Notes on Data Structures and Programming Techniques (CPSC 223, Spring 2018)

James Aspnes

2020-01-25T10:12:33-0500

Contents

1 Course administration **13**
1.1 Overview . . . 13
1.1.1 License . . . 13
1.1.2 Resources . . . 14
1.1.3 Documentation . . . 14
1.1.4 Questions and comments . . . 14
1.2 Lecture schedule . . . 15
1.3 Syllabus . . . 18
1.3.1 On-line course information . . . 18
1.3.2 Meeting times . . . 18
1.3.3 Synopsis of the course . . . 19
1.3.4 Prerequisites . . . 19
1.3.5 Textbook . . . 19
1.3.6 Course requirements . . . 19
1.3.7 Staff . . . 19
1.3.7.1 Instructor . . . 19
1.3.7.2 Teaching Fellow . . . 20
1.3.7.3 Undergraduate Learning Assistants . . . 20
1.3.8 Use of outside help . . . 20
1.3.9 Clarifications for homework assignments . . . 20
1.3.10 Late assignments . . . 21
1.4 Introduction . . . 21
1.4.1 Why should you learn to program in C? . . . 21
1.4.2 Why should you learn about data structures and programming techniques? . . . 22

2 The Zoo **22**
2.1 Getting an account . . . 23
2.2 Getting into the room . . . 23

2.3 Remote use . . . 23
2.3.1 Terminal access . . . 23
2.3.2 GUI access . . . 25
2.3.3 GUI access using FastX . . . 26
2.4 Developing on your own machine . . . 26
2.4.1 Linux . . . 26
2.4.2 OSX . . . 27
2.4.3 Windows . . . 27
2.5 How to compile and run programs . . . 27
2.5.1 Creating the program . . . 28
2.5.2 Compiling and running a program . . . 28
2.5.3 Some notes on what the program does . . . 29

3 The Linux programming environment 30
3.1 The shell . . . 30
3.1.1 Getting a shell prompt in the Zoo . . . 30
3.1.2 The Unix filesystem . . . 31
3.1.3 Unix command-line programs . . . 31
3.1.4 Stopping and interrupting programs . . . 32
3.1.5 Running your own programs . . . 33
3.1.6 Redirecting input and output . . . 33
3.2 Text editors . . . 34
3.2.1 Writing C programs with Emacs . . . 34
3.2.1.1 My favorite Emacs commands . . . 34
3.2.2 Using Vi instead of Emacs . . . 35
3.2.2.1 My favorite Vim commands . . . 36
3.2.2.1.1 Normal mode . . . 36
3.2.2.1.2 Insert mode . . . 37
3.2.2.2 Settings . . . 37
3.3 Compilation tools . . . 38
3.3.1 The GNU C compiler `gcc` . . . 38
3.3.2 Make . . . 38
3.3.2.1 Make gotchas . . . 40
3.4 Debugging tools . . . 40
3.4.1 Debugging in general . . . 40
3.4.2 Assertions . . . 41
3.4.3 The GNU debugger `gdb` . . . 41
3.4.3.1 My favorite gdb commands . . . 44
3.4.3.2 Debugging strategies . . . 45
3.4.3.3 Common applications of `gdb` . . . 45
3.4.3.3.1 Watching your program run . . . 45
3.4.3.3.2 Dealing with failed assertions . . . 46
3.4.3.3.3 Dealing with segmentation faults . . . 48
3.4.3.3.4 Dealing with infinite loops . . . 49
3.4.3.3.5 Mysterious variable changes . . . 49
3.4.4 Valgrind . . . 52

3.4.4.1 Compilation flags . . . 52
3.4.4.2 Automated testing . . . 52
3.4.4.3 Examples of some common valgrind errors . . . 53
3.4.4.3.1 Uninitialized values . . . 53
3.4.4.3.2 Bytes definitely lost . . . 54
3.4.4.3.3 Invalid write or read operations . . . 55
3.4.5 Not recommended: debugging output . . . 57
3.5 Performance tuning . . . 58
3.5.1 Timing under Linux . . . 59
3.5.2 Profiling with `valgrind` . . . 59
3.5.3 Profiling with `gprof` . . . 66
3.5.3.1 Effect of optimization during compilation . . . 73
3.6 Version control . . . 75
3.6.1 Setting up Git . . . 75
3.6.2 Editing files . . . 77
3.6.3 Renaming files . . . 78
3.6.4 Adding and removing files . . . 78
3.6.5 Recovering files from the repository . . . 79
3.6.6 Undoing bad commits . . . 80
3.6.7 Looking at old versions . . . 81
3.6.8 More information about Git . . . 82
3.7 Submitting assignments . . . 82

4 The C programming language 83
4.1 Structure of a C program . . . 84
4.2 Numeric data types . . . 91
4.2.1 Integer types in C . . . 92
4.2.1.1 Basic integer types . . . 92
4.2.1.2 Overflow and the C standards . . . 95
4.2.1.3 C99 fixed-width types . . . 95
4.2.2 `size_t` and `ptrdiff_t` . . . 97
4.2.2.1 Integer constants . . . 97
4.2.2.1.1 Naming constants . . . 98
4.2.2.2 Integer operators . . . 100
4.2.2.2.1 Arithmetic operators . . . 100
4.2.2.2.2 Bitwise operators . . . 101
4.2.2.2.3 Logical operators . . . 103
4.2.2.2.4 Relational operators . . . 103
4.2.2.3 Converting to and from strings . . . 104
4.2.3 Floating-point types . . . 105
4.2.3.1 Floating point basics . . . 105
4.2.3.2 Floating-point constants . . . 105
4.2.3.3 Operators . . . 106
4.2.3.4 Conversion to and from integer types . . . 106
4.2.3.5 The IEEE-754 floating-point standard . . . 107
4.2.3.6 Error . . . 108

4.2.3.7 Reading and writing floating-point numbers . . . 109
4.2.3.8 Non-finite numbers in C 109
4.2.3.9 The math library 110
4.3 Operator precedence 110
4.4 Programming style 111
4.5 Variables 113
4.5.1 Memory 114
4.5.2 Variables as names 114
4.5.2.1 Variable declarations 114
4.5.2.2 Variable names 116
4.5.3 Using variables 117
4.5.4 Initialization 118
4.5.5 Storage class qualifiers 119
4.5.5.1 Scope and extent 119
4.5.5.1.1 Additional qualifiers for global variables 120
4.5.6 Marking variables as constant 121
4.5.6.1 Pointers to `const` 121
4.6 Input and output 121
4.6.1 Character streams 122
4.6.2 Reading and writing single characters 122
4.6.3 Formatted I/O 124
4.6.4 Rolling your own I/O routines 125
4.6.5 File I/O 127
4.7 Statements and control structures 129
4.7.1 Simple statements 129
4.7.2 Compound statements 130
4.7.2.1 Conditionals 130
4.7.2.2 Loops 133
4.7.2.2.1 The while loop 133
4.7.2.2.2 The do..while loop 134
4.7.2.2.3 The for loop 134
4.7.2.2.4 Loops with break, continue, and goto . 136
4.7.2.3 Choosing where to put a loop exit 137
4.8 Functions 138
4.8.1 Function definitions 138
4.8.2 When to write a function 139
4.8.3 Calling a function 141
4.8.4 The return statement 141
4.8.5 Function declarations and modules 142
4.8.6 Static functions 143
4.8.7 Local variables 144
4.8.8 Mechanics of function calls 145
4.9 Pointers 146
4.9.1 Memory and addresses 146
4.9.2 Pointer variables 147
4.9.2.1 Declaring a pointer variable 147

4.9.2.2 Assigning to pointer variables 147
4.9.2.3 Using a pointer 148
4.9.2.4 Printing pointers 148
4.9.3 The null pointer 149
4.9.4 Pointers and functions 150
4.9.5 Pointer arithmetic and arrays 153
4.9.5.1 Arrays 153
4.9.5.2 Arrays and functions 154
4.9.5.3 Multidimensional arrays 155
4.9.5.4 Variable-length arrays 158
4.9.6 Pointers to void 161
4.9.6.1 Alignment 162
4.9.7 Run-time storage allocation using `malloc` 164
4.9.8 Function pointers 167
4.9.8.1 Function pointer declarations 167
4.9.8.2 Callbacks 168
4.9.8.3 Dispatch tables 169
4.9.9 The restrict keyword 170
4.10 Strings 171
4.10.1 C strings 172
4.10.2 String constants 172
4.10.2.1 String encodings 172
4.10.3 String buffers 173
4.10.3.1 String buffers and the perils of `gets` 174
4.10.4 Operations on strings 175
4.10.5 Finding the length of a string 178
4.10.5.1 The strlen tarpit 178
4.10.6 Comparing strings 179
4.10.7 Formatted output to strings 180
4.10.8 Dynamic allocation of strings 180
4.10.9 Command-line arguments 181
4.11 Structured data types 182
4.11.1 Structs 182
4.11.1.1 Operations on structs 186
4.11.1.2 Layout in memory 186
4.11.1.3 Bit fields 187
4.11.2 Unions 187
4.11.3 Enums 188
4.11.3.1 Specifying particular values 189
4.11.3.2 What most people do 189
4.11.3.3 Using `enum` with `union` 189
4.12 Type aliases using `typedef` 190
4.12.1 Opaque structs 191
4.13 Macros 193
4.13.1 Macros with arguments 194
4.13.1.1 Multiple arguments 194

Click here to download full PDF material