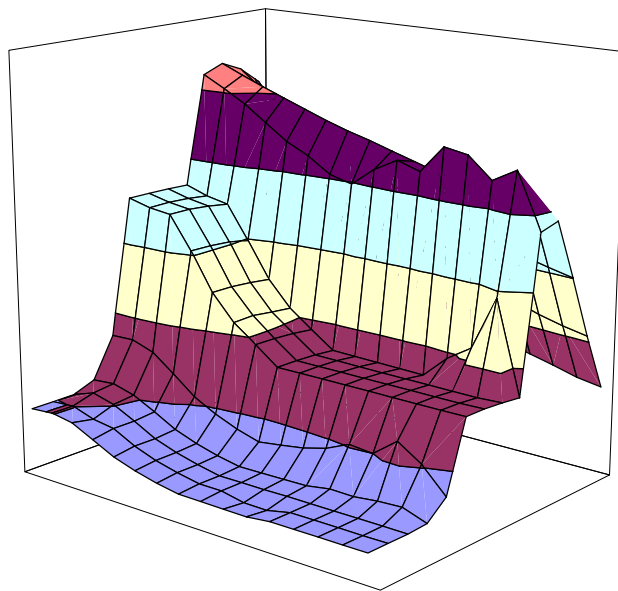


Computer Systems  
*A Programmer's Perspective*<sup>1</sup>  
(Beta Draft)



Randal E. Bryant  
David R. O'Hallaron

August 1, 2001

<sup>1</sup>Copyright © 2001, R. E. Bryant, D. R. O'Hallaron. All rights reserved.



# Contents

|  |           |
|--|-----------|
| <b>Preface</b>   | <b>i</b>  |
| <b>1 Introduction</b>  | <b>1</b>  |
| 1.1 Information is Bits in Context . . . . .                                 | 2         |
| 1.2 Programs are Translated by Other Programs into Different Forms . . . . . | 3         |
| 1.3 It Pays to Understand How Compilation Systems Work . . . . .             | 4         |
| 1.4 Processors Read and Interpret Instructions Stored in Memory . . . . .    | 5         |
| 1.4.1 Hardware Organization of a System . . . . .                            | 5         |
| 1.4.2 Running the <code>hello</code> Program . . . . .                       | 8         |
| 1.5 Caches Matter . . . . .  | 9         |
| 1.6 Storage Devices Form a Hierarchy . . . . .                               | 10        |
| 1.7 The Operating System Manages the Hardware . . . . .                      | 11        |
| 1.7.1 Processes . . . . .  | 13        |
| 1.7.2 Threads . . . . .  | 14        |
| 1.7.3 Virtual Memory . . . . .   | 14        |
| 1.7.4 Files . . . . .  | 15        |
| 1.8 Systems Communicate With Other Systems Using Networks . . . . .          | 16        |
| 1.9 Summary . . . . .  | 18        |
| <b>I Program Structure and Execution</b>                                     | <b>19</b> |
| <b>2 Representing and Manipulating Information</b>                           | <b>21</b> |
| 2.1 Information Storage . . . . .  | 22        |
| 2.1.1 Hexadecimal Notation . . . . .   | 23        |
| 2.1.2 Words . . . . .  | 25        |

|        |  |    |
|--------|--|----|
| 2.1.3  | Data Sizes . . . . .                                   | 25 |
| 2.1.4  | Addressing and Byte Ordering . . . . .                 | 26 |
| 2.1.5  | Representing Strings . . . . .                         | 33 |
| 2.1.6  | Representing Code . . . . .                            | 33 |
| 2.1.7  | Boolean Algebras and Rings . . . . .                   | 34 |
| 2.1.8  | Bit-Level Operations in C . . . . .                    | 37 |
| 2.1.9  | Logical Operations in C . . . . .                      | 39 |
| 2.1.10 | Shift Operations in C . . . . .                        | 40 |
| 2.2    | Integer Representations . . . . .                      | 41 |
| 2.2.1  | Integral Data Types . . . . .                          | 41 |
| 2.2.2  | Unsigned and Two's Complement Encodings . . . . .      | 41 |
| 2.2.3  | Conversions Between Signed and Unsigned . . . . .      | 45 |
| 2.2.4  | Signed vs. Unsigned in C . . . . .                     | 47 |
| 2.2.5  | Expanding the Bit Representation of a Number . . . . . | 49 |
| 2.2.6  | Truncating Numbers . . . . .                           | 51 |
| 2.2.7  | Advice on Signed vs. Unsigned . . . . .                | 52 |
| 2.3    | Integer Arithmetic . . . . .                           | 53 |
| 2.3.1  | Unsigned Addition . . . . .                            | 53 |
| 2.3.2  | Two's Complement Addition . . . . .                    | 56 |
| 2.3.3  | Two's Complement Negation . . . . .                    | 60 |
| 2.3.4  | Unsigned Multiplication . . . . .                      | 61 |
| 2.3.5  | Two's Complement Multiplication . . . . .              | 62 |
| 2.3.6  | Multiplying by Powers of Two . . . . .                 | 63 |
| 2.3.7  | Dividing by Powers of Two . . . . .                    | 64 |
| 2.4    | Floating Point . . . . .                               | 66 |
| 2.4.1  | Fractional Binary Numbers . . . . .                    | 67 |
| 2.4.2  | IEEE Floating-Point Representation . . . . .           | 69 |
| 2.4.3  | Example Numbers . . . . .                              | 71 |
| 2.4.4  | Rounding . . . . .                                     | 74 |
| 2.4.5  | Floating-Point Operations . . . . .                    | 76 |
| 2.4.6  | Floating Point in C . . . . .                          | 77 |
| 2.5    | Summary . . . . .                                      | 79 |

|          |   |           |
|----------|---|-----------|
| <b>3</b> | <b>Machine-Level Representation of C Programs</b> | <b>91</b> |
| 3.1      | A Historical Perspective . . . . .                | 92        |
| 3.2      | Program Encodings . . . . .                       | 94        |
| 3.2.1    | Machine-Level Code . . . . .                      | 95        |
| 3.2.2    | Code Examples . . . . .                           | 96        |
| 3.2.3    | A Note on Formatting . . . . .                    | 99        |
| 3.3      | Data Formats . . . . .                            | 100       |
| 3.4      | Accessing Information . . . . .                   | 101       |
| 3.4.1    | Operand Specifiers . . . . .                      | 102       |
| 3.4.2    | Data Movement Instructions . . . . .              | 104       |
| 3.4.3    | Data Movement Example . . . . .                   | 105       |
| 3.5      | Arithmetic and Logical Operations . . . . .       | 107       |
| 3.5.1    | Load Effective Address . . . . .                  | 108       |
| 3.5.2    | Unary and Binary Operations . . . . .             | 108       |
| 3.5.3    | Shift Operations . . . . .                        | 109       |
| 3.5.4    | Discussion . . . . .                              | 110       |
| 3.5.5    | Special Arithmetic Operations . . . . .           | 111       |
| 3.6      | Control . . . . .                                 | 112       |
| 3.6.1    | Condition Codes . . . . .                         | 112       |
| 3.6.2    | Accessing the Condition Codes . . . . .           | 113       |
| 3.6.3    | Jump Instructions and their Encodings . . . . .   | 116       |
| 3.6.4    | Translating Conditional Branches . . . . .        | 119       |
| 3.6.5    | Loops . . . . .                                   | 121       |
| 3.6.6    | Switch Statements . . . . .                       | 130       |
| 3.7      | Procedures . . . . .                              | 134       |
| 3.7.1    | Stack Frame Structure . . . . .                   | 134       |
| 3.7.2    | Transferring Control . . . . .                    | 136       |
| 3.7.3    | Register Usage Conventions . . . . .              | 137       |
| 3.7.4    | Procedure Example . . . . .                       | 139       |
| 3.7.5    | Recursive Procedures . . . . .                    | 142       |
| 3.8      | Array Allocation and Access . . . . .             | 144       |
| 3.8.1    | Basic Principles . . . . .                        | 145       |
| 3.8.2    | Pointer Arithmetic . . . . .                      | 146       |

|          |  |            |
|----------|--|------------|
| 3.8.3    | Arrays and Loops . . . . .                                       | 147        |
| 3.8.4    | Nested Arrays . . . . .  | 147        |
| 3.8.5    | Fixed Size Arrays . . . . .                                      | 150        |
| 3.8.6    | Dynamically Allocated Arrays . . . . .                           | 152        |
| 3.9      | Heterogeneous Data Structures . . . . .                          | 155        |
| 3.9.1    | Structures . . . . .   | 155        |
| 3.9.2    | Unions . . . . .   | 158        |
| 3.10     | Alignment . . . . .  | 162        |
| 3.11     | Putting it Together: Understanding Pointers . . . . .            | 164        |
| 3.12     | Life in the Real World: Using the GDB Debugger . . . . .         | 167        |
| 3.13     | Out-of-Bounds Memory References and Buffer Overflow . . . . .    | 169        |
| 3.14     | *Floating-Point Code . . . . .                                   | 174        |
| 3.14.1   | Floating-Point Registers . . . . .                               | 174        |
| 3.14.2   | Extended-Precision Arithmetic . . . . .                          | 175        |
| 3.14.3   | Stack Evaluation of Expressions . . . . .                        | 178        |
| 3.14.4   | Floating-Point Data Movement and Conversion Operations . . . . . | 181        |
| 3.14.5   | Floating-Point Arithmetic Instructions . . . . .                 | 183        |
| 3.14.6   | Using Floating Point in Procedures . . . . .                     | 185        |
| 3.14.7   | Testing and Comparing Floating-Point Values . . . . .            | 186        |
| 3.15     | *Embedding Assembly Code in C Programs . . . . .                 | 188        |
| 3.15.1   | Basic Inline Assembly . . . . .                                  | 189        |
| 3.15.2   | Extended Form of <code>asm</code> . . . . .                      | 191        |
| 3.16     | Summary . . . . .  | 194        |
| <b>4</b> | <b>Processor Architecture</b>                                    | <b>203</b> |
| <b>5</b> | <b>Optimizing Program Performance</b>                            | <b>205</b> |
| 5.1      | Capabilities and Limitations of Optimizing Compilers . . . . .   | 206        |
| 5.2      | Expressing Program Performance . . . . .                         | 209        |
| 5.3      | Program Example . . . . .  | 211        |
| 5.4      | Eliminating Loop Inefficiencies . . . . .                        | 214        |
| 5.5      | Reducing Procedure Calls . . . . .                               | 218        |
| 5.6      | Eliminating Unneeded Memory References . . . . .                 | 220        |

|          |   |            |
|----------|---|------------|
| 5.7      | Understanding Modern Processors . . . . .                                       | 222        |
| 5.7.1    | Overall Operation . . . . .   | 223        |
| 5.7.2    | Functional Unit Performance . . . . .   | 226        |
| 5.7.3    | A Closer Look at Processor Operation . . . . .                                  | 227        |
| 5.8      | Reducing Loop Overhead . . . . .  | 235        |
| 5.9      | Converting to Pointer Code . . . . .  | 240        |
| 5.10     | Enhancing Parallelism . . . . .   | 243        |
| 5.10.1   | Loop Splitting . . . . .  | 243        |
| 5.10.2   | Register Spilling . . . . .   | 247        |
| 5.10.3   | Limits to Parallelism . . . . .   | 249        |
| 5.11     | Putting it Together: Summary of Results for Optimizing Combining Code . . . . . | 249        |
| 5.11.1   | Floating-Point Performance Anomaly . . . . .                                    | 250        |
| 5.11.2   | Changing Platforms . . . . .  | 251        |
| 5.12     | Branch Prediction and Misprediction Penalties . . . . .                         | 251        |
| 5.13     | Understanding Memory Performance . . . . .                                      | 254        |
| 5.13.1   | Load Latency . . . . .  | 255        |
| 5.13.2   | Store Latency . . . . .   | 257        |
| 5.14     | Life in the Real World: Performance Improvement Techniques . . . . .            | 262        |
| 5.15     | Identifying and Eliminating Performance Bottlenecks . . . . .                   | 263        |
| 5.15.1   | Program Profiling . . . . .   | 263        |
| 5.15.2   | Using a Profiler to Guide Optimization . . . . .                                | 265        |
| 5.15.3   | Amdahl's Law . . . . .  | 268        |
| 5.16     | Summary . . . . .   | 269        |
| <b>6</b> | <b>The Memory Hierarchy</b>   | <b>277</b> |
| 6.1      | Storage Technologies . . . . .  | 278        |
| 6.1.1    | Random-Access Memory . . . . .  | 278        |
| 6.1.2    | Disk Storage . . . . .  | 287        |
| 6.1.3    | Storage Technology Trends . . . . .   | 295        |
| 6.2      | Locality . . . . .  | 297        |
| 6.2.1    | Locality of References to Program Data . . . . .                                | 297        |
| 6.2.2    | Locality of Instruction Fetches . . . . .                                       | 299        |
| 6.2.3    | Summary of Locality . . . . .   | 299        |

|  |  |            |
|--|--|------------|
| 6.3                                    | The Memory Hierarchy . . . . .   | 300        |
| 6.3.1                                  | Caching in the Memory Hierarchy . . . . .                                  | 303        |
| 6.3.2                                  | Summary of Memory Hierarchy Concepts . . . . .                             | 305        |
| 6.4                                    | Cache Memories . . . . .   | 306        |
| 6.4.1                                  | Generic Cache Memory Organization . . . . .                                | 307        |
| 6.4.2                                  | Direct-Mapped Caches . . . . .   | 308        |
| 6.4.3                                  | Set Associative Caches . . . . .   | 315        |
| 6.4.4                                  | Fully Associative Caches . . . . .   | 317        |
| 6.4.5                                  | Issues with Writes . . . . .   | 320        |
| 6.4.6                                  | Instruction Caches and Unified Caches . . . . .                            | 321        |
| 6.4.7                                  | Performance Impact of Cache Parameters . . . . .                           | 322        |
| 6.5                                    | Writing Cache-friendly Code . . . . .                                      | 324        |
| 6.6                                    | Putting it Together: The Impact of Caches on Program Performance . . . . . | 329        |
| 6.6.1                                  | The Memory Mountain . . . . .  | 329        |
| 6.6.2                                  | Rearranging Loops to Increase Spatial Locality . . . . .                   | 333        |
| 6.6.3                                  | Using Blocking to Increase Temporal Locality . . . . .                     | 337        |
| 6.7                                    | Summary . . . . .  | 340        |
| <b>II Running Programs on a System</b> |  | <b>349</b> |
| <b>7</b>                               | <b>Linking</b>   | <b>351</b> |
| 7.1                                    | Compiler Drivers . . . . .   | 352        |
| 7.2                                    | Static Linking . . . . .   | 353        |
| 7.3                                    | Object Files . . . . .   | 354        |
| 7.4                                    | Relocatable Object Files . . . . .   | 355        |
| 7.5                                    | Symbols and Symbol Tables . . . . .  | 356        |
| 7.6                                    | Symbol Resolution . . . . .  | 359        |
| 7.6.1                                  | How Linkers Resolve Multiply-Defined Global Symbols . . . . .              | 360        |
| 7.6.2                                  | Linking with Static Libraries . . . . .                                    | 363        |
| 7.6.3                                  | How Linkers Use Static Libraries to Resolve References . . . . .           | 366        |
| 7.7                                    | Relocation . . . . .   | 367        |
| 7.7.1                                  | Relocation Entries . . . . .   | 368        |
| 7.7.2                                  | Relocating Symbol References . . . . .                                     | 369        |



|          |   |            |
|----------|---|------------|
| 7.8      | Executable Object Files . . . . .   | 373        |
| 7.9      | Loading Executable Object Files . . . . .                                 | 374        |
| 7.10     | Dynamic Linking with Shared Libraries . . . . .                           | 376        |
| 7.11     | Loading and Linking Shared Libraries from Applications . . . . .          | 378        |
| 7.12     | *Position-Independent Code (PIC) . . . . .                                | 379        |
| 7.13     | Tools for Manipulating Object Files . . . . .                             | 383        |
| 7.14     | Summary . . . . .   | 384        |
| <b>8</b> | <b>Exceptional Control Flow</b>   | <b>393</b> |
| 8.1      | Exceptions . . . . .  | 394        |
| 8.1.1    | Exception Handling . . . . .  | 395        |
| 8.1.2    | Classes of Exceptions . . . . .   | 396        |
| 8.1.3    | Exceptions in Intel Processors . . . . .                                  | 399        |
| 8.2      | Processes . . . . .   | 400        |
| 8.2.1    | Logical Control Flow . . . . .  | 400        |
| 8.2.2    | Private Address Space . . . . .   | 401        |
| 8.2.3    | User and Kernel Modes . . . . .   | 402        |
| 8.2.4    | Context Switches . . . . .  | 403        |
| 8.3      | System Calls and Error Handling . . . . .                                 | 404        |
| 8.4      | Process Control . . . . .   | 405        |
| 8.4.1    | Obtaining Process ID's . . . . .  | 406        |
| 8.4.2    | Creating and Terminating Processes . . . . .                              | 406        |
| 8.4.3    | Reaping Child Processes . . . . .   | 411        |
| 8.4.4    | Putting Processes to Sleep . . . . .                                      | 416        |
| 8.4.5    | Loading and Running Programs . . . . .                                    | 417        |
| 8.4.6    | Using <code>fork</code> and <code>execve</code> to Run Programs . . . . . | 420        |
| 8.5      | Signals . . . . .   | 421        |
| 8.5.1    | Signal Terminology . . . . .  | 421        |
| 8.5.2    | Sending Signals . . . . .   | 425        |
| 8.5.3    | Receiving Signals . . . . .   | 428        |
| 8.5.4    | Signal Handling Issues . . . . .  | 431        |
| 8.5.5    | Portable Signal Handling . . . . .  | 436        |
| 8.6      | Nonlocal Jumps . . . . .  | 438        |

|           |  |            |
|-----------|--|------------|
| 8.7       | Tools for Manipulating Processes . . . . .   | 443        |
| 8.8       | Summary . . . . .  | 443        |
| <b>9</b>  | <b>Measuring Program Execution Time</b>  | <b>451</b> |
| 9.1       | The Flow of Time on a Computer System . . . . .  | 452        |
| 9.1.1     | Process Scheduling and Timer Interrupts . . . . .  | 453        |
| 9.1.2     | Time from an Application Program's Perspective . . . . .                                     | 454        |
| 9.2       | Measuring Time by Interval Counting . . . . .  | 456        |
| 9.2.1     | Operation . . . . .  | 458        |
| 9.2.2     | Reading the Process Timers . . . . .   | 458        |
| 9.2.3     | Accuracy of Process Timers . . . . .   | 459        |
| 9.3       | Cycle Counters . . . . .   | 461        |
| 9.3.1     | IA32 Cycle Counters . . . . .  | 462        |
| 9.4       | Measuring Program Execution Time with Cycle Counters . . . . .                               | 462        |
| 9.4.1     | The Effects of Context Switching . . . . .   | 464        |
| 9.4.2     | Caching and Other Effects . . . . .  | 465        |
| 9.4.3     | The <i>K</i> -Best Measurement Scheme . . . . .  | 469        |
| 9.5       | Time-of-Day Measurements . . . . .   | 478        |
| 9.6       | Putting it Together: An Experimental Protocol . . . . .                                      | 480        |
| 9.7       | Looking into the Future . . . . .  | 482        |
| 9.8       | Life in the Real World: An Implementation of the <i>K</i> -Best Measurement Scheme . . . . . | 482        |
| 9.9       | Summary . . . . .  | 483        |
| <b>10</b> | <b>Virtual Memory</b>  | <b>487</b> |
| 10.1      | Physical and Virtual Addressing . . . . .  | 488        |
| 10.2      | Address Spaces . . . . .   | 489        |
| 10.3      | VM as a Tool for Caching . . . . .   | 490        |
| 10.3.1    | DRAM Cache Organization . . . . .  | 491        |
| 10.3.2    | Page Tables . . . . .  | 491        |
| 10.3.3    | Page Hits . . . . .  | 492        |
| 10.3.4    | Page Faults . . . . .  | 493        |
| 10.3.5    | Allocating Pages . . . . .   | 494        |
| 10.3.6    | Locality to the Rescue Again . . . . .   | 495        |

|         |   |     |
|---------|---|-----|
| 10.4    | VM as a Tool for Memory Management . . . . .                            | 495 |
| 10.4.1  | Simplifying Linking . . . . .   | 496 |
| 10.4.2  | Simplifying Sharing . . . . .   | 496 |
| 10.4.3  | Simplifying Memory Allocation . . . . .                                 | 497 |
| 10.4.4  | Simplifying Loading . . . . .   | 497 |
| 10.5    | VM as a Tool for Memory Protection . . . . .                            | 498 |
| 10.6    | Address Translation . . . . .   | 499 |
| 10.6.1  | Integrating Caches and VM . . . . .                                     | 502 |
| 10.6.2  | Speeding up Address Translation with a TLB . . . . .                    | 502 |
| 10.6.3  | Multi-level Page Tables . . . . .                                       | 503 |
| 10.6.4  | Putting it Together: End-to-end Address Translation . . . . .           | 506 |
| 10.7    | Case Study: The Pentium/Linux Memory System . . . . .                   | 510 |
| 10.7.1  | Pentium Address Translation . . . . .                                   | 510 |
| 10.7.2  | Linux Virtual Memory System . . . . .                                   | 515 |
| 10.8    | Memory Mapping . . . . .  | 518 |
| 10.8.1  | Shared Objects Revisited . . . . .                                      | 519 |
| 10.8.2  | The <code>fork</code> Function Revisited . . . . .                      | 521 |
| 10.8.3  | The <code>execve</code> Function Revisited . . . . .                    | 521 |
| 10.8.4  | User-level Memory Mapping with the <code>mmap</code> Function . . . . . | 522 |
| 10.9    | Dynamic Memory Allocation . . . . .                                     | 524 |
| 10.9.1  | The <code>malloc</code> and <code>free</code> Functions . . . . .       | 525 |
| 10.9.2  | Why Dynamic Memory Allocation? . . . . .                                | 526 |
| 10.9.3  | Allocator Requirements and Goals . . . . .                              | 528 |
| 10.9.4  | Fragmentation . . . . .   | 530 |
| 10.9.5  | Implementation Issues . . . . .   | 531 |
| 10.9.6  | Implicit Free Lists . . . . .   | 531 |
| 10.9.7  | Placing Allocated Blocks . . . . .                                      | 533 |
| 10.9.8  | Splitting Free Blocks . . . . .   | 533 |
| 10.9.9  | Getting Additional Heap Memory . . . . .                                | 534 |
| 10.9.10 | Coalescing Free Blocks . . . . .  | 534 |
| 10.9.11 | Coalescing with Boundary Tags . . . . .                                 | 535 |
| 10.9.12 | Putting it Together: Implementing a Simple Allocator . . . . .          | 537 |
| 10.9.13 | Explicit Free Lists . . . . .   | 545 |

|  |            |
|--|------------|
| 10.9.14 Segregated Free Lists . . . . .  | 546        |
| 10.10 Garbage Collection . . . . .   | 548        |
| 10.10.1 Garbage Collector Basics . . . . .   | 549        |
| 10.10.2 Mark&Sweep Garbage Collectors . . . . .  | 550        |
| 10.10.3 Conservative Mark&Sweep for C Programs . . . . .                                 | 552        |
| 10.11 Common Memory-related Bugs in C Programs . . . . .                                 | 553        |
| 10.11.1 Dereferencing Bad Pointers . . . . .   | 553        |
| 10.11.2 Reading Uninitialized Memory . . . . .   | 553        |
| 10.11.3 Allowing Stack Buffer Overflows . . . . .  | 554        |
| 10.11.4 Assuming that Pointers and the Objects they Point to Are the Same Size . . . . . | 554        |
| 10.11.5 Making Off-by-one Errors . . . . .   | 555        |
| 10.11.6 Referencing a Pointer Instead of the Object it Points to . . . . .               | 555        |
| 10.11.7 Misunderstanding Pointer Arithmetic . . . . .                                    | 556        |
| 10.11.8 Referencing Non-existent Variables . . . . .                                     | 556        |
| 10.11.9 Referencing Data in Free Heap Blocks . . . . .                                   | 557        |
| 10.11.10 Introducing Memory Leaks . . . . .  | 557        |
| 10.12 Summary . . . . .  | 558        |
| <br>   |            |
| <b>III Interaction and Communication Between Programs</b>                                | <b>563</b> |
| <br>   |            |
| <b>11 Concurrent Programming with Threads</b>  | <b>565</b> |
| 11.1 Basic Thread Concepts . . . . .   | 565        |
| 11.2 Thread Control . . . . .  | 568        |
| 11.2.1 Creating Threads . . . . .  | 569        |
| 11.2.2 Terminating Threads . . . . .   | 569        |
| 11.2.3 Reaping Terminated Threads . . . . .  | 570        |
| 11.2.4 Detaching Threads . . . . .   | 570        |
| 11.3 Shared Variables in Threaded Programs . . . . .                                     | 572        |
| 11.3.1 Threads Memory Model . . . . .  | 572        |
| 11.3.2 Mapping Variables to Memory . . . . .   | 572        |
| 11.3.3 Shared Variables . . . . .  | 574        |
| 11.4 Synchronizing Threads with Semaphores . . . . .                                     | 575        |
| 11.4.1 Sequential Consistency . . . . .  | 575        |

|           |  |            |
|-----------|--|------------|
| 11.4.2    | Progress Graphs . . . . .  | 578        |
| 11.4.3    | Protecting Shared Variables with Semaphores . . . . .                                  | 581        |
| 11.4.4    | Posix Semaphores . . . . .   | 582        |
| 11.4.5    | Signaling With Semaphores . . . . .  | 583        |
| 11.5      | Synchronizing Threads with Mutex and Condition Variables . . . . .                     | 585        |
| 11.5.1    | Mutex Variables . . . . .  | 585        |
| 11.5.2    | Condition Variables . . . . .  | 588        |
| 11.5.3    | Barrier Synchronization . . . . .  | 589        |
| 11.5.4    | Timeout Waiting . . . . .  | 590        |
| 11.6      | Thread-safe and Reentrant Functions . . . . .  | 594        |
| 11.6.1    | Reentrant Functions . . . . .  | 595        |
| 11.6.2    | Thread-safe Library Functions . . . . .  | 598        |
| 11.7      | Other Synchronization Errors . . . . .   | 598        |
| 11.7.1    | Races . . . . .  | 598        |
| 11.7.2    | Deadlocks . . . . .  | 601        |
| 11.8      | Summary . . . . .  | 602        |
| <b>12</b> | <b>Network Programming</b>   | <b>607</b> |
| 12.1      | Client-Server Programming Model . . . . .  | 607        |
| 12.2      | Networks . . . . .   | 608        |
| 12.3      | The Global IP Internet . . . . .   | 613        |
| 12.3.1    | IP Addresses . . . . .   | 614        |
| 12.3.2    | Internet Domain Names . . . . .  | 616        |
| 12.3.3    | Internet Connections . . . . .   | 620        |
| 12.4      | Unix file I/O . . . . .  | 621        |
| 12.4.1    | The <code>read</code> and <code>write</code> Functions . . . . .                       | 622        |
| 12.4.2    | Robust File I/O With the <code>readn</code> and <code>writen</code> Functions. . . . . | 623        |
| 12.4.3    | Robust Input of Text Lines Using the <code>readline</code> Function . . . . .          | 625        |
| 12.4.4    | The <code>stat</code> Function . . . . .   | 625        |
| 12.4.5    | The <code>dup2</code> Function . . . . .   | 628        |
| 12.4.6    | The <code>close</code> Function . . . . .  | 629        |
| 12.4.7    | Other Unix I/O Functions . . . . .   | 630        |
| 12.4.8    | Unix I/O vs. Standard I/O . . . . .  | 630        |

|          |  |            |
|----------|--|------------|
| 12.5     | The Sockets Interface . . . . .                      | 631        |
| 12.5.1   | Socket Address Structures . . . . .                  | 631        |
| 12.5.2   | The <code>socket</code> Function . . . . .           | 633        |
| 12.5.3   | The <code>connect</code> Function . . . . .          | 633        |
| 12.5.4   | The <code>bind</code> Function . . . . .             | 635        |
| 12.5.5   | The <code>listen</code> Function . . . . .           | 635        |
| 12.5.6   | The <code>accept</code> Function . . . . .           | 637        |
| 12.5.7   | Example Echo Client and Server . . . . .             | 638        |
| 12.6     | Concurrent Servers . . . . .                         | 640        |
| 12.6.1   | Concurrent Servers Based on Processes . . . . .      | 640        |
| 12.6.2   | Concurrent Servers Based on Threads . . . . .        | 642        |
| 12.7     | Web Servers . . . . .                                | 648        |
| 12.7.1   | Web Basics . . . . .                                 | 649        |
| 12.7.2   | Web Content . . . . .                                | 649        |
| 12.7.3   | HTTP Transactions . . . . .                          | 650        |
| 12.7.4   | Serving Dynamic Content . . . . .                    | 653        |
| 12.8     | Putting it Together: The TINY Web Server . . . . .   | 654        |
| 12.9     | Summary . . . . .                                    | 664        |
| <b>A</b> | <b>Error handling</b>                                | <b>667</b> |
| A.1      | Introduction . . . . .                               | 667        |
| A.2      | Error handling in Unix systems . . . . .             | 668        |
| A.3      | Error-handling wrappers . . . . .                    | 669        |
| A.4      | The <code>csapp.h</code> header file . . . . .       | 673        |
| A.5      | The <code>csapp.c</code> source file . . . . .       | 677        |
| <b>B</b> | <b>Solutions to Practice Problems</b>                | <b>693</b> |
| B.1      | Intro . . . . .                                      | 693        |
| B.2      | Representing and Manipulating Information . . . . .  | 693        |
| B.3      | Machine Level Representation of C Programs . . . . . | 702        |
| B.4      | Processor Architecture . . . . .                     | 717        |
| B.5      | Optimizing Program Performance . . . . .             | 717        |
| B.6      | The Memory Hierarchy . . . . .                       | 719        |

|  |     |
|--|-----|
| B.7 Linking . . . . .                              | 725 |
| B.8 Exceptional Control Flow . . . . .             | 727 |
| B.9 Measuring Program Performance . . . . .        | 729 |
| B.10 Virtual Memory . . . . .                      | 731 |
| B.11 Concurrent Programming with Threads . . . . . | 736 |
| B.12 Network Programming . . . . .                 | 738 |

