

Effective DEBUGGING

66 Specific Ways to Debug Software and Systems

Diomidis Spinellis

Effective Debugging

Effective Debugging: 66 Specific Ways to Debug Software and Systems

Table of Contents

Cover

Title Page

Copyright Page

Contents

Figures

Listings

Preface

Acknowledgments

About the Author

Chapter 1: High-Level Strategies

Item 1: Handle All Problems through an Issue-Tracking System

Item 2: Use Focused Queries to Search the Web for Insights into Your Problem

Item 3: Confirm That Preconditions and Postconditions Are Satisfied

Item 4: Drill Up from the Problem to the Bug or Down from the Programs Start to the Bug

Item 5: Find the Difference between a Known Good

Table of Contents

System and a Failing One

Item 6: Use the Softwares Debugging Facilities

Item 7: Diversify Your Build and Execution Environment

Item 8: Focus Your Work on the Most Important
Problems

Chapter 2: General-Purpose Methods and Practices

Item 9: Set Yourself Up for Debugging Success

Item 10: Enable the Efficient Reproduction of the Problem

Item 11: Minimize the Turnaround Time from Your
Changes to Their Result

Item 12: Automate Complex Testing Scenarios

Item 13: Enable a Comprehensive Overview of Your
Debugging Data

Item 14: Consider Updating Your Software

Item 15: Consult Third-Party Source Code for Insights on Its
Use

Item 16: Use Specialized Monitoring and Test
Equipment

Item 17: Increase the Prominence of a Failures Effects

Item 18: Enable the Debugging of Unwieldy Systems
from Your Desk

Item 19: Automate Debugging Tasks

Item 20: Houseclean Before and After Debugging

Item 21: Fix All Instances of a Problem Class

Chapter 3: General-Purpose Tools and Techniques

Table of Contents

Item 22: Analyze Debug Data with Unix Command-Line Tools

Item 23: Utilize Command-Line Tool Options and Idioms

Item 24: Explore Debug Data with Your Editor

Item 25: Optimize Your Work Environment

Item 26: Hunt the Causes and History of Bugs with the Revision Control System

Item 27: Use Monitoring Tools on Systems Composed of Independent Processes

Chapter 4: Debugger Techniques

Item 28: Use Code Compiled for Symbolic Debugging

Item 29: Step through the Code

Item 30: Use Code and Data Breakpoints

Item 31: Familiarize Yourself with Reverse Debugging

Item 32: Navigate along the Calls between Routines

Item 33: Look for Errors by Examining the Values of Variables and Expressions

Item 34: Know How to Attach a Debugger to a Running Process

Item 35: Know How to Work with Core Dumps

Item 36: Tune Your Debugging Tools

Item 37: Know How to View Assembly Code and Raw Memory

Chapter 5: Programming Techniques

Item 38: Review and Manually Execute Suspect Code

Item 39: Go Over Your Code and Reasoning with a

Table of Contents

Colleague

Item 40: Add Debugging Functionality

Item 41: Add Logging Statements

Item 42: Use Unit Tests

Item 43: Use Assertions

Item 44: Verify Your Reasoning by Perturbing the
Debugged Program

Item 45: Minimize the Differences between a Working
Example and the Failing Code

Item 46: Simplify the Suspect Code

Item 47: Consider Rewriting the Suspect Code in
Another Language

Item 48: Improve the Suspect Codes Readability and
Structure

Item 49: Fix the Bugs Cause, Rather Than Its
Symptom

Chapter 6: Compile-Time Techniques

Item 50: Examine Generated Code

Item 51: Use Static Program Analysis

Item 52: Configure Deterministic Builds and Executions

Item 53: Configure the Use of Debugging Libraries and
Checks

Chapter 7: Runtime Techniques

Item 54: Find the Fault by Constructing a Test Case

Item 55: Fail Fast

Item 56: Examine Application Log Files

Table of Contents

Item 57: Profile the Operation of Systems and Processes

Item 58: Trace the Codes Execution

Item 59: Use Dynamic Program Analysis Tools

Chapter 8: Debugging Multi-threaded Code

Item 60: Analyze Deadlocks with Postmortem Debugging

Item 61: Capture and Replicate

Item 62: Uncover Deadlocks and Race Conditions with
Specialized Tools

Item 63: Isolate and Remove Nondeterminism

Item 64: Investigate Scalability Issues by Looking at
Contention

Item 65: Locate False Sharing by Using Performance
Counters

Item 66: Consider Rewriting the Code Using
Higher-Level Abstractions

Web Resources

Index