<<Windows内核原理与实现>>

图书基本信息

书名：<<Windows内核原理与实现>>

13位ISBN编号：9787121105289

10位ISBN编号：7121105284

出版时间：2010年4月

出版时间：电子工业出版社

作者：潘爱民

页数：689

字数：850000

前言

《Windows内核原理与实现》是一本讲述Windows内核机理，并配合Windows的源代码来解释其实现细节的学习用书。
全书从操作系统原理的角度来组织内容。
本书适合于已经有了操作系统基本概念的读者进一步理解Windows操作系统。
通过学习本书的内容，读者不仅可以掌握Windows的核心机制，也可以理解像Windows这样的现代操作系统是如何构建起来的。
因此，这是一本学习Windows操作系统内核的书，而不是一本指导在Windows平台上进行软件开发的用书。

对于绝大多数IT从业人员以及高校学生来说，Windows是一个熟悉得不能再熟悉的操作系统了，而且也有相当多人能够熟练地开发各种类型的Windows应用程序，然而，真正熟悉Windows内部机理的人却少而又少。
究其根源，很多人将这归咎于Windows是一个闭源操作系统，也就是说，除了Microsoft内部的员工以外，人们接触不到Windows的源代码。
这样的解释只能说是部分正确的，因为从历史发展来看，UNIX类操作系统确实有广泛的群众基础，很多人得益于阅读各种UNIX版本的源代码（包括Linux和FreeBSD）。
但对于Windows操作系统，这似乎并非一个合理的解释，因为事实上，有相当多人在过去三四年间已经获得了Windows的源代码，但是在代码阐释和核心深入理解方面却并未表现出拥有源代码而带来的优势。
反倒是在过去几年间基于逆向工程而获得的Windows核心知识要深刻得多。

这么说并非指Windows源代码没有帮助，而是间接地揭示了一个事实：Windows的源代码不是那么容易读的，Windows的核心也不是那么好理解的。
本书作者在过去几年间，与国内高校的操作系统课程的老师和学生有广泛的接触和了解，真切地理解到，要想在操作系统课程中完全融入Windows的内容，迫切需要一本按照操作系统基本概念和理论来阐释Windows实现机理的书籍。
基于这样的动机，本书力图让学习操作系统的读者能够理解Windows中的核心机制，而且还可以近距离地观察到Windows内核中的实现细节。

内容概要

本书从操作系统原理的角度，详细解析了Windows如何实现现代操作系统的各个关键部件，包括进程、线程、物理内存和虚拟内存的管理，Windows中的同步和并发性支持，以及Windows的I/O模型。
在介绍这些关键部件时，本书直接以Windows的源代码（WRK, Windows Research Kernel）为参照，因而读者可以了解像Windows这样的复杂操作系统是如何在x86处理器上运行的。

在内容选取方面，本书侧重于Windows内核中最基本的系统部件，同时也兼顾到作为一个操作系统的完整性，所以，本书也介绍了像存储体系、网络、Windows环境子系统等，这些虽然并不位于内核模块但却支撑整个Windows运行的重要部件。
在本书最后，也介绍了Windows Server 2003以后的内核发展和变化。

虽然书中有大量关于Windows代码实现的描述，但是本书并没有罗列WRK中的代码，即使读者不对照WRK的源代码，也可以从这些章节的描述中理解Windows的实现机理。
在每一个技术专题的介绍中，本书几乎都提供了一个框架图，并且有关键细节的实现分析，这样做的意图是让读者既能够对一项技术有总体上的把握，也通晓关键的实现细节。

Windows操作系统已经有20年历史了，市面上有大量关于Windows技术的文档和书籍，但是，真正从源代码来诠释Windows底层机理的，本书还是第一次尝试。
在本书覆盖的内容中，有相当一部分是第一次以文字形式披露出来的，期望这些内容能消除人们对于Windows的神秘感。

写作本书的目的是让对Windows有好奇心的人真正了解到Windows中的核心机理，让计算机专业的学生和老师，以及系统软件工程师可以快速地领略到Windows中先进的系统技术，以及在Windows上编写出更加高效的软件。
本书也配备了一些小工具，通过这些小工具，读者可以查看内核中的静态或动态的信息，甚至观察系统的行为，可通过Internet下载这些工具。

作者简介

潘爱民，浙江海宁人，获得了南开大学数学学士学位，清华大学工学硕士学位，以及北京大学计算机科学博士学位。
他从中学时代开始接触计算机编程，经历了从DOS到Windows各种版本的发展历程。
潘爱民曾经长期从事软件技术的研究和开发工作，撰写了大量软件技术文章，1999年曾经是

<<Windows内核原理与实现>>

书籍目录

章节摘录

其次，在支持多任务的基础上，如果这些任务是相互独立的，则操作系统总是可以采用适当的方式让它们获得执行机会，但在实践中，应用程序为了实现各种功能逻辑，不同任务之间往往有一些逻辑上的关联性。
比如，任务之间必须强加某种时序关系，才能保证每个任务的状态是有效的；当多个任务竞争某些稀有资源（注意，系统的硬件资源，像打印机和显示器等，是共享的）时，系统必须确保这些任务有序执行。
所以，凡是存在共享资源的地方，操作系统都必须提供恰当的方法来同步应用程序对它的访问。
现代操作系统通常会提供多种同步机制，例如互斥体（mutex）、信号量（semaphore）、临界区（criticalsection）等。
应用程序可以有选择地使用这些同步机制，以确保多个任务有序地共享资源。
除了任务和同步的概念，每个应用程序还必须有它自己相对独立的执行空间。
在现代操作系统中，进程也代表了一个应用程序和它的执行空间。
不同进程的空间是相互隔离的，这是现代操作系统的基本需求。
操作系统必须在处理器的硬件特性基础之上，实现一套行之有效的空间隔离方案。
每个进程有它自己的内存空间，并且无法直接访问其他进程的内存空间。
进程之间如果要共享数据，则必须通过操作系统提供的机制来进行。
Intel x86体系结构上的操作系统基本上都利用硬件的虚拟内存映射机制来隔离每个进程的内存空间。
所以，操作系统的职责是为每个进程维护好从虚拟地址到物理地址的映射关系，并且管理好物理内存的分配和回收。
另一方面，除了进程的空间隔离性，操作系统还必须提供相应的机制让不同的进程可以相互通信，毕竟，很多软件需要进程之间的协作来完成一些上层功能。
同步机制和跨进程地共享内存是典型的进程间通信（IPC，Inter-Process Communication）手段。
前面提到，现代操作系统往往以统一的框架来管理I／0设备，这也隐含了操作系统向应用程序暴露的I／0接口是统一的这样一层意思。
应用程序通过此接口来访问系统的外部设备，而操作系统不仅要管理好应用程序访问外设的各种请求，包括它们的时序，还必须将应用程序的请求发送到对应的设备驱动程序中，最终由设备驱动程序来处理这些请求，而处理的结果也必须以某种方式回送给应用程序。
操作系统通常以句柄（handle）来代表一个可访问的抽象设备，抽grist备可能与物理设备连接，也可能并不存在对应的物理设备或资源。
操作系统还提供在一个句柄上读（read）、写（write）数据，以及发送控制命令的能力。
所以，应用程序与系统设备打交道的方式非常简洁明了：打开设备获得句柄、向设备发送命令、读或写设备，以及关闭设备。
操作系统的任务是管理这些设备和驱动程序，以及传递或解释应用程序的命令。

编辑推荐

微软亚洲研究院资深研究员潘爱民老师十年磨一剑之作　　用真实的源代码剖析Windows操作系统核心原理　　Windows内核专家Dave Probert作序力荐

版权说明