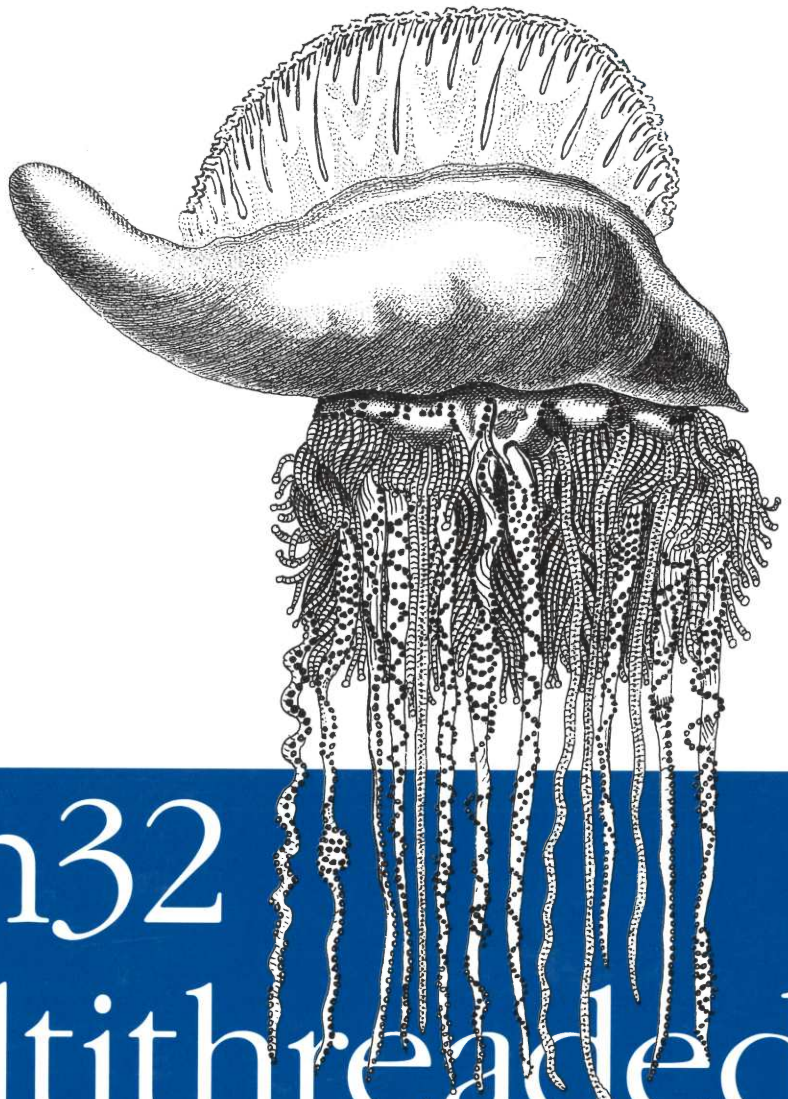
*Building Thread-Safe Applications*



# Win32 Multithreaded Programming

O'REILLY™

*Aaron Cohen & Mike Woodring*

# Win32 Multithreaded Programming

Aaron Cohen and Mike Woodring

O'REILLY™

*Cambridge · Köln · Paris · Sebastopol · Tokyo*

***Win32 Multithreaded Programming***
by Aaron Cohen and Mike Woodring



Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

***Editor:*** Ron Petrusha

***Production Editors:*** Jane Ellin and Nancy Crumpton

***Printing History:***

January 1998: First Edition



This book is printed on acid-free paper with 85% recycled content, 15% post-consumer waste. O'Reilly & Associates is committed to using paper with the highest recycled content available consistent with high quality.

ISBN: 1-56592-296-4

*In this chapter:*

- *What Is Multithreaded Programming?*
- *Why Write a Multithreaded Program (Why Use Threads)?*
- *When Not to Use Threads*
- *Making the Transition to Multithreaded Programming*

# 1

# *Introduction*

*If there be no great love in the beginning, yet heaven may decrease it upon better acquaintance, when we are married and have more occasion to know one another: I hope, upon familiarity will grow more contempt.*

—William Shakespeare
*The Merry Wives of Windsor*

While multithreading has long been available on mainframes and workstations, it is a new capability for personal computers. Prior to the first release of Windows NT, 16-bit versions of the Microsoft Windows operating system provided only a crude form of multitasking known as *cooperative multitasking*. With cooperative multitasking, all programs needed to be "good citizens" and share the CPU with other programs to enable the user to run more than one program at the same time. Unfortunately, most software was not always so well behaved, and the result was that running more than one program at a time was often more trouble than it was worth. All this changed for the better with the 32-bit Windows operating systems, Windows NT and Windows 95, which support *preemptive multitasking* and multithreading. Applications can now be written pretty much as if they are the only program running on the system, and the operating system ensures that all of the programs share the CPU and behave themselves.

With the new multithreading capabilities came new challenges for programmers. Most PC programmers were raised on DOS and other simple operating systems. Making the transition to Win32 programming and taking full advantage of the advanced features of the 32-bit operating systems can be difficult. Programmers who have not had experience with more advanced systems may not have been exposed to even the basic concepts of multitasking and multithreading.

In order to provide good grounding for the reader, this chapter will explain what multithreading is, and why you would want to use it. Because multithreading is not a solution to every problem, we will then discuss situations in which you would not want to use multithreading. The chapter ends with an introduction to the basic mindset you must have in order to write correct multithreaded programs.

## What Is Multithreaded Programming?

So what *is* multithreaded programming? Basically, *multithreaded programming* is implementing software so that two or more activities can be performed in parallel within the same application. This is accomplished by having each activity performed by its own thread. A *thread* is a path of execution through the software that has its own call stack and CPU state. Threads run within the context of a *process*, which defines an address space within which code and data exist, and threads execute. This is what most people think of when they refer to "multithreaded programming," but there really is a lot more to programming in a multithreaded environment.

*Good* multithreaded programming involves more than simply creating additional threads. We can loosely divide the issues into two categories. The first issue involves writing your software to use multiple threads in a useful and efficient manner. Carefully written multithreaded programs should be superior to single threaded designs in terms of execution time, user responsiveness, architecture, or all three.

The second issue is awareness of the operating system's rules that govern the behavior of your program while it's running, as well as understanding how your program interacts—directly or indirectly—with other programs running at the same time. You need to understand not just how the operating system will treat your program, but how it will treat your program when other programs are running at the same time. Likewise, understanding the impact your program has on other programs trying to run at the same time is just as important.

Becoming knowledgeable and adept at both aspects of multithreaded programming will be crucial to your success as a programmer on Windows 95 and Windows NT. This book will cover both aspects of writing good multithreaded programs.

## Why Write a Multithreaded Program (Why Use Threads)?

So why would you want to add extra threads to your programs? After all, you've been getting by just fine without support from Windows for incorporating

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.