第一图书网, tushu007.com

<<老码识途>>

图书基本信息

书名：<<老码识途>>

13位ISBN编号：9787121173820

10位ISBN编号：7121173824

出版时间：2012-8

出版时间：电子工业出版社

作者：韩宏

页数：344

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

第一图书网, tushu007.com

内容概要

本书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。
本书内容涉及反汇编、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。
书中包含不少工业级或非公开案例。
读者不仅能以底层观和调试技巧解决各种实际问题；还可掌握一套学习方法，如“猜测—实证—构建”，调构学习法。

书籍目录

"第1章 欲向码途问大道，锵锵bit是吾刀
1.1 全局变量引发的故事
1.1.1 剖析赋值语句机器码
1.1.2 修改赋值语句机器码
1.1.3 直接构建新的赋值语句
1.1.4 小结
1.2 理解指针和指针强制转换
1.2.1 指针和它丢失的类型信息
1.2.2 指针强制转换
1.3 函数调用和局部变量
1.3.1 计算指令中的跳转地址
1.3.2 返回故乡的准备
1.3.3 给函数传递参数
1.3.4 函数获取参数
1.3.5 局部变量
1.3.6 返回故乡
1.3.7 返回点什么
1.3.8 扫尾工作
1.3.9 调用惯例
1.3.10 函数指针
1.4 数组、结构体
1.4.1 数组
1.4.2 结构体
1.5 无法沟通——对齐的错误
1.5.1 结构体对齐
1.5.2 无法沟通
1.6 switch语句的思考
1.6.1 switch机制探索
1.6.2 switch语句一定比if-else语句快吗
1.6.3 switch的再次优化
1.7 关于其他高级语言要素的反汇编学习
1.8 全局变量的疑问——重定位和程序结构
1.8.1 独一无二的全局变量？

1.8.2 不变的地址和重定位
1.8.3 动态链接库中的重定位
1.9 汇编的学习之路——阅读RTL
1.10 程序设置说明
习题1
第2章 庖丁解“码”：底层的力量与乐趣
2.1 解密之hello world
2.2 奇怪的死循环
2.3 我们都犯过的错——指针的指针
2.4 互通的障碍（跨语种调用）
2.5 错误的目的地

2.6 网络发送出错了
2.7 为什么代码运行完毕却出错
2.8 失效的管道
2.8.1 管道的力量
2.8.2 我要控制Telnet客户端
2.8.3 不是所有管道都可抽象等价
2.8.4 一动不动的48小时
2.9 异常世界历险记
2.9.1 学习基础概念
2.9.2 如何返回
2.9.3 那些状态保存到哪里了
2.9.4 意外的秘密
习题2
第3章 成长：与程序一起茁壮
3.1 初写系统
3.1.1 代码风格
3.1.2 常量
3.1.3 最简单的电话簿（1）：功能设计和相关库函数学习
3.1.4 最简单的电话簿（2）：系统实现，分割函数
3.1.5 空字符结尾串的警觉
3.1.6 让程序更有组织性
3.2 有序的世界：可测试与跟踪的系统
3.2.1 电话簿扩展（1）：硬盘结构体数组
3.2.2 指针的陷阱
3.2.3 动态数组
3.2.4 变化的压力与出路：重构、单元测试和日志
3.2.5 电话簿扩展（2）：可测试的恩赐
3.3 优雅的积木
3.3.1 可复用硬盘数组
3.3.2 分享它（1）：理解编译链接过程
3.3.3 分享它（2）：我的丑陋链接器
3.3.4 分享它（3）：静态链接库
3.3.5 分享它（4）：动态链接库
3.3.6 积木的艺术
习 题 3
第4章 让我们创造面向对象语言吧
4.1 “ 封装 ” 数据函数合一，陈仓暗度this传递
4.1.1 那些讨厌的事
4.1.2 像芯片一样工作（1）：数据合一
4.1.3 像芯片一样工作（2）：行为与数据合一
4.1.4 不想让你传递“ 自己 ”
4.1.5 创造吧，新的语言
4.1.6 是这样吗？
我们需要证明
4.2 太麻烦了，需要更简单的创造与销毁
4.2.1 创造构造和析构函数
4.2.2 构造中分配资源，析构中释放资源

4.3 对比C语言的“对象”和面向对象
4.4 体验封装的力量
4.4.1 生死原点，整体资源管理
4.4.2 文件流
4.5 整体资源管理的爱恨
4.5.1 扩展技巧：保证成对出现，巧妙的自动线程锁
4.5.2 美丽的幻影：不可靠的自动析构
4.5.3 隐藏的敌人：不请而至的析构和拷贝构造
4.6 封装之强化：内外之别，亲疏之分
4.6.1 私有的诞生
4.6.2 私有？
阻止不了我
4.6.3 理解继承的机制（1）：模型
4.6.4 理解继承的机制（2）：在C语言中“玩”继承
4.6.5 保护的诞生
4.7 “变”的烦恼与出路：创造虚函数
4.7.1 “三变”之苦：格式化字符串
4.7.2 函数指针，请带我走出不断修改的泥潭
4.7.3 再进一步：做成对象
4.7.4 我们需要性能更好的版本
4.7.5 我们需要新语法，创造虚函数吧
4.7.6 验证虚表机制（1）：反汇编分析
4.7.7 验证虚表机制（2）：直接用虚表来调用虚函数
4.8 虚函数的那些事儿
4.8.1 理解“=”
4.8.2 纯虚函数，从dll导入对象
4.8.3 C语言实现虚函数
4.8.4 魂归何处：析构之“虚”
4.8.5 理解运行期类型判断dynamic_cast
4.9 静态覆盖
4.10 静态与非静态成员函数的区别
4.11 遥远的风景：管窥.NET对象
习题4
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
5.1 先学习多线程编程吧
5.1.1 概念
5.1.2 Windows下的线程接口
5.1.3 第一个线程程序
5.1.4 那些复杂的参数和bug
5.2 简单、重用，让我们构造线程类吧
5.2.1 无赖的尝试，原来是它——static
5.2.2 可爱的virtual和可恨的this
5.2.3 私有、保护、公有、只读、纯虚函数，一个都不能少
5.2.4 析构中释放资源
5.2.5 我们发现了一个设计模式
5.2.6 我关心，你通知——我们的第二个设计模式
5.3 跨平台的线程设计

5.3.1 讨厌的Linux版本
5.3.2 源代码跨平台技术
5.3.3 跨平台的版本
5.4 崩溃，哪里出错了
5.4.1 寻找错误
5.4.2 C++下整体资源管理的反思
5.4.3 生生死死虚表误，剥离策略世界殊——重生
习题5
第6章 插件养成记
6.1 一个修改已有功能的实例
6.2 一个可以动态添加功能的简单实例
6.3 一个可以动态添加功能的复杂实例
6.4 从函数到插件对象
6.5 delete的灾难：谁的书
6.5.1 释放内存的崩溃
6.5.2 解决之道：新生活，各管各
习题6
第7章 天堂的阶梯
7.1 遥望天堂，那些美丽与简洁我向往
7.2 从最基础开始吧，SDK编写窗体程序
7.2.1 hello window和基本原理
7.2.2 来个复杂点的窗体程序
7.3 构建我的GUI组件（1）：简单组件
7.4 构建我的GUI组件（2）：天堂的机器码跳板
7.4.1 调试，我要看清你
7.4.2 我们的自定位代码
7.4.3 自定位代码版Button类
7.4.4 自定位代码版Form类
7.4.5 为什么不错呢
7.5 构建我的GUI组件（3）：更多的组件
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
7.7 构建我的GUI组件（4）：我的天堂
7.8 他们的天堂
习题7

章节摘录

版权页： 插图： 那么，登记窗体句柄和处理函数的对应关系有两种方法。
①在窗体描述结构WNDCLASS中指出。
窗体千差万别，如果让系统构造一个窗体，需要给系统一份窗体图纸。
该图纸就是WNDCLASS结构体，其中有两个重要的成员变量，一个是DM7—2的第l3行的lpszClassName，标识这份图纸的名字，例中给定名字为“ testwin ”。
后面构造窗体时，就用这个名字告知系统我们想用哪份图纸构造，第15行CreateWindow构造窗体，其第1个参数就是“ testwin ”，指出了图纸的名字。
要用这份图纸自然需要将它注册给系统，所以第14行RegisterClass完成了该工作。
WNDCLASS结构体另一个成员是第l2行的lpfnWndProc，指出了窗体处理函数的地址，指示的函数就是第1～9行的函数WndProc（）。
这种方法指出窗体的处理函数，意味着所有用名为“ testwin ”的WNDCLASS构造出的窗体都拥有同样的处理函数，如果构造了多个窗体，如何让函数区分到底当前在处理哪个窗体的消息？
DM7—2中，第1行窗体处理函数的第1个参数hwnd指出了被处理消息属于哪个窗体。
只要在函数中用判断hwnd的不同，即可做出相应处理，这样一个窗体函数可被多个窗体共用，却又能各行其是。
②第①种方法为某类型窗体指定相同处理函数，这种方法可为某个具体窗体对象指定处理函数。
也就是说，可以用同一个WNDCLASS结构创建出的窗体拥有各自不同的处理函数。
调用函数SetWindowLong（）即可，SetWindowLong（hwnd，GWL WNDPROC，（LONG）wndProc）将消息处理函数指针wndProc设定给句柄hwnd指示的窗体。
消息处理函数怎样处理消息？
从DM7—2第1行可知，它有4个参数。
第1个已解释，是接收消息的窗体的句柄。
第2个是无符号整数，代表消息类型，如宏WM_PAINT就是重绘消息。
从winuser.h中可知，该宏声明为#defme WM_PAINT 0x000F，即说明十进制数15代表重绘消息。
一个消息可能包含相关参数，则由第3、4个参数指出，其类型为指针，但很多时候被直接当成整数。
比如，鼠标左键按下消息WM_LBUTTONDOWN中，wParam的整数值代表了此时其他键按下的状态，如MK SHIFT代表键盘Shift键是否按下。
读者可能会问，如果有2个甚至以上的参数需要表示怎么办？
系统非常节约，比如还是WM_LBUTTONDOWN消息，需要包含此时鼠标的坐标x和y，它用1Param的高2字节代表y，低2字节代表X。
如果还有更多信息呢？
因为wParam和lParam都声明为指针，那么就可以将任何结构体的指针作为wParam或1Param传递即可。
这与第5章线程函数的参数只接收一个void*类型却可传递任何信息是一样的。

编辑推荐

《老"码"识途:从机器码到框架的系统观逆向修炼之路》包含不少工业级或非公开案例，读者不仅能以底层观和调试技巧解决各种实际问题；还可掌握一套学习方法，如“猜测—实证—构建”，调构学习法。

第一图书网, tushu007.com

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com