万水计算机编程技术与应用系列

Windows 多线程编程技术与实例

郝文化 主 编
文自勇 王浩强 曹华伟 等编著

中国水利水电出版社
www.waterpub.com.cn

万水计算机编程技术与应用系列

Windows 多线程编程技术与实例

郝文化　主　编

文自勇　王浩强　曹华伟　等编著

中国水利水电出版社

内 容 提 要

本书通过众多实例介绍了如何实现 Windows 下的多线程编程，既重点介绍了 Win32 API 下的多线程编程和 MFC 下的多线程编程，又介绍了多线程机制在网络编程、图形编程和数据库中的应用。本书每一章都从简单的多线程实例出发逐渐深入，紧紧围绕应用程序实例，向读者展示了利用多线程技术来编写高效、友好的 Windows 应用程序的方法，并对常用的 Win32 线程函数进行了深入详细的说明。本书共分 8 章，第 1 章介绍了多线程编程的基础知识；第 2～5 章通过实例阐明 Win32 下多线程的几种不同实现形式及多进程的实现机制，这是本书介绍的重点内容，也是读者学习后面几章内容所必须掌握的基础知识；第 6～8 章介绍了多线程技术在网络、图形处理和数据库中的应用。

本书语言通俗易懂，内容丰富翔实，突出了以实例为中心的特点，既适合具有一定 C++和 VC 编程基础的高校相关专业学生选作多线程编程的学习用书，也适用于具有一定实际编程经验的中高级开发人员作为学习多线程编程思想的自学用书。

本书源代码可以到中国水利水电出版社网站上下载，网址：www.waterpub.com.cn/softdown/。

图书在版编目（CIP）数据

Windows 多线程编程技术与实例 / 郝文化主编. —北京：中国水利水电出版社，2005

（万水计算机编程技术与应用系列）

ISBN 7-5084-3316-5

Ⅰ. W… Ⅱ. 郝… Ⅲ. 窗口软件，Windows—程序设计 Ⅳ. TP316.7

中国版本图书馆 CIP 数据核字（2005）第 114417 号

书　　名	Windows 多线程编程技术与实例
作　　者	郝文化　主编　文自勇　王浩强　曹华伟　等编著
出版　发行	中国水利水电出版社（北京市三里河路 6 号　100044）
	网址：www.waterpub.com.cn
	E-mail：mchannel@263.net（万水）
	sales@waterpub.com.cn
	电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京北医印刷厂
规　　格	787mm×1092mm　16 开本　17 印张　332 千字
版　　次	2005 年 10 月第 1 版　2005 年 10 月第 1 次印刷
印　　数	0001—5000 册
定　　价	28.00 元

前　言

在各种各样的项目开发中，多线程可谓是无处不在，而这恰恰又是项目的难点。Microsoft 32 位平台的多线程技术是从事 Windows 应用程序开发的程序员必备的知识。在 Win32 下多线程的实现方式很多，可以利用 C Run-time Library、Windows API 或 MFC 来实现；实现形式的多样化给我们编程带来了很大的灵活性，但同时也使 Win32 下的多线程编程更趋复杂。

市场对程序员的需求很大，但需要的是真正的程序员，绝不是只会 C++语法却不能与实际项目结合的“程序机”，现在，计算机专业本科毕业生都很难找到合适的工作就是其最好的例证。勿庸置疑，多线程与组件编程已成为 IT 的热点，但介绍多线程的书并不多，即使有也是太偏重于原理知识，书中的实例少且过于简单。正是针对这种现状，作者才着手编写了本书。本书从实例开发入手，深入浅出地阐明了多线程编程机制，使读者能真正抓住程序的千头万绪，从而成为真正的程序员。

本书通过众多实例介绍了如何实现 Windows 下的多线程编程，既重点介绍了多线程机制在 Win32 API 下的多线程编程和 MFC 下的多线程编程，又介绍了多线程机制在网络编程、图形编程和数据库中的应用。

本书共分 8 章，第 1 章介绍了多线程编程的基础知识；第 2～5 章通过实例阐明 Win32 下多线程的几种不同实现形式及多线程的实现机制，这是本书介绍的重点内容，也是读者学习后面几章内容所必须掌握的基础知识；第 6～8 章介绍了多线程技术在网络、图形处理和数据库中的应用。

本书由郝文化主编，主要由文自勇、王浩强和曹华伟编写。另外，参加本书编写工作的还有邹素琼、王安贵、陈郭宜、程小英、谭小丽、卢丽娟、刘育志、吴淬砺、赵明星、贺洪俊、李小平、史利、张燕秋、周林英、黄茂英、李力、李小琼、李修华、田茂敏、苏萍、巫文斌、邹勤、粟德容、童芳、李中全、蒋敏、刘华菊、袁媛、李建康等。

由于时间仓促，书中疏漏之处在所难免，恳请广大读者和同行批评指正。

如果读者愿意参加《Windows 多线程技术实例》的学习培训，或是在学习过程中发现问题，或有更好的建议，欢迎致函。同时，我们也非常愿意随时同 Windows 多线程编程的爱好者保持经常的联系，E-mail：bojia@bojia.net，网址：http://www.bojia.net，我们将认真、负责地对待每位读者的来信。

编　者

2005 年 7 月

目　　录

第 1 章　线程编程基础知识

本章简单介绍在 Win32 环境下多线程编程的基础知识，分析了进程和线程的特征与异同，并对进程和线程以直观的形式给读者以清晰的认识。另外还详细分析了使用多线程的优越性以及在什么情况下使用多线程，并对线程的同步和多线程的性能进行了详细分析。

1.1　进程与线程

1.1.1　进程与线程的概念

进程是指在系统中正在运行的一个应用程序，在传统的操作系统中，是资源的分配单位又是调度运行的单位。在现代操作系统中，进程是资源的分配单位，一个进程通常定义为程序的一个实例。在 32 位 Windows 中，进程占据 4GB 的虚拟地址空间。与它们在 MS-DOS 和 16 位 Windows 操作系统中不同，32 位 Windows 进程是没有活力的，这就是说，一个 32 位 Windows 进程并不执行什么指令，它只是占据着 4GB 的地址空间，此空间中有应用程序 EXE 文件的代码和数据。EXE 需要的 DLL 也将它们的代码数据装入到进程的地址空间。除了地址空间，进程还占有某些资源，如文件、动态内存分配和线程，当进程终止时，在其生命期中创建的各种资源将被清除，所以说进程是没有活力的，它只是一个静态的概念。

线程是指进程中执行运行的最小单位，即处理机调度的基本单位。和进程相比，进程是一项任务，线程是独立子任务，可由不同处理器分别来完成，提高了进程运行速度。在 Win32 编程环境下，每个运行的应用程序都构成一个进程，而每个进程都包含一个或多个执行线程。“线程”是指程序代码的执行途径外加一组操作系统分配的资源（堆栈、寄存器状态等）。

由此看来，线程是系统分配处理器时间资源的基本单元或者是进程内独立执行的一个单元，对于操作系统而言，其调度单元是线程。一个进程至少包括一个线程，通常将该线程称为主线程。一个进程从主线程的执行开始进而创建一个或多个附加线程就是所谓的基于多线程的多任务。

为了让进程完成一些工作，进程必须至少占有一个线程，所以线程描述进程内的执行，正是线程负责执行包含左进程地址空间中的代码。实际上，单个进程可能包含

几个线程，它们可以同时执行进程地址空间中的代码。为了做到这一点，每个线程有自己的一组CPU寄存器和堆栈，一个进程至少有一个线程在执行其地址空间中的代码，如果没有线程执行进程地址空间中的代码，进程也就没有继续存在的理由，系统将自动清除进程及其地址空间。为了运行所有这些线程，操作系统为每个独立线程安排一些CPU时间，操作系统以轮转方式向线程提供时间片，这就给人一种假象，好象这些线程都在同时运行。创建一个32位Windows进程时，它的第一个线程称为主线程，由系统自动生成，然后可由此主线程生成额外的线程，这些线程又可生成更多的线程。

1.1.2 进程与线程的比较

在Win32中，进程拥有内存和资源，内存理论上可以达到4GB。资源包括核心对象（如file handles和线程）、USER资源（如对话框和字符串）、GDI资源（如Device Context和brushes）。进程本身并不能执行，而只是提供一个安置内存和线程的地方。一旦CPU开始执行代码，就有了一个线程，所以有可能是许多线程执行同一段代码。线程在任何时刻下的状态被定义在进程的某块内存以及CPU寄存器中，而变量和应用程序的调用堆栈存储在可以被其他线程共享的内存中。

进程与线程的区别到底是什么？进程是执行程序的实例。例如，当用户运行记事本程序（Nodepad）时，用户就创建了一个用来容纳组成Notepad.exe的代码及其所需调用动态链接库的进程。每个进程均运行在其专用且受保护的地址空间内，因此，如果用户同时运行记事本的两个复制，该程序正在使用的数据在各自实例中是彼此独立的。在记事本的一个复制中将无法看到该程序的第二个实例打开的数据。

以生产线为例进行阐述。一个进程就好比一条生产线。线程就如同生产线上负责相同或者不同任务的工人们。工人们在生产线中独立完成自己的工作，并且不同工人之间相互协作，相互沟通。不同生产线之间的相互合作和沟通不妨看成是不同进程之间的相互协作和沟通，不同生产线上的工人之间的联络便可以看成是不同进程间线程的相互联系了。然而，每个进程就像一个被保护起来的生产线，有自己的资源、地址空间，未经许可，无人可以进出，而同一条线上的工人能够共享生产线上的资源，就像同一进程的不同线程拥有相同的地址空间，能够共享其中一部分资源一样。

1.1.3 为什么用线程而不用进程

数据表明，线程是更加经济的，启动快、退出快、对系统资源冲击少，在Win32中，线程分享了大部分核心对象，如文件句柄的拥有权。启动一个进程，必须为进程分配内存，初始化，有太多工作要做。

因为多线程可以独立运行，所以用多线程编程可以：

（1）提高应用程序响应。利用多线程编程并不一定能加快程序运行的速度，其目

的是拥有更加良好的程序响应。任何一个包含很多互不关联的操作的程序都可以被重新设计，使得每一个操作成为一个线程。例如，在一个 GUI（图形用户界面）内执行一个操作的同时启动另外一个，则可以用多线程改善性能。

（2）使多 CPU 系统更加有效。典型情况下，有同时性需求的多线程应用程序不需要考虑处理器的数量。应用程序的性能在被多处理器改善的同时对用户是透明的。对于数学计算和有高度并发性需求的应用程序，如矩阵乘法，在多处理器平台上可以用多线程来提高速度。

（3）改善程序结构。许多应用程序可以从一个单一的、巨大的线程改造成一些独立或半独立的执行部分，从而得到更有效的运行。多线程程序比单线程程序更能适应用户需求的变更。

（4）占用更少的系统资源。线程的数据结构简单，停止、运行速度快，进程是线程的靠山，是车间，线程是小组，活动方便。进程任务大、信息多，将它细分为线程后，各线程可共享进程分到的资源，调度方便。

（5）改善性能。改善性能是把多线程和 RPC（Remote Procedure Call，远程过程调用）结合起来，用户可以使用没有内存共享的多处理器（如一个工作站组）。这种结构把这组工作站当作一个大的多处理器系统，使应用程序分布得更加容易。例如，一个线程可以创建子线程，每一个子进程可以做 RPC，调用另外一台机器上的过程。尽管最早的线程仅仅创建一些并行的线程，这种并行可以包括多台机器的运行。

但是，世界上没有免费的午餐，多线程的编程为了拥有良好的响应性能必然要付出代价。使用线程的代价是用户必须小心设计，且运行不可预测，测试困难。同时运行的多个线程所要求的并行性给程序的编写也添加了困难。如果运用得当，会提高应用程序的响应能力，反之后果也往往是难以预料的。

1.2　直观理解进程和线程

当打开 Windows 2000 的任务管理器时，可以看到应用程序、进程、性能等选项卡，下面分析一下任务管理器里的进程选项卡。

1.2.1　进程选项卡

进程由它们所运行的可执行程序实例来识别，并且通过由 Windows NT 或 Windows 2000 生成进程 ID 来彼此区别，该进程 ID 能循环使用。由于每个进程都是独一无二的，所以如果用户打开多个 DOS 窗口，用户将会看到有多个 cmd.exe 的进程，它们并不具有独立的映射名称，在进程选项卡中可以看到有系统线程和用户界面线程。有些进程能够人为地关闭，有些则不能。

1. 最基本的系统进程

smss.exe：会话管理器进程。

csrss.exe：子系统服务器进程。

winlogon.exe：管理用户登录的进程。

services.exe：包含很多系统服务的进程。

lsass.exe：为系统服务。用于管理 IP 安全策略以及启动 ISAKMP/Oakley（IKE）和 IP 安全驱动程序。

svchost.exe：包含很多系统服务的进程。

spoolsv.exe：为系统服务，将文件加载到内存中以便延迟打印。

2. 基本的用户进程

explorer.exe：资源管理器。

internat.exe：托盘区的拼音图标。

3. 附加的系统进程

这些进程不是必要的，可以根据需要通过服务管理器来增加或减少。

mstask.exe：为系统服务，允许程序在指定时间运行。

regsvc.exe：为系统服务，允许远程注册表操作。

winmgmt.exe：为系统服务，用于提供系统管理信息。

inetinfo.exe：为系统服务，可通过 Internet 信息服务的管理单元提供 FTP 连接和管理。

tlntsvr.exe：为系统服务，允许远程用户登录到系统并使用命令行运行控制台程序。

tftpd.exe：为系统服务，可以实现 TFTP Internet 标准，该标准不要求用户名和密码。是远程安装服务的一部分。

termsrv.exe：为系统服务，提供多会话环境允许客户端设备访问虚拟的 Windows 2000 Professional 桌面会话以及运行在服务器上的基于 Windows 的程序。

dns.exe：为系统服务，应对域名系统（DNS）名称的查询和更新请求。

以上服务都对安全有害，如果不是必要的应该关掉。

4. 很少用到的系统进程

tcpsvcs.exe：为系统服务，提供在 PXE 可远程启动客户计算机上远程安装 Windows 2000 Professional 的能力，支持以下 TCP/IP 服务：Character Generator、Daytime、Discard、Echo 以及 Quote of the Day。

ismserv.exe：为系统服务，允许在 Windows Advanced Server 站点间发送和接收消息。

ups.exe：为系统服务，用于管理连接到计算机的不间断电源（UPS）。

wins.exe：为系统服务，用于为注册和解析 NetBIOS 型名称的 TCP/IP 客户提供

NetBIOS 名称服务。

ntfrs.exe：为系统服务，用于在多个服务器间维护文件目录内容的文件同步。

RsSub.exe：为系统服务，用于控制用来远程存储数据的媒体。

locator.exe：为系统服务，用于管理 RPC 名称服务数据库。

lserver.exe：为系统服务，用于注册客户端许可证。

dfssvc.exe：为系统服务，用于管理分布于局域网或广域网的逻辑卷。

clipsrv.exe：为系统服务，用于支持“剪贴簿查看器”，以便可以从远程剪贴簿查阅剪贴页面。

msdtc.exe：为系统服务，并列事务，分布于两个以上的数据库、消息队列、文件系统或其他事务保护资源管理器。

faxsvc.exe：为系统服务，用于帮助用户发送和接收传真。

cisvc.exe：为系统服务，用于索引服务。

dmadmin.exe：为系统服务，用于磁盘管理请求的系统管理服务。

mnmsrvc.exe：为系统服务，用于允许有权限的用户使用 NetMeeting 远程访问 Windows 桌面。

netdde.exe：为系统服务，可用于提供动态数据交换（DDE）的网络传输和安全特性。

smlogsvc.exe：为系统服务，用于配置性能日志和警报。

rsvp.exe：为系统服务，用于为依赖质量服务（QoS）的程序和控制应用程序提供网络信号和本地通信控制安装功能。

RsEng.exe：为系统服务，用于协调用来存储不常用数据的服务和管理工具。

RsFsa.exe：为系统服务，用于管理远程存储文件的操作。

grovel.exe：为系统服务，用于扫描零备份存储（SIS）卷上的重复文件，并且将重复文件指向一个数据存储点，以节省磁盘空间。

SCardSvr.exe：为系统服务，对插入在计算机智能卡阅读器中的智能卡进行管理和访问控制。

snmp.exe：为系统服务，用于包含代理程序可以监视网络设备的活动并向网络控制台工作站汇报。

snmptrap.exe：为系统服务，用于接收由本地或远程 SNMP 代理程序产生的陷阱消息，然后将消息传递到运行在这台计算机上的 SNMP 管理程序。

UtilMan.exe：为系统服务，用于从一个窗口中启动和配置辅助工具。

msiexec.exe：为系统服务，用于依据 MSI 文件中包含的命令来安装、修复以及删除软件。

1.2.2 各个进程的详细说明

（1）Svchost.exe。Svchost.exe 文件对那些从动态链接库中运行的服务来说是一个普通的主机进程名。Svchost.exe 文件定位在系统的%systemroot%\system32 文件夹下。在启动时，Svchost.exe 检查注册表中的位置来构建需要加载的服务列表，这就会使多个 Svchost.exe 在同一时间运行。每个 Svchost.exe 的会话期间都包含一组服务，以至于单独的服务必须依靠 Svchost.exe 怎样启动和在哪里启动，这样就更加容易控制和查找错误。

Svchost.exe 组是用下面的注册表值来识别的：

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Svchost

每个在这个键下的值代表一个独立的 Svchost 组，并且当用户查看活动的进程时，它显示作为一个单独的例子。每个键值都是 REG_MULTI_SZ 类型的值且包括运行在 Svchost 组内的服务。每个 Svchost 组都包含一个或多个从注册表值中选取的服务名，这个服务的参数值包含了一个 ServiceDLL 值。

为了能看到正运行在 Svchost 列表中的服务，可以依次执行“开始”→“运行”，输入 cmd 命令，然后再输入 tlist -s。tlist 显示一个活动进程的列表，开关-s 显示在每个进程中的活动服务列表。如果想知道更多的关于进程的信息，可以输入 tlist pid。

下面是 tlist 显示 Svchost.exe 运行的两个例子。

```
0 System Process
8 System
132 smss.exe
160 csrss.exe Title:
180 winlogon.exe Title: NetDDE Agent
208services.exe
Svcs:
AppMgmt,Browser,Dhcp,dmserver,Dnscache,Eventlog,lanmanserver,LanmanWorkstation,LmHosts,Messenger,
PlugPlay,ProtectedStorage,seclogon,TrkWks,W32Time,Wmi
220 lsass.exe Svcs: Netlogon,PolicyAgent,SamSs
404 svchost.exe Svcs: RpcSs
452 spoolsv.exe Svcs: Spooler
544 cisvc.exe Svcs: cisvc
556 svchost.exe Svcs: EventSystem,Netman,NtmsSvc,RasMan,SENS,TapiSrv
580 regsvc.exe Svcs: RemoteRegistry
596 mstask.exe Svcs: Schedule
660 snmp.exe Svcs: SNMP
728 winmgmt.exe Svcs: WinMgmt
852 cidaemon.exe Title: OleMainThreadWndName
```

812 explorer.exe Title: Program Manager

1032 OSA.EXE Title: Reminder

1300 cmd.exe Title: D:\WINNT5\System32\cmd.exe - tlist -s

1080 MAPISP32.EXE Title: WMS Idle

1264 rundll32.exe Title:

1000 mmc.exe Title: Device Manager

1144 tlist.exe

在这个例子中，注册表设置了两个组。

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Svchost:

netsvcs: Reg_Multi_SZ: EventSystem Ias Iprip Irmon Netman Nwsapagent Rasauto Rasman Remoteaccess SENS Sharedaccess

Tapisrv Ntmssvc

rpcss :Reg_Multi_SZ: RpcSs

smss.exe

csrss.exe

这个是用户模式 Win32 子系统的一部分。csrss 代表客户/服务器运行子系统且是一个基本的子系统，必须一直运行。csrss 负责控制 Windows，创建或删除线程和一些 16 位的虚拟 MS-DOS 环境。

（2）explorer.exe。这是一个用户的 shell，这个进程并不像用户想象的那样，作为一个重要的进程运行在 Windows 中，用户可以从任务管理器中关掉它或者重新启动。通常不会对系统产生什么负面影响。

（3）internat.exe。该进程是可以从任务管理器中关掉的。internat.exe 在启动时开始运行，它加载由用户指定的不同的输入点。输入点从注册表的：HKEY_USERS\DEFAULT\Keyboard Layout\Preload 加载内容。internat.exe 加载“EN”图标进入系统的图标区，允许使用者可以很容易地转换不同的输入点。当进程停掉时，图标就会消失，但是输入点仍然可以通过控制面板来改变。

（4）lsass.exe。该进程是不可以从任务管理器中关掉的。这是一个本地的安全授权服务，并且它会为使用 winlogon 服务的授权用户生成一个进程。这个进程是通过使用授权的包，如默认的 msgina.dll 来执行的。如果授权是成功的，lsass 就会产生用户的进入令牌，使用此令牌启动初始 shell，执行初始化工作。其他的由用户初始化的进程会继承这个令牌。

（5）mstask.exe。该进程是不可以从任务管理器中关掉的，是一个任务调度服务，负责用户事先决定在某一时间运行的任务的运行。

（6）smss.exe。一个会话管理子系统，负责启动用户会话，不能从任务管理器中关掉。这个进程由系统进程初始化，并与许多活动相关联，包括对正在运行的 Winlogon、Win32（Csrss.exe）线程和设定的系统变量作出反映。在它启动这些进程后，它等待

Winlogon 或 Csrss 结束。如果这些过程是正常的，系统就关掉了。如果发生了什么不可预料的事情，smss.exe 就会让系统停止响应。

（7）spoolsv.exe。这是个缓冲（spooler）服务，用来管理缓冲池中的打印和传真作业，这个进程不能从任务管理器中关掉。

（8）service.exe。大多数的系统核心模式进程是作为系统进程在运行，这个进程不能从任务管理器中关掉。

（9）winlogon.exe。这个进程是管理用户登录和退出的，而且 winlogon 在用户按下 Ctrl+Alt+Del 时激活，显示安全对话框。

（10）winmgmt.exe。winmgmt 是 Win2000 客户端管理的核心组件。当客户端应用程序连接或当管理程序需要他本身的服务时这个进程初始化。

（11）taskmagr.exe。这个进程是任务管理器。

1.2.3 性能选项卡

任务管理器中的性能选项卡是被进程中的线程所占用的 CPU 时间百分比。这个时间百分比是指被进程占用的 CPU 时间百分比，而不是 CPU 的编号，而此时的系统基本上是空闲的。尽管系统看上去每一秒左右都只使用一小部分 CPU 时间，但该系统空闲进程仍耗用了大约 99%的 CPU 时间。任务管理器中进程选项卡的第 4 列的 CPU 时间是 CPU 被进程中的线程累计占用的小时、分钟及秒数。注意，对进程中的线程使用占用一词，这并不一定意味着进程已耗用的 CPU 时间总和，因为一会儿将看到，NT 计时的方式是，当特定的时钟间隔激发时，无论谁恰巧处于当前的线程中，它都将计算到 CPU 周期之内。通常情况下，在大多数 NT 系统中，时钟以 10 毫秒的间隔运行。每 10 毫秒 NT 的“心脏”就跳动一下。有一些驱动程序代码片段运行并显示谁是当前的线程，将 CPU 时间的最后 10 毫秒记在它的账上。因此，如果一个线程开始运行，并在持续运行 8 毫秒后完成，接着，第二个线程开始运行并持续了 2 毫秒，这时，时钟激发，请猜一猜这整整 10 毫秒的时钟周期到底记在哪个线程的账上了？答案是第二个线程。因此，NT 中存在一些固有的不准确性，而 NT 恰是以这种方式进行计时，实际情况也是这样，大多数 32 位操作系统中都存在一个基于间隔的计时机制。请记住这一点，因为有时当用户观察线程所耗用的 CPU 时间总和时，会发现尽管该线程或许看上去已运行过数十万次，但其 CPU 时间占用量却可能是零或非常短暂的现象，那么，上述解释便是原因所在。上述也就是在任务管理器的进程选项卡中所能看到的基本信息列。

1.2.4 木马中的线程

如果把一条生产线看成一个进程，那么线上的众多人员分别负责完成相同或不同

的工作，这些人员就可以看成是一个个的线程。在网络中，木马程序是常见的网络攻击方式，这里用木马程序来说明线程的运行状况，以期读者对线程有一个直观的理解。木马程序的服务器端为了避免被发现，多数要进行隐藏处理。一个正常的 Windows 程序在运行后，会在系统中产生一个进程，同时，每个进程分别对应了一个不同的 PID（Progress ID，进程标识符），这个进程会被系统分配一个虚拟的内存空间地址段，一切相关的程序操作都会在这个虚拟的空间中进行。一个进程当以服务的方式工作时，它将会在后台工作，不会弹出在任务列表中，但是在 Windows NT/2000 下，用户仍然可以通过服务器管理器检查任何的服务程序是否被启动运行。一个进程可以存在一个或多个线程，线程之间同步执行多种操作，一般线程之间是相对独立的，当一个线程发生错误时，并不一定会导致整个系统的崩溃，想要木马隐藏于服务器端，可以伪隐藏，也可以真隐藏。伪隐藏是指程序的进程仍然存在，只不过是让它消失在进程列表中；真隐藏是让程序彻底消失，不以一个进程或服务的方式工作。伪隐藏的方法比较容易实现，只要把木马服务器端的程序注册为一个服务就可以了。这样，程序就会从任务列表中消失，因为系统不认为它是一个进程，当按下 Ctrl+Alt+Delete 键时，也看不到这个程序。但是它只适用于 Windows 9x 系统，对于 Windows NT/2000 等，通过服务器管理器就能发现系统中注册过的服务。但是利用 API 的拦截技术，通过建立一个后台的系统钩子拦截 psapi 的 enumprocessmodules 等相关的函数来实现对进程和服务的遍历调用的控制。当检测到进程 ID 为木马程序的服务器端进程时直接跳过就实现了进程的隐藏。金山词霸等就是使用类似的方法拦截了 textouta 和 textoutw 等函数来截获屏幕输出，实现即时翻译，同样的方法也用于进程隐藏上。当进程真隐藏时，那么这个木马的服务器端程序运行后，则不应该具备一般进程，也不应该具备服务，也就是说，完全溶进了系统的内核。可以不把它作成一个应用程序，而把它作成一个线程，一个其他应用程序的线程，把自身注入其他应用程序的地址空间。而这个应用程序对于系统来说是一个绝对安全的程序，这样就彻底达到了隐藏的效果。

1.3　线程和同步

程序的并发执行往往带来与时间有关的错误，甚至引发灾难性的后果，这需要引入同步机制。使用多进程与多线程时，有时需要协同两种或多种动作，此过程就称为同步（Synchronization）。引入同步机制的第一个原因是为了控制线程之间的资源同步访问，因为多个线程在共享资源时如果发生访问冲突通常会带来不正确的后果。例如，一个线程正在更新一个结构，同时另一个线程正试图读取同一个结构。结果，将无法得知所读取的数据是新的还是旧的，或者是二者的混合。第二个原因是有时要求确保线程之间的动作以指定的次序发生，如一个线程需要等待由另外一个线程所引起的事件。