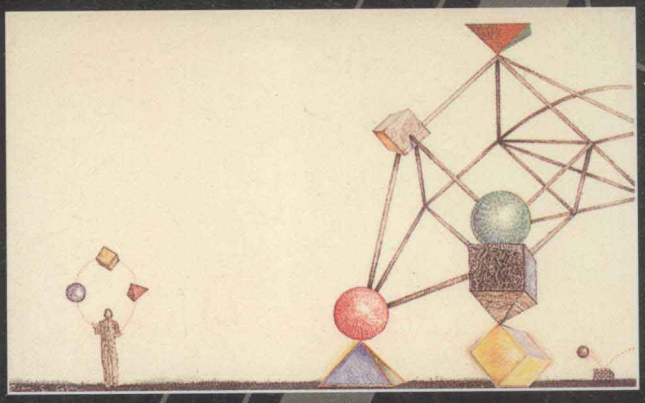
Brent B. Welch



Practical Programming in Tcl and Tk

DISKETTE INCLUDED

Practical Programming in Tcl and Tk

Brent B. Welch

Prentice Hall PTR
Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Welch, Brent.
Practical programming in Tcl and Tk / Brent Welch.
p. cm.
Includes index.
ISBN 0-13-182007-9
1. Tcl (Computer program language) 2. Tk toolkit. I. Title
QA76.73.T44W45 1995
005.13'3—dc20 95-1159
CIP

Editorial/production supervision: *Kerry Reardon*
Cover design: *Design Source*
Manufacturing buyer: *Alexis R. Heydt*
Acquisitions editor: *Mark Taub*
Cover photo: *Alan Cober / The Stock Illustration Source, Inc.*

Prentice-Hall, Inc.
A Simon & Schuster Company
Upper Saddle River, New Jersey 07458

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact:

Corporate Sales Department
Prentice Hall PTR
One Lake Street
Upper Saddle River, NJ 07458

Phone: 800-382-3419
Fax: 201-236-7141
E-mail: corpsales@prenhall.com

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-182007-9

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

to

Jody
and
Christopher

Preface

Tcl stands for *Tool Command Language*. Tcl is really two things: a scripting language, and an interpreter for that language that is designed to be easy to embed into your application. Tcl and its associated X windows toolkit, Tk, were designed and crafted by Professor John Ousterhout of the University of California, Berkeley. You can find these packages on the Internet, (as explained later), and use them freely in your application, even if it is commercial. The Tcl interpreter has been ported from UNIX to DOS, Windows, OS/2, NT, and Macintosh environments.

I first heard about Tcl in 1988 while I was Ousterhout's Ph.D. student at Berkeley. We were designing a network operating system, Sprite. While the students hacked on a new kernel, John wrote a new editor and terminal emulator. He used Tcl as the command language for both tools so users could define menus and otherwise customize those programs. This was in the days of X10, and he had plans for an X toolkit based on Tcl that would help programs cooperate with each other by communicating with Tcl commands. To me, this cooperation among tools was the essence of Tcl.

This early vision imagined that applications would be large bodies of compiled code and a small amount of Tcl used for configuration and high-level commands. John's editor, *mx*, and the terminal emulator, *tx*, followed this model. While this model remains valid, it has also turned out to be possible to write entire applications in Tcl. This is because of the Tcl/Tk shell, *wish*, that provides all the functionality of other shell languages, including running other programs, plus the ability to create a graphical user interface. For better or worse, it is now common to find applications that contain thousands of lines of Tcl script.

This book was written because, while I found it enjoyable and productive to use Tcl and Tk, there were times when I was frustrated. In addition, working at Xerox PARC, with many experts in languages and systems, I was compelled to understand both the strengths and weaknesses of Tcl and Tk. While many of my colleagues adopted Tcl and Tk for their projects, they were also just as quick to point out its flaws. In response, I have built up a set of programming techniques that exploit the power of Tcl and Tk while avoiding troublesome areas. Thus, this book is meant as a practical guide to help you get the most out of Tcl and Tk and avoid some of the frustrations I experienced.

Why Tcl?

As a scripting language, Tcl is similar to other UNIX shell languages such as the Bourne Shell (sh), the C Shell (csh), the Korn Shell (ksh), and Perl. Shell programs let you execute other programs. They provide enough programmability (variables, control flow, and procedures) to let you build complex scripts that assemble existing programs into a new tool tailored for your needs. Shells are wonderful for automating routine chores.

It is the ability to easily add a Tcl interpreter to your application that sets it apart from other shells. Tcl fills the role of an extension language that is used to configure and customize applications. There is no need to invent a command language for your new application, or struggle to provide some sort of user-programmability for your tool. Instead, by adding a Tcl interpreter, you structure your application as a set of primitive operations that can be composed by a script to best suit the needs of your users. It also allows other programs to have programmatic control over your application, leading to suites of applications that work well together.

There are other choices for extension languages that include Scheme, Elisp, and Python. Your choice between them is partly a matter of taste. Tcl has simple constructs and looks somewhat like C. It is easy to add new Tcl primitives by writing C procedures. In addition, the Tcl community has contributed many Tcl commands that you can access as-is. To me, the strength of the Tcl community is more important than the details of the language.

The Tcl C library has clean interfaces and is simple to use. The library implements the basic interpreter and a set of core scripting commands that implement variables, flow control, file I/O, and procedures (see page 17). In addition, your application can define new Tcl commands. These commands are associated with a C or C++ procedure that your application provides. The result is applications that are split into a set of primitives written in a compiled language, and exported as Tcl commands. A Tcl script is used to compose the primitives into the overall application. The script layer has access to shell-like capability to run other programs and access the file system, as well as call directly into the compiled part of the application through the application-specific Tcl commands you define. In addition, from the C programming level, you can call Tcl scripts, set and query Tcl variables, and even trace the execution of the Tcl interpreter.

There are many Tcl extensions freely available on the Internet. Most extensions include a C library that provides some new functionality, and a Tcl interface to the library. Examples include socket access for network programming, database access, telephone control, MIDI controller access, and *expect*, which adds Tcl commands to control interactive programs.

The most notable extension is Tk, a toolkit for X windows. Tk is currently being ported to Windows and Macintosh environments, too. Tk defines Tcl commands that let you create and manipulate user interface widgets. The script-based approach to user interface programming has three benefits:

- Development is fast because of the rapid turnaround; there is no waiting for long compilations.
- The Tcl commands provide a higher-level interface to X than most standard C library toolkits. Simple user interfaces require just a handful of commands to define them. At the same time, it is possible to refine the user interface in order to get every detail just so. The fast turnaround aids the refinement process.
- The user interface is clearly factored out from the rest of your application. The developer can concentrate on the implementation of the application core, and then fairly painlessly work up a user interface. The core set of Tk widgets is often sufficient for all your user interface needs. However, it is also possible to write custom Tk widgets in C, and again there are many contributed Tk widgets available on the network.

Tcl 7.4 and Tk 4.0

This book is up-to-date with Tcl version 7.4 and Tk version 4.0. There are occasional descriptions of Tk 3.6 features. The last chapter has some notes about porting scripts written in earlier versions of Tk.

Who Should Read This Book

This book is meant to be useful to the beginner in Tcl as well as the expert. For the beginner and expert alike I recommend careful study of Chapter One, *Tcl Fundamentals*. The programming model of Tcl is different from many programming languages. The model is based on string substitutions, and it is important that you understand it properly to avoid trouble in complex cases. The remainder of the book consists of examples that demonstrate how to use Tcl and Tk productively. For your reference, there are tables that summarize all the Tcl and Tk commands and widgets.

This book assumes that you have some UNIX and X background, although you should be able to get by even if you are a complete novice. Knowledge of UNIX shell programming will help, but it is not required. Where aspects of X are relevant, I provide some background information. Tcl has been widely ported, and the book will be helpful if you are using Tcl on a Macintosh, Windows machine, or some other environment.

How To Read This Book

This book is best used in a hands-on manner, at the computer, trying the examples. The book tries to fill the gap between the terse Tcl and Tk manual pages, which are complete but lack context and examples, and existing Tcl programs that may or may not be documented or well written.

I recommend the on-line manual pages for the Tcl and Tk commands. They provide a detailed reference guide to each command. This book summarizes much of the information from the manual pages, but it does not provide the complete details, which can vary from release to release.

I also recommend the book by John Ousterhout, *Tcl and the Tk Toolkit*, which provides a broad overview of all aspects of Tcl and Tk. There is some overlap with Ousterhout's book, although that book provides a more detailed treatment of C programming for Tcl, and this book provides more Tcl examples.

Ftp Archives

The primary site for the Tcl and Tk distributions is given below as a Universal Resource Location (URL). You can use FTP and login to the host (e.g., `ftp.cs.berkeley.edu`) under the `anonymous` user name. Give your email address as the password. The directory is in the URL after the host name (e.g., `ucb/tcl`).

```
ftp://ftp.cs.berkeley.edu/ucb/tcl
```

There are many sites that mirror this distribution. In addition, they provide an archive site for contributed Tcl commands, Tk widgets, and applications. There is also a set of Frequently Asked Questions files. As of this writing, these sites had Tcl sources:

```
ftp://ftp.aud.alcatel.com/tcl
ftp://syd.dit.csiro.au/pub/tk
ftp://ftp.ibp.fr/pub/tcl
ftp://src.doc.ic.ac.uk/packages/tcl/
ftp://ftp.luth.se/pub/unix/tcl/
ftp://ftp.switch.ch/mirror/tcl
ftp://ftp.sterling.com/programming/languages/tcl
ftp://ftp.sunet.se/pub/lang/tcl
mailto://ftpmail@ftp.sunet.se
ftp://ftp.cs.columbia.edu/archives/tcl
ftp://ftp.uni-paderborn.de/pub/unix/tcl
ftp://sunsite.unc.edu/pub/languages/tcl/
ftp://ftp.funet.fi/pub/languages/tcl/
```

You can also use a World Wide Web browser like *mosaic*, *netscape*, or *lynx* to access these sites. Enter the URL as specified above and you are presented with a directory listing of that location. From there you can change directories and fetch files.

If you do not have direct FTP access, you can use an email server for FTP. Send email to `ftpmail@decwrl.dec.com` with the message `Help` to get directions. If you are on BITNET, send email to `bitftp@pucc.princeton.edu`.

You can search for FTP sites that have Tcl by using the *archie* service that indexes the contents of anonymous FTP servers. Information about using *archie* can be obtained by sending mail to `archie@archie.sura.net` that contains the message `Help`.

World Wide Web

There are a number of pages on the World Wide Web about Tcl:

```
http://www.sco.com/IXI/of_interest/tcl/Tcl.html
http://web.cs.ualberta.ca/~wade/Auto/Tcl.html
```

Newsgroups

The `comp.lang.tcl` newsgroup is very active. It provides a forum for questions and answers about Tcl. Announcements about Tcl extensions and applications are posted frequently to this group.

Typographic Conventions

The more important examples are set apart with a title and horizontal rules, while others appear in-line. The examples use `courier` for Tcl and C code. When interesting results are returned by a Tcl command, those are presented below in *`oblique courier`*. The => is not part of the return value in the following example.

```
expr 5 + 8
=> 13
```

The `courier` font is also used when naming Tcl commands and C procedures within sentences.

The usage of a Tcl command is presented in the following example. The command name and constant keywords appear in `courier`. Variable values appear in *`courier oblique`*. Optional arguments are surrounded with question marks.

```
set varname ?value?
```

The name of a UNIX program is in italics:

xterm

Hot Tips

The icon in the margin marks a "hot tip" as judged by the reviewers of the book. The visual markers help you locate the more useful sections in the book. These are also listed in the index under Hot Tip.

On-line examples

The book comes with a floppy disk that has source for all of the examples. The floppy is in *tar* format, which you should be able to read on any UNIX system. On SunOS, use the following command to read the floppy:

```
% tar xvf /dev/rfd0c
```

The name of the floppy device may be different on your system.
You can also get the examples via FTP:

```
ftp://mercury.prenhall.com/pub/software/welch/tkbook.tar
```

Book Organization

The first chapter of this book describes the fundamental mechanisms that characterize the Tcl language. This is an important chapter that provides the basic grounding you will need to use Tcl effectively. Even if you have programmed in Tcl already, you should review this chapter.

Chapters 2-5 cover the basic Tcl commands in more detail, including string handling, regular expressions, data types, control flow, procedures and scoping issues. You can skip these chapters if you already know Tcl.

Chapter 6 discusses `eval` and more advanced Tcl coding techniques. If you are running into quoting problems, check out this chapter.

Chapter 7 describes the interface to the operating system for shell-like capabilities to run other programs and examine the file system. The I/O commands are described here.

Chapter 8 describes the facilities provided by the interpreter for introspection. You can find out about all the internal state of Tcl. The chapter describes development aids and debugging.

Chapter 9 describes the script library facility. If you do much Tcl programming, you should collect useful scripts into a library. This chapter also describes coding conventions to support larger scale programming efforts.

Chapter 10 is an introduction to Tk. It explains the relevant aspects of the X window system and the basic model provided by the Tk toolkit.

Chapter 11 illustrates Tk programming with a number of short examples. One of the examples is a browser for the code examples in this book.

Chapter 12 explains geometry management, which is responsible for arranging widgets on the screen. The chapter is primarily about the `pack` geometry manager. The simpler `place` geometry manager is described briefly.

Chapter 13 covers event binding. A binding registers a Tcl script that is executed in response to events from the X window system.

Chapter 14 describes the button and menu widgets. The chapter includes a simple menu package that hides some of details of setting up Tk menus.

Chapter 15 describes the X resource mechanism and how it relates to the Tk toolkit. The extended examples show how users can use resource specifications to define custom buttons and menus for an application.

Chapter 16 describes several simple Tk widgets: the frame, the toplevel, the label, the message, the scale, and the scrollbar. These widgets can be added to

your interface with two or three commands. The `bell` command is also described here.

Chapter 17 describes the entry and listbox widgets. These are specialized text widgets that provide a single line of text input and a scrollable list of text items, respectively. You are likely to program specialized behavior for these widgets.

Chapter 18 covers the issues related to dialog boxes. This includes input focus and grabs for modal interactions. A file selection dialog box is presented as an example.

Chapter 19 describes the text widget, which is a general purpose text widget with advanced features for text formatting, editing, and embedded images.

Chapter 20 describes the canvas widget that provides a general drawing interface.

Chapter 21 explains how to use the selection mechanism for cut-and-paste. Tk supports different selections, including the `CLIPBOARD` selection used by OpenLook tools.

Chapter 22 describes the `after`, `fileevent`, and `send` commands. These commands let you create sophisticated application structures, including cooperating suites of applications.

Chapter 23 is the first of three chapters that review the attributes that are shared among the Tk widget set. This chapter describes sizing and borders.

Chapter 24 describes colors, images and cursors. It explains how to use the bitmap and color photo image types. The chapter includes a complete map of the X cursor font.

Chapter 25 describes fonts and other text-related attributes. The extended example is a font selection application.

Chapter 26 explains how to interact with the window manager using the `wm` command. The chapter describes all the information available through the `winfo` command.

Chapter 27 presents a user interface to the binding mechanism. You can browse and edit bindings for widgets and classes with the interface.

Chapter 28 builds upon Chapter 15 to create a user preferences package and an associated user interface. The preference package links a Tcl variable used in your application to an X resource specification.

Chapter 29 provides a short introduction to using Tcl at the C programming level. It gets you started with integrating Tcl into an existing application, and it provides a survey the facilities in the Tcl C library.

Chapter 30 introduces C programming with the Tk toolkit. It surveys the Tk C library.

Chapter 31 presents a sample digital clock Tk widget implementation in C.

Chapter 32 is a survey of several interesting Tcl extension packages. The packages extend Tcl to provide access to more UNIX functionality (TclX), control over interactive programs (Expect), network programming (Tcl-DP), more Tk widgets (BLT), and an object system ([incr Tcl]). The chapter concludes with a C program that integrates all of these extensions into one *supertcl* application.

Chapter 33 has notes about porting your scripts to Tk 4.0.

Thanks

I would like to thank my managers and colleagues at Xerox PARC for their patience with me as I worked on this book. The tips and tricks in this book came partly from my own work as I helped lab members use Tcl, and partly from them as they taught me. Dave Nichols' probing questions forced me to understand the basic mechanisms of the Tcl interpreter. Dan Swinehart and Lawrence Butcher kept me sharp with their own critiques. Ron Frederick and Berry Kerchival adopted Tk for their graphical interfaces and amazed me with their rapid results. Becky Burwell, Rich Gold, Carl Hauser, John Maxwell, Ken Pier, Marvin Theimer and Mohan Vishwanath made use of my early drafts, and their questions pointed out large holes in the text. Karin Petersen, Bill Schilit, and Terri Watson kept life interesting by using Tcl in very non-standard ways. I especially thank my managers, Mark Weiser and Doug Terry, for their understanding and support.

I thank John Ousterhout for Tcl and Tk, which are wonderful systems built with excellent craftsmanship. John was kind enough to provide me with an advance version of Tk 4.0 so I could learn about its new features well before its first beta release.

Thanks to the Tcl programmers out on the net, from which I learned many tricks. John LoVerso and Steven Uhler are the hottest Tcl programmers I know.

Many thanks to the patient reviewers of early drafts: Pierre David, Clif Flynt, Simon Kenyon, Eugene Lee, Don Libes, Lee Moore, Joe Moss, Hador Shemtov, Frank Stajano, Charles Thayer, and Jim Thornton.

Many folks contributed suggestions by email: Miguel Angel, Stephen Bensen, Jeff Blaine, Tom Charnock, Brian Cooper, Patrick D'Cruze, Benoit Desrosiers, Ted Dunning, Mark Eichin, Paul Friberg, Carl Gauthier, David Gerdes, Klaus Hackenberg, Torkle Hasle, Marti Hearst, Jean-Pierre Herbert, Jamie Honan, Norman Klein, Joe Konstan, Susan Larson, Håkan Liljegren, Lionel Mallet, Dejan Milojicic, Greg Minshall, Bernd Mohr, Will Morse, Heiko Nardmann, Gerd Neugebauer, TV Raman, Cary Renzema, Rob Riepel, Dan Schenk, Jean-Guy Schneider, Elizabeth Scholl, Karl Schwamb, Rony Shapiro, Peter Simanyi, Vince Skahan, Bill Stumbo, Glen Vanderburg, Larry Virden, Reed Wade, and Jim Wight. Unfortunately I could not respond to every suggestion, even some that were excellent. I am still open to comments about this edition. My email address is `welch@parc.xerox.com`.

Thanks to the editors and staff at Prentice Hall. Mark Taub has been very helpful as I progressed through my first book. Lynn Schneider and Kerry Reardon were excellent copy and production editors, respectively.

Finally, I thank my wonderful wife Jody for her love, kindness, patience, wit, and understanding as I worked long hours. Happily, many of those hours were spent working from home. My new son Christopher gets the credit for keeping me from degenerating into a complete nerd.

Contents