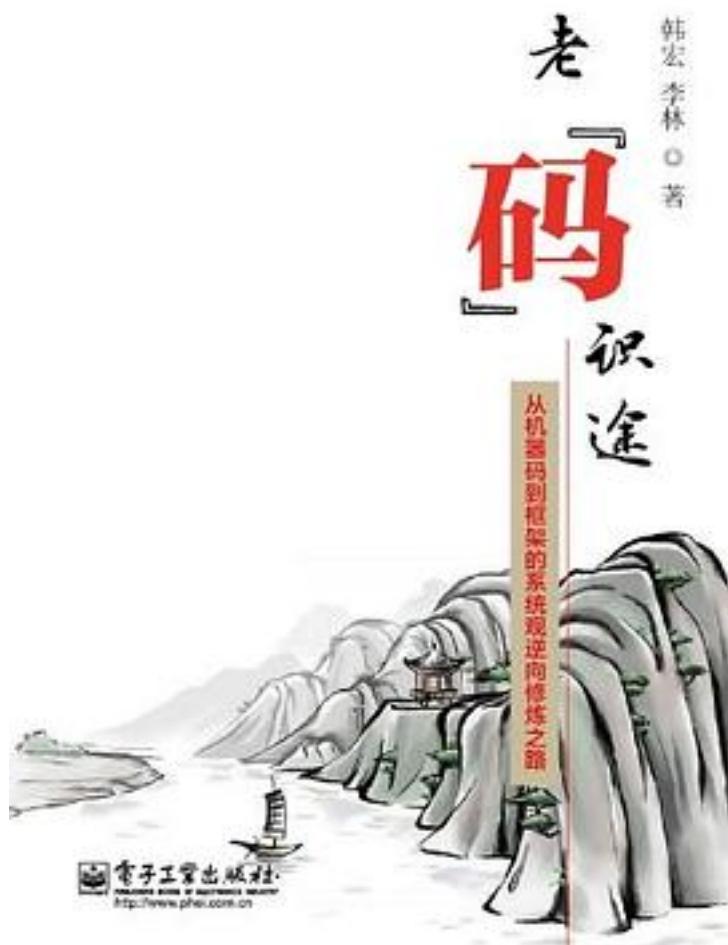


老码识途



[老码识途 下载链接1](#)

著者:韩宏

出版者:电子工业出版社

出版时间:2012-8

装帧:

isbn:9787121173820

《老"码"识途:从机器码到框架的系统观逆向修炼之路》以逆向反汇编为线索，自底向上

，从探索者的角度，原生态地刻画了对系统机制的学习，以及相关问题的猜测、追踪和解决过程，展现了系统级思维方式的淬炼方法。该思维方式是架构师应具备的一种重要素质。《老"码"识途:从机器码到框架的系统观逆向修炼之路》内容涉及反汇编、底层调试、链接、加载、钩子、异常处理、测试驱动开发、对象模型和机制、线程类封装、跨平台技术、插件框架、设计模式、GUI框架设计等。

作者介绍:

目录: 第1章 欲向码途问大道，锵锵bit是吾刀 1

1.1 全局变量引发的故事 2

1.1.1 剖析赋值语句机器码 2

1.1.2 修改赋值语句机器码 6

1.1.3 直接构建新的赋值语句 7

1.1.4 小结 10

1.2 理解指针和指针强制转换 11

1.2.1 指针和它丢失的类型信息 11

1.2.2 指针强制转换 13

1.3 函数调用和局部变量 15

1.3.1 计算指令中的跳转地址 15

1.3.2 返回故乡的准备 16

1.3.3 给函数传递参数 17

1.3.4 函数获取参数 18

1.3.5 局部变量 20

1.3.6 返回故乡 20

1.3.7 返回点什么 23

1.3.8 扫尾工作 28

1.3.9 调用惯例 30

1.3.10 函数指针 31

1.4 数组、结构体 34

1.4.1 数组 34

1.4.2 结构体 35

1.5 无法沟通——对齐的错误 37

1.5.1 结构体对齐 37

1.5.2 无法沟通 41

1.6 switch语句的思考 44

1.6.1 switch机制探索 45

1.6.2 switch语句一定比if-else语句快吗 50

1.6.3 switch的再次优化 50

1.7 关于其他高级语言要素的反汇编学习 54

1.8 全局变量的疑问——重定位和程序结构 54

1.8.1 独一无二的全局变量？ 54

1.8.2 不变的地址和重定位 56

1.8.3 动态链接库中的重定位 64

1.9 汇编的学习之路——阅读RTL 68

1.10 程序设置说明 76

习题1 76

第2章 庖丁解“码”：底层的力量与乐趣 79

2.1 解密之hello world 80

2.2 奇怪的死循环 83

2.3 我们都犯过的错——指针的指针 85

2.4 互通的障碍（跨语种调用） 87

2.5 错误的目的地 90

2.6 网络发送出错了 91
2.7 为什么代码运行完毕却出错 93
2.8 失效的管道 96
2.8.1 管道的力量 97
2.8.2 我要控制Telnet客户端 101
2.8.3 不是所有管道都可抽象等价 101
2.8.4 一动不动的48小时 103
2.9 异常世界历险记 112
2.9.1 学习基础概念 112
2.9.2 如何返回 113
2.9.3 那些状态保存到哪里了 117
2.9.4 意外的秘密 120

习题2 127

第3章 成长：与程序一起茁壮 131
3.1 初写系统 132
3.1.1 代码风格 132
3.1.2 常量 133
3.1.3 最简单的电话簿（1）：功能设计和相关库函数学习 134
3.1.4 最简单的电话簿（2）：系统实现，分割函数 141
3.1.5 空字符串结尾的警觉 143
3.1.6 让程序更有组织性 144
3.2 有序的世界：可测试与跟踪的系统 146
3.2.1 电话簿扩展（1）：硬盘结构体数组 146
3.2.2 指针的陷阱 148
3.2.3 动态数组 149
3.2.4 变化的压力与出路：重构、单元测试和日志 151
3.2.5 电话簿扩展（2）：可测试的恩赐 155

3.3 优雅的积木 155
3.3.1 可复用硬盘数组 155
3.3.2 分享它（1）：理解编译链接过程 161
3.3.3 分享它（2）：我的丑陋链接器 167
3.3.4 分享它（3）：静态链接库 173
3.3.5 分享它（4）：动态链接库 175
3.3.6 积木的艺术 178

习题3 182

第4章 让我们创造面向对象语言吧 185
4.1 “封装”数据函数合一，陈仓暗度this传递 186
4.1.1 那些讨厌的事 186
4.1.2 像芯片一样工作（1）：数据合一 187
4.1.3 像芯片一样工作（2）：行为与数据合一 188
4.1.4 不想让你传递“自己” 189
4.1.5 创造吧，新的语言 190
4.1.6 是这样吗？我们需要证明 191
4.2 太麻烦了，需要更简单的创造与销毁 194
4.2.1 创造构造和析构函数 194
4.2.2 构造中分配资源，析构中释放资源 197
4.3 对比C语言的“对象”和面向对象 199
4.4 体验封装的力量 201
4.4.1 生死原点，整体资源管理 201
4.4.2 文件流 203
4.5 整体资源管理的爱恨 204
4.5.1 扩展技巧：保证成对出现，巧妙的自动线程锁 204
4.5.2 美丽的幻影：不可靠的自动析构 205
4.5.3 隐藏的敌人：不请而至的析构和拷贝构造 206

4.6 封装之强化：内外之别，亲疏之分	209
4.6.1 私有的诞生	209
4.6.2 私有？阻止不了我	210
4.6.3 理解继承的机制（1）：模型	211
4.6.4 理解继承的机制（2）：在C语言中“玩”继承	214
4.6.5 保护的诞生	218
4.7 “变”的烦恼与出路：创造虚函数	218
4.7.1 “三变”之苦：格式化字符串	218
4.7.2 函数指针，请带我走出不断修改的泥潭	220
4.7.3 再进一步：做成对象	221
4.7.4 我们需要性能更好的版本	223
4.7.5 我们需要新语法，创造虚函数吧	225
4.7.6 验证虚表机制（1）：反汇编分析	226
4.7.7 验证虚表机制（2）：直接用虚表来调用虚函数	227
4.8 虚函数的那些事儿	227
4.8.1 理解“=”	227
4.8.2 纯虚函数，从dll导入对象	230
4.8.3 C语言实现虚函数	231
4.8.4 魂归何处：析构之“虚”	232
4.8.5 理解运行期类型判断dynamic_cast	232
4.9 静态覆盖	235
4.10 静态与非静态成员函数的区别	235
4.11 遥远的风景：管窥.NET对象	235
习题4	236
第5章 底层与抽象的混沌：一个跨平台线程类的封装、错误与进化	239
5.1 先学习多线程编程吧	240
5.1.1 概念	240
5.1.2 Windows下的线程接口	240
5.1.3 第一个线程程序	242
5.1.4 那些复杂的参数和bug	243
5.2 简单、重用，让我们构造线程类吧	247
5.2.1 无赖的尝试，原来是它—static	248
5.2.2 可爱的virtual和可恨的this	249
5.2.3 私有、保护、公有、只读、纯虚函数，一个都不能少	251
5.2.4 析构中释放资源	252
5.2.5 我们发现了一个设计模式	252
5.2.6 我关心，你通知——我们的第二个设计模式	253
5.3 跨平台的线程设计	255
5.3.1 讨厌的Linux版本	255
5.3.2 源代码跨平台技术	256
5.3.3 跨平台的版本	257
5.4 崩溃，哪里出错了	262
5.4.1 寻找错误	262
5.4.2 C++下整体资源管理的反思	265
5.4.3 生生死死虚表误，剥离策略世界殊——重生	267
习题5	268
第6章 插件养成记	271
6.1 一个修改已有功能的实例	272
6.2 一个可以动态添加功能的简单实例	273
6.3 一个可以动态添加功能的复杂实例	276
6.4 从函数到插件对象	280
6.5 delete的灾难：谁的书	283
6.5.1 释放内存的崩溃	283
6.5.2 解决之道：新生活，各管各	288

习题6 291

第7章 天堂的阶梯 293

7.1 遥望天堂，那些美丽与简洁我向往 294

7.2 从最基础开始吧，SDK编写窗体程序 295

7.2.1 hello window和基本原理 295

7.2.2 来个复杂点的窗体程序 298

7.3 构建我的GUI组件（1）：简单组件 300

7.4 构建我的GUI组件（2）：天堂的机器码跳板 304

7.4.1 调试，我要看清你 304

7.4.2 我们的自定位代码 313

7.4.3 自定位代码版Button类 314

7.4.4 自定位代码版Form类 315

7.4.5 为什么不错呢 316

7.5 构建我的GUI组件（3）：更多的组件 317

7.6 天堂阶梯，玩赏框架那如花散落的繁复与如索串珠的简洁之美 319

7.7 构建我的GUI组件（4）：我的天堂 326

7.8 他们的天堂 330

习题7 332

• • • • • (收起)

[老码识途](#) [下载链接1](#)

标签

计算机

逆向工程

编程

计算机底层

汇编

程序设计

反汇编

操作系统

评论

勘误表 http://blog.sina.com.cn/s/blog_7d5a09f90101dukr.html

我们《软件开发环境》老师写的书，先教你通过反汇编来分析、修改、自己写底层机器码，后面着重探讨面向对象特性在底层的实现和体现。

知识点都是底层的干货，对理解高层封装出来的一些概念的本质非常有帮助。比如指针本质上就是个4字节的地址，指针类型只是由编译器识别，然后体现在控制访问多少个字节的CPU指令上；

比如函数是怎么实现调用、传参、返回的，传参又有寄存器传值、压栈传值、压栈传地址等方式，跨语言调用函数时调用惯例的协调。

总之弄懂了这些底层的机制，对高层语言的理解会透彻很多。

不过最好有一点汇编基础再读，否则略艰涩。

另一个特点是全书一直贯彻一种“猜测——实证”的思想，跟作者交流过这本书好几次，感觉这种思想是他最想传达的东西。

要有汇编基础！！！

这是一本非常需要带着批判思维阅读的书。

表述差劲，而且有马后炮嫌疑，但是视角不错

内功方面的书

很用心，都是干货，强烈推荐

有些是“标题党”，例如使用c创建面向对象语言，插件调用等，总体3星

从汇编角度来看C的各类功能

对于了解编码底层知识非常有效。

内容跨度太大，不适合用来深度学习，拿来看看，作为了解和扩展思路比较好。

很好的书，可惜还没看完就要还了，打底层基础必备

深入理解C语言，很好的书。

喜欢这种学习方式，从机器码分析，相当于拿第一手资料破案，你的推理、猜测都能第一时间得到验证。这也避免了，只学概念，得不到求证，久之产生了困惑。

看了下三星作为鼓励新手可以看看 深度不够

国内少见的程序员必读书籍。适合有一定基础的C程序员看，最好对汇编语言有一些了解。总之，这本书66的。

去年看过很好的一本书之一

被名字吸引，本以为是逆向工程，结果只是简单的语言到机器码，看完还是写不出shell code。不过对于那些刚入门的小白应该够了

[老码识途](#) [下载链接1](#)

书评

看完第一章时的感觉是惊喜，但是看完整本书之后就只剩下还行了。前不久读着读着实在忍不住了就在饭否上吐槽到：『(作者) 沉迷于用底层的方法去分析，但是又囿于各种细枝末节；到了高处又缺乏一定的核心抽象部分的分析』。书中很多问题看似用底层分析的方法解决得比较巧妙，但...

我们《软件开发环境》老师写的书，先教你通过反汇编来分析、修改、自己写底层机器码，后面着重探讨面向对象特性在底层的实现和体现。
知识点都是底层的干货，对理解高层封装出来的一些概念的本质灰常有帮助。比如指针本质上就是个4字节的地址，指针类型只是由编译器识别，然后...

这几年，有过几本类似的书，希望把c语言讲明白。但是，没有达到这本书的程度。
这是一本含有真知灼见的计算机学习指南，可以把你从码农变成系统工程师。
如果你不想成为一位码农，无论是刚开始接触计算机，还是已经工作了，如果对计算机一知半解，拿起这本书从头到尾好好读一遍...

[老码识途](#) [下载链接1](#)