 - 13.5 查看节的信息 242
 - 13.5.1 列出所有节的信息 242
 - 13.5.2 查看节的信息 242
 - 13.6 查看段的信息 243
 - 13.7 反汇编代码 244
 - 13.7.1 反汇编二进制文件 244
 - 13.7.2 反汇编正在运行的进程 244
 - 13.8 判断是否为调试构建 244
 - 13.9 查看加载时依赖项 245
 - 13.10 查看装载器可以找到的库文件 245
 - 13.11 查看运行时动态链接的库文件 245
 - 13.11.1 strace实用程序 245
 - 13.11.2 LD_DEBUG环境变量 246
 - 13.11.3 /proc//maps文件 246
 - 13.11.4 lsof实用程序 247
 - 13.11.5 通过编程方式查看 248
 - 13.12 创建和维护静态库 251
- 第14章 Windows工具集 252
 - 14.1 库管理器 (lib.exe) 252
 - 14.1.1 使用lib.exe处理静态库 253
 - 14.1.2 使用lib.exe处理动态库 (导入库生成工具) 257
 - 14.2 dumpbin实用程序 258
 - 14.2.1 确定二进制文件类型 258
 - 14.2.2 查看DLL的导出符号 258
 - 14.2.3 查看节的信息 259
 - 14.2.4 反汇编代码 262
 - 14.2.5 确定是否使用了调试模式构建 263
 - 14.2.6 查看加载时依赖项 265
 - 14.3 Dependency Walker工具 265
 - • • • • [\(收起\)](#)

[高级C/C++编译技术_下载链接1](#)

标签

C/C++

编译原理

C++

计算机

编程

软件开发

计算机科学-C/C++

程序设计

评论

信息量还没国内那本大，错误也不少，不过可能是翻译编辑问题

很一般的书，谈不上高级，很多工具和工程经验可以参照

除了翻译以外，吐血推荐！

还可以，市面上c++的此类书很少

一本好书，看完就可以照着代码练习了，有折扣而且有货的时候可以入一本。

讲真，这种类型的书对于我现在没有太大意义了。顶多就是遇到问题查阅下细节。不如《程序员的自我修养》那本书好。

内存结构与文件结构、动态库静态编译算是清除了 翻译的稍微有点儿奇怪 瑕不掩瑜

动态库链接方面讲的比较好

我一直不清楚程序究竟是如何运行起来的，很早以前我看书知道，C/C++要经过预处理，编译，汇编，链接等步骤才能得到最终的可执行文件。这些年因为一直从事在应用层面编程，对这些也没有做过研究。最近深究 C++ 和 Unix 环境，我觉得很有必要了解 C++ 的编译和链接都做了些什么，以及操作系统如何加载可执行文件的。这本书，正好是我需要的。本书讲解二进制文件的组成部分，编译和链接的过程，静态链接和动态链接的原理和实现。写的很不错，只要了解虚拟存储器，具有一点 C 和 C++ 的基础，知道汇编语言是咋回事，这本书就很容易读懂，没有什么难点。有人提到翻译的问题，我觉得问题不大，至少我在读的时候，没有发现那里因翻译的不好对理解有阻碍。我确实发现了一些翻译错误，仔细读d都能轻易发现这些错误。

1-主要对C/C++的链接装载技术细节进行了阐述，以Linux平台的ELF为主，也介绍了Windows平台的一些工具与技术细节。和侧重理论和细节分析的《链接器与装载器》一书不同，该书偏重于对概念进行形象的阐述，并介绍一些具体的工具和技术的使用方法与注意事项。而且《链接器与装载器》很多内容过于陈旧（虽然讲了许多源头性的概念与技术），相比来说此书则非常贴近现在的程序开发认知的。
2-内容有点啰嗦，繁琐，图很大，知识点不密集。第8、9章是最精华的部分。
3-两天看完（我是多年cpp程序员）

翻译真心烂，很多地方都不知所云。

收益很大，有一部分还是没明白。
详细从底层说明了程序设计，从最初的设计，一步一步说明如何修改，很好的一本书

书中讲得浅显易懂，是本优秀的书

刚转行时读的--没怎么读懂

[高级C/C++编译技术_下载链接1_](#)

书评

我只看了三章试读,因为原书略口水,看不下去. 第三章3.1第三段第二行. 译文:
这么做的原因很可能是新进程的内存映射与shell的内存映射完全不同.
其实应该是:因为很可能新进程的存储map和shell的存储map几乎没有什么相同的地方.
42页中间部分涉及到一个kernel里的struct名字,我...

[高级C/C++编译技术_下载链接1_](#)