R. H. Carver and K. C. Tai, *Modern Multithreading*, Wiley-Interscience, 2006.

Preface

This is a textbook on multithreaded programming. The objective of this book is to teach students about languages and libraries for multithreaded programming, to help students develop problem solving and programming skills, and to describe and demonstrate various testing and debugging techniques that have been developed for multithreaded programs over the past twenty years. It covers threads, semaphores, locks, monitors, message passing, and the relevant parts of Java, the POSIX Pthreads library, and the Windows Win32 Application Programming Interface (API).

This book is unique in that it provides in-depth coverage on testing and debugging multithreaded programs, a topic that typically receives little attention. The title "Modern Multithreading" reflects the fact that there are effective and relatively new testing and debugging techniques for multithreaded programs. The material in this text was developed in concurrent programming courses that the authors have taught for twenty years. This material includes results from the authors' research in concurrent programming, emphasizing tools and techniques that are of practical use. A class library has been implemented to provide working examples of all the material that is covered.

Classroom Use
In our experience, students have a hard time learning to write concurrent programs. If they manage to get their programs to run, they usually encounter deadlocks and other intermittent failures, and soon discover how hard it is to reproduce the failures and locate the cause of the problem. Essentially, they have no way to check the correctness of their programs, which interferes with learning. Instructors face the same problem when grading multithreaded programs. It is tedious, time-consuming, and often impossible to assess student programs by hand. The class libraries we have developed, and the testing techniques they support, can be used to assess student programs. When we assign programming problems in our courses, we also provide test cases that the students must use to assess the correctness of their programs. This is very helpful for the students and the instructors.

This text is designed for upper-level undergraduates and graduate students in computer science. It can be used as a main text in a concurrent programming course or it could be used as a supplementary text for an operating systems course or a software engineering course. Since the text emphasizes practical material, provides working code, and addresses testing and debugging problems that receive little or no attention in many other books, we believe that it will also be helpful to programmers in industry.

The text assumes students have the following background:

- Programming experience as typically gained in CS 1 and CS 2 courses
- Knowledge of elementary data structures as learned in a CS 2 course
- An understanding of Java fundamentals. Students should be familiar with object-oriented programming in Java, but no "advanced" knowledge is necessary.
- An understanding of C++ fundamentals. We use only the basic object-oriented programming features of C++.
- A prior course on operating systems is helpful but not required.

We have made an effort to minimize the differences between our Java and C++ programs. We use object-oriented features that are common to both languages and the class library has been implemented in both languages. While we don't illustrate every example in both Java and C++, the differences are very minor and it is easy to translate program examples from one language to the other.

Content
This book has seven chapters.

Chapter 1 defines operating systems terms such as process, thread, context switch, etc. It then shows how to create threads, first in Java and then in C++ using both the POSIX Pthreads library and the Win32 API. A C++ *Thread* class is provided to hide the details of thread creation in Pthreads/Win32. C++ programs that use the *Thread* class look remarkably similar to multithreaded Java programs. Fundamental concepts such as atomicity and non-

determinism are described using simple program examples. Chapter 1 ends by listing the issues and problems that arise when testing and debugging multithreaded programs. To illustrate the interesting things to come, we present a simple multithreaded C++ program that is capable of tracing and replaying its own executions.

Chapter 2 introduces concurrent programming by describing various solutions to the critical section problem. This problem is easy to understand, but hard to solve. The advantage of focusing on this problem is that it can be solved without introducing any complicated new programming constructs. Students gain a quick appreciation for the programming skills that they need to acquire. Chapter 2 also demonstrates how to trace and replay Peterson's solution to the critical section problem, which offers a straightforward introduction to several testing and debugging issues. The synchronization library implements the various techniques that are described.

Chapters 3, 4, and 5 cover semaphores, monitors and message passing, respectively. Each chapter describes one of these constructs and shows how to use it to solve programming problems. Semaphore and Lock classes for Java and C++/Win32/Pthreads are presented in Chapter 3. Chapter 4 presents monitor classes for Java and C++/Win32/Pthreads. Chapter 5 presents mailbox classes with send/receive methods and a selective wait statement. These chapters also cover the built-in support that Win32 and Pthreads provide for these constructs, as well as the support provided by J2SE 5.0 (Java 2 Platform, Standard Edition 5.0). Each chapter addresses a particular testing or debugging problem and shows how to solve it. The synchronization library implements the testing and debugging techniques so that students can apply them to their own programs.

Chapter 6 covers message passing in a distributed environment. It presents several Java mailbox classes that hide the details of TCP message passing and shows how to solve several distributed programming problems in Java. It also shows how to test and debug programs in a distributed environment (e.g., accurately tracing program executions by using vector timestamps). This chapter by no means provides complete coverage of distributed programming. Rather, it is meant to introduce students to the difficulty of distributed programming and to show them that the testing and debugging techniques presented in earlier chapters can be extended to work in a distributed environment. The synchronization library implements the various techniques.

Chapter 7 covers concepts that are fundamental to testing and debugging concurrent programs. It defines important terms, presents several test coverage criteria for concurrent programs, and describes the various approaches to testing concurrent programs. This chapter organizes and summarizes the testing and debugging material that is presented in-depth in Chapters 2 - 6. This organization provides two paths through the text. Instructors can cover the testing and debugging material in the last sections of Chapters 2 – 6 as they go through those chapters, or they can cover those sections when they cover Chapter 7. Chapter 7 also discusses reachability testing, which offers a bridge between testing and verification, and is implemented in the synchronization library.

Each chapter has exercises at the end. Some of the exercises explore the concepts covered in the chapter, while others require a program to be written. In our courses, we cover all the chapters, and give six homework assignments, two in-class exams and a project. We usually supplement the text with readings on model checking, process algebra, specification languages, and other research topics.

Online Resources
The home page for this book is located at

http://www.cs.gmu.edu/~rcarver/ModernMultithreading

This web site contains the source code for all the listings in the text and for the synchronization libraries. It also contains startup files and test cases for some of the exercises. Solutions to the exercises are available for instructors, as is a copy of our lecture notes. There will also be an errata page.

Acknowledgements
The suggestions we received from the anonymous reviewers were very helpful. The National Science Foundation supported our research through grants CCR-8907807, CCR-9320992, CCR-9309043, and CCR-9804112. We thank our research assistants and the students in our courses at North Carolina State and George Mason University for helping us solve many interesting problems. We also thank Prof. Jeff Lei at The University of Texas at Arlington for using early versions of this text in his courses.

R. H. Carver and K. C. Tai, *Modern Multithreading*, Wiley-Interscience, 2006.

My friend, colleague, and co-author Prof. K. C. Tai passed away before we could complete this book. K.C. was an outstanding teacher, a world-class researcher in the areas of software engineering, concurrent systems, programming languages, and compiler construction, and an impeccable and highly respected professional. If the reader finds this text helpful, it is a tribute to K.C.'s many contributions. Certainly, K.C. would have fixed the faults that I failed to find.

Richard H. Carver

Fairfax, Virginia
July 2005
rcarver@cs.gmu.edu