

Teil I: Die Umgebung	1
1 Richten Sie sich Ihre Umgebung ein	3
Einen Paketmanager einsetzen	4
C unter Windows kompilieren	6
POSIX für Windows	6
C mit POSIX kompilieren	8
C ohne POSIX kompilieren	8
Wo bitte geht es zur Bibliothek?	9
Ein paar meiner Lieblings-Flags	11
Pfade	12
Runtime-Linking	15
Makefiles verwenden	16
Variablen setzen	17
Die Regeln	19
Bibliotheken über ihren Quellcode nutzen	23
Bibliothek über ihren Quellcode nutzen – auch wenn Ihr Sysadmin das nicht will	24
C-Programme über Here-Dokumente kompilieren	26
Header-Dateien an der Befehlszeile einbinden	26
Der vereinheitlichte Header	27
Here-Dokumente	28
Von Stdin kompilieren	29

2 Debuggen, Testen, Dokumentieren	31
Einen Debugger verwenden	31
GDB-Variablen	35
Geben Sie Ihre Strukturen aus	37
Mit Valgrind auf Fehler prüfen	40
Unit-Tests	41
Ein Programm als Bibliothek verwenden	44
Abdeckung	46
Dokumentation einweben	47
Doxygen	47
Literaler Code mit CWEB	49
Fehlerprüfung	51
Wie ist der Anwender in den Fehler involviert?	51
Der Kontext, in dem der Anwender arbeitet	53
Wie sollte ein Hinweis auf einen Fehler zurückgegeben werden?	54
3 Verpacken Sie Ihr Projekt	55
Die Shell	56
Shell-Befehle durch ihre Ausgabe ersetzen	57
Die Shell für Schleifen nutzen, um auf einem Satz Dateien zu arbeiten	58
Dateien prüfen	60
fc	63
Makefiles vs. Shell-Skripten	65
Packen Sie Ihren Code mit den Autotools	67
Ein Autotools-Beispiel	69
Das Makefile durch <code>makefile.am</code> beschreiben	72
Das <code>configure</code> -Skript	76
4 Versionsverwaltung	81
Änderungen per <code>diff</code>	82
Git-Objekte	83
Der Stash	87
Bäume und ihre Zweige	88
Merging	89
Der Rebase	91
Remote-Repositories	92

5 Mit anderen zusammenspielen	95
Das Vorgehen	95
Schreiben, damit es von anderen Sprachen gelesen werden kann	95
Die Wrapper-Funktion	96
Datenstrukturen über die Grenze schmuggeln	97
Linken	98
Python als Host	99
Kompilieren und linken	100
Das bedingte Unterverzeichnis für Automake	101
Distutils mit Unterstützung durch die Autotools	102
<hr/>	
Teil II: Die Sprache	105
6 Ihr Weg zum Zeiger	107
Automatischer, statischer und manueller Speicher	107
Persistente Statusvariablen	110
Zeiger ohne malloc	111
Strukturen werden kopiert, Arrays werden als Alias weitergegeben	113
malloc und Speichertricks	115
Das Schicksal liegt in den Sternen	117
All die Zeigerarithmetik, die Sie kennen müssen	118
7 C-Syntax, die Sie ignorieren können	123
Kümmern Sie sich nicht darum, explizit aus main zurückzukehren	124
Lassen Sie Deklarationen fließen	124
Die Array-Größe zur Laufzeit setzen	126
Weniger Casting	127
Enums und Strings	128
Labels, goto, switch und break	130
Durchdachtes goto	131
switch	132
Veraltetes Float	135
8 Hindernisse und Gelegenheiten	139
Robuste und ansprechende Makros schreiben	139
Präprozessortricks	143
Mit static und extern verlinken	146
Extern zu verlinkende Elemente nur in Header-Dateien deklarieren	148

Das Schlüsselwort const	150
Nomen-Adjektiv-Form	151
Spannungen	152
Tiefe	153
Das Problem mit <code>char const**</code>	154
9 Text	157
Den Umgang mit Strings mithilfe von <code>asprintf</code> einfacher gestalten	157
Sicherheit	159
Konstante Strings	159
Strings mit <code>asprintf</code> erweitern	161
Ein Loblied auf <code>strtok</code>	162
Unicode	167
Das Kodieren für C-Code	169
Unicode-Bibliotheken	170
Der Beispielcode	171
10 Bessere Strukturen	175
Compound-Literale	176
Initialisierung per Compound-Literal	177
Variadische Makros	177
Listen sicher abschließen	179
foreach	180
Eine Funktion vektorisieren	180
Designated Initializers	182
Arrays und Structs mit Nullen initialisieren	184
Typedefs retten Ihnen den Tag	185
Über Stil	186
Mehrere Elemente aus einer Funktion zurückgeben	187
Fehler melden	189
Flexible Eingabewerte für Funktionen	191
Deklarieren Sie Ihre Funktion im <code>printf</code> -Stil	192
Optionale und benannte Argumente	193
Eine alte Funktion aufpolieren	195
Der <code>void</code> -Zeiger und die Strukturen, auf die er zeigt	201
Funktionen mit generischen Eingabewerten	201
Generische Strukturen	206

11 Objektorientierte Programmierung in C	211
Was Sie nicht bekommen (und warum Sie es nicht vermissen werden)	212
Gültigkeitsbereich	212
Überladen mit Operator-Überladung	215
Strukturen und Dictionaries erweitern	220
Eine Struktur erweitern	221
Ein Dictionary implementieren	225
Lassen Sie Ihren Code auf Zeigern auf Objekte basieren	229
Funktionen in Ihren Structs	230
Referenzen zählen	235
Beispiel: Ein Substring-Objekt	235
Ein agentenbasiertes Modell der Gruppenbildung	239
12 Bibliotheken	247
GLib	247
POSIX	248
Mit mmap riesige Datensätze verarbeiten	248
Einfaches Threading mit Pthreads	251
Die GNU Scientific Library	259
SQLite	261
Die Abfragen	262
libxml und cURL	264
Epilog	269
Glossar	271
Bibliografie	275
Index	277