

# BUILDING CUSTOM SOFTWARE TOOLS AND LIBRARIES

# Martin Stitt



In recognition of the important of preserving what has been written, it is a policy of John Wiley & Sons, Inc. to have books of enduring value published in the United States printed on acid-free paper, and we exert our best efforts to that end.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Copyright © 1993 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permission Department, John Wiley & Sons, Inc.

#### Library of Congress Cataloging-in-Publication Data

```
Stir , Martin
Building custom software tools and libraries / Martin Stitt.
p. cm.
Includes index.
ISBN 0-471-57914-9 (alk. paper : disk). — ISBN 0-471-57915-7 (pbk.). — ISBN 0-471-57916-5 (disk)
1. Computer software—Development. 2. Utilities (Computer programs). 1. Title.
OA76.76.D47S47 1993
005.1—dc20
92-33072
```

Printed in the United States of America 10 9 8 7 6 5 4 3 2 1

## TRADEMARK ACKNOWLEDGMENTS

TASM, TLINK, and TLIB are registered trademarks of Borland International, Inc. MASM, LINK, LIB, MS-DOS, and Windows are registered trademarks of Microsoft Corporation.

IBM, PC, XT, AT, and OS/2 are registered trademarks of International Business Machines Corporation.

BRIEF is a trademark of Solution Systems Company.

#### **ABOUT THE AUTHOR**

Martin Stitt began working as an electronics technician in 1972. His first brush with computers came when he designed and hand-built an 8080-based system. Martin progressed from there to work as a systems design engineer, developing microprocessor-based control systems for data communications, energy management, and data acquisition.

Martin now works as a contract software engineer and is involved in such projects as a multitasking/multiuser operating system, X.25, and NET-BIOS communications drivers, database applications, CASE systems, debugging tools, and, of course, library development.

His first book Debugging: Creative Techniques and Tools for Software Repair, was also published by John Wiley & Sons, Inc. He has written articles on software development published in PC Tech Journal, Dr. Dobb's Journal, Computer Language, C User's Journal, and Tech Specialist.

# PREFACE

This is a book for those of you who need to write software utilities. You might be an engineer in need of a custom software tool to model the characteristics of a new design. You might be a scientist who requires a custom tool to interface a piece of prototype laboratory equipment. You might be a print-shop technician in need of a custom utility to translate foreign typesetting codes to your system. You may be a programmer in need of a custom tool to create data sets for an application you are testing. The possibilities are endless!

I wrote this book to show how custom software tooling can be created in an efficient manner.

This is not a book on "How to Program in Assembler" or "How to Program in C" (or in any other specific language, for that matter). Books of that sort are plentiful. Nor is it a book that presents a specific library of tool-building components, requiring you to wade through pages and pages of source code listings.

This book addresses the nuts-and-bolts issues involved in the design and implementation of custom tools, including the following:

- Different forms tools can take
- User interfaces
- System interfaces
- Tool-to-tool interfaces
- How compilation and linking work
- The design, construction, and maintenance of function libraries
- Documentation strategies for tools and library modules
- File processing

Library techniques are covered because they make reusable software possible. If you aren't familiar with the design and maintenance of a function library, then, as you build more and more tools, you either have to reinvent certain wheels or you end up using other, less efficient means to share code.

One of my goals in writing this book was to balance the presentation. When a book on software development is too general, readers must do additional research to attain knowledge that they can apply in their work. Conversely, a book with a narrow focus, such as just covering a certain language or specific brand of compiler, becomes dated quickly.

This book generalizes by presenting its program code in pseudocode form first. The action chart diagramming method is used to impart a graphic clarity to this pseudocode. This approach provides significant detail about the function of the software in an easy-to-read and language-independent manner. (See Appendix A for an explanation of this pseudocode method.)

Many of the pseudocode presentations are followed by an actual implementation in C or assembler. For your convenience, all code examined in this book is available in both source and binary form on the Companion Diskette.

A combination of C and assembler is used in the templates and library functions on the Companion Diskette. The C portions were developed in standard C using Borland C++ version 2.0; the assembler portions were developed using Microsoft's MASM 5.1 and are compatible with Borland's TASM.

The program templates are designed to produce drivers and utilities composed of various combinations of C and assembler code. Device drivers and TSRs can be written entirely in assembler or in a mixture of assembler and C. Stand-alone utility programs are supported as only assembler, only C, or a combination. The program library functions are available to all configurations.

Because of the generalized presentation of the code in the book, implementing this book's techniques in languages other than C should be an easy task. A working knowledge of the MS-DOS operating system and its system-level interface is presumed.

#### THE COMPANION LIBRARY

The Companion Library contains the following types of functions:

Parsing Logic
Parse fixed and switch parameters
Support response files
Console Input Processing
Get keys
Perform background processing while waiting for key input

Check key input for hot keys Simulate key input

File Processing Logic

Read and process each line of a file Find first and find next file in a directory

Traverse directory tree on a disk

Stack-Switching Logic

Switch to local stack

Switch back to caller's stack

Pass call on to previous vector holder

String/Memory Logic

Copy blocks of memory

Fill memory

Copy/fill strings

Measure the length of strings

Check for character inclusion in a string

Display Logic

Display characters and strings

I/O Logic

Input a byte from a port

Output a byte to a port

Conversion Logic

Convert binary to decimal ASCII

Convert binary to hexadecimal ASCII

Convert text ASCII to hexadecimal

Convert character to upper case

Miscellaneous Logic

Derive a program's home path

Enable and disable interrupts

# CONTENTS OVERVIEW

	Preface	xxi
1	Basic Tool Design	1
2	Basic Library Design	17
3	Language Issues	37
4	Building Tools	45
5	Building Libraries	81
6	Documentation Management	105
7	User Interfaces	115
8	File and Directory Processing	139
Α	Pseudocode Conventions	167
В	C Source Code Listings	195
	On the Companion Diskette	265
	Index	267

# CONTENTS

	Preface	XXI
1	Basic Tool Design	1
	Types of Tools	1
	The Standalone Utility	1
	The TSR	3
	The Device Driver	5
	Modular Functions and Reusable Functions	6
	Designing for Modularity and Reusability	8
	Consequences	10
	Filters and Pipes	11
	Standalone Utilities versus Filters	14
	The Best of Both Worlds	15
2	Basic Library Design	17
	Source/Include Libraries	17
	Resident Libraries	19
	Dynamic Link Libraries	22
		xiii

## XIV CONTENTS

	Object Modules and Static-Link Libraries	22
	Object Module Records	25
	Linker Action for Object Modules and Libraries	27
	Introduction to Link Order Control	31
3	Language Issues	37
	Single-Language Programs	37
	Mixing ASM Functions with an HLL Base	40
	Mixing HLL Functions with an ASM Base	42
	Summary of Language Models	43
4	Building Tools	45
	The Template Files	45
	GETSTOCK.BAT and Other Batch Files	46
	Link Order Control for Tools	48
	Resident/Nonresident Programming	51
	Stack Issues	53
	Stack Addressing with the BP Register	53
	Stack Switching	56
	A First-Cut Attempt	58
	A Better Way	60
	Accessing the Caller's Registers	66
	Placing the Stack Switch Calls	68
	The Stack Pocket Technique	70
	TSR Specifics	70
	Device Driver Specifics	72
	The MODOBJ.EXE Utility	77
	Using Other High-Level Languages	77
	Stack Issues Pavisited	77

		Contents	XV
5	Building Libraries		81
•	Identifying Library Candidates		81
	An Example of Layering		82
	Shell Processes		83
	Designing Functions for Library Inclusion		83
	Bottom-Up Design		84
	Planning for Future Growth		84
	Error Handling		86
	Information Hiding, Encapsulation, and Abstraction		87
	Private and Public		88
	Private Global Variables		88
	Public Global Variables		90
	Granularity		91
	Private Helper Functions		91
	Function Hooks		92
	Video Driver Hooks		93
	Getkeys Hooks		95
	Function Hook Wrap-up		98
	Testing Library Functions		98
	Traditional Libraries and Link Order Control		99
	Header Files		99
	Generating a Library		100
	Library Maintenance Tips		101
6	Documentation Management		105
	EXTRACT.EXE—Basic Operation		106
	Program Summary Comment Blocks		107
	Library Function Comment Blocks		108

# XVI CONTENTS

	EXTRACT.EXE—Parameters	109
	Running EXTRACT.EXE from a Library Makefile	. 112
7	User Interfaces	115
	Command Line Parameter Parsing	115
	Fixed-Position Parameters	115
	Multiple File Specifications	116
	Switch Parameters	117
	Setting Up the Parsing Logic	119
	Calling the Parsing Logic	124
	Parsing Logic for Switches	125
	Parsing Logic for Fixed Parameters	126
	Parameter Defaults	127
	Console Input Processing	128
	The Polling Hook	131
	The Filter Hook	131
	Hook Function Interfacing	132
	The Library Functions	132
	Macro Peculiarities	133
	Mouse Movement to Cursor Key Emulation	133
	A Console Stack	135
	Executing Multiple Threads	135
	Coding for Reentrance	136
	Interactions Between Hook Functions	136
8	File and Directory Processing	139
	File Processing	140
	Directory Processing	147
	A Directory Search Library Module	148

		Contents	XVII
	The Need for List Building		151
	Processing a Series of Source Files		153
	Tree Processing		155
	The Tree Structure		155
	The lc_build_tree() Function		158
	The lc_trace_tree() Function		158
	The lc_free_tree() Function		161
	Interconnections		161
	Wildcard to Wildcard Transformations		162
	Filter-Style File Processing		165
Α	Pseudocode Conventions		167
	Action Charts		167
	Function Brackets		169
	Logic Statement Constructs		170
	Levels of Detail in Pseudocode		172
	Simple Data Declarations		173
	Structure Declarations		175
	Arrays and Strings		176
	Pointers and Hex Numbers		177
	Function Declarations and Return Values		180
	Operators and Readability		182
	Controlling Execution Flow		186
	Defined Constants and Reserved Variables		188
	Comment Headers		188
	Translation to Actual Code		190
В	C Source Listings		195
	<pre>lc_get_cmtail()</pre>		195
	lc_get_ddtail()		197

## XVIII CONTENTS

<pre>lc_rspfile()</pre>	198
<pre>lc_parse_sw()</pre>	200
<pre>lc_parse_fx()</pre>	206
<pre>lc_swp_assign()</pre>	210
<pre>lc_fxp_assign()</pre>	211
<pre>lc_isempty()</pre>	212
<pre>lc_trim_parm()</pre>	213
<pre>lc.inset()</pre>	214
<pre>lc_disp_char(), lc_disp_str(), lc_disp_err_lead()</pre>	215
<pre>lc_setup_showsw(), lc_report_showsw()</pre>	217
<pre>lc_toupper()</pre>	219
<pre>lc_getchar()</pre>	220
<pre>lc_getfname()</pre>	222
<pre>lc_verify_hex_fx()</pre>	224
<pre>lc_verify_hex_sw()</pre>	225
<pre>lc_verify_hexstr()</pre>	227
<pre>lc.find_files()</pre>	229
<pre>lc_trace_dir()</pre>	232
<pre>free_lptr_list()</pre>	234
<pre>lc_trace_dirl()</pre>	235
<pre>lc_tracdir_prep()</pre>	239
<pre>lc_build_tree(), lc_trace_tree(), lc_free_tree()</pre>	240
<pre>lc_eat_key()</pre>	248
<pre>lc_getkey_set()</pre>	249
<pre>lc_beep()</pre>	250
<pre>lc_set_phook(), lc_set_fhook(), lc_set_ahook(), lc_getkey (),</pre>	
<pre>lc_ifkey()</pre>	251
<pre>lc_process_src_parms()</pre>	255

	Contents	XiX
<pre>lc_home_path()</pre>		258
<pre>lc_form_template(), lc_translate_template()</pre>		259
<pre>lc_subst_meta()</pre>		262
On the Companion Diskette		265
Installing the Companion Toolset		265
Index		267

# Basic Tool Design

In this chapter we'll take a look at some of the basic aspects of tool design. Because building tools and libraries are often interdependent practices, you'll find that this chapter on tool design also discusses libraries and the chapters on library design cover the construction and design of tools as well.

This may seem like a lot of bouncing back and forth. At times, it may even seem like we're building a house both from the ground up and from the top down. In fact, that is precisely what is being done, such is the nature of interdependence.

### **TYPES OF TOOLS**

## The Standalone Utility

The standalone utility is the most familiar and common form of program. Most likely, you have used this type of program many times. Its code and data are contained within a file of the .COM or .EXE type. If you've assembled or compiled the obligatory "Hello world" sample program in your favorite language, then you've already made a standalone utility.

Standalone programs can be invoked manually from the command line or from within a batch file, or they can be executed as a child process of another utility program, such as a command shell, that allows you to start programs by pointing to their name in a list or by clicking on an icon. This type of tool can be as simple as the TREE utility included with MS-DOS, or it can be a more interactive tool—one that presents menus and prompts such as a spreadsheet or a CAD package. This book focuses on the development of