

# C von A bis Z

Das umfassende Handbuch

# DAS INHALTS- VERZEICHNIS

» Hier geht's  
direkt  
zum Buch

# Inhalt

Vorwort .....	25
<b>1 Einstieg in C</b> .....	<b>27</b>
<b>1.1 Übersicht zu C</b> .....	<b>27</b>
<b>1.2 Der ANSI-C-Standard</b> .....	<b>29</b>
1.2.1 Welcher C-Standard wird in diesem Buch verwendet? .....	31
1.2.2 Der Vorteil des ANSI-C-Standards .....	32
<b>1.3 Der POSIX-Standard</b> .....	<b>32</b>
<b>1.4 Vor- und Nachteile der Programmiersprache C</b> .....	<b>33</b>
<b>1.5 C in diesem Buch</b> .....	<b>34</b>
<b>1.6 Was benötigen Sie für C?</b> .....	<b>35</b>
1.6.1 Texteditor .....	35
1.6.2 Compiler .....	35
1.6.3 All in one – die Entwicklungsumgebung .....	36
<b>1.7 Welcher Compiler und welches Betriebssystem?</b> .....	<b>37</b>
<b>1.8 Unterschiede zwischen Windows und Linux</b> .....	<b>38</b>
<b>1.9 Eine Programmierumgebung einrichten am Beispiel von Pelles C 8.00 für Windows</b> .....	<b>38</b>
1.9.1 Pelles C installieren .....	39
1.9.2 Ein einfaches C-Projekt mit Pelles C anlegen .....	43
<b>1.10 Eine Programmierumgebung einrichten – Nano/GCC für Linux</b> .....	<b>47</b>
<b>2 Eine günstige Programmierplattform – der Raspberry Pi</b> .....	<b>49</b>
<b>2.1 Was genau ist der Raspberry Pi?</b> .....	<b>50</b>
2.1.1 Die Hardware des Raspberry Pi .....	50
2.1.2 Welche Betriebssysteme gibt es für den Raspberry Pi? .....	52

<b>2.2</b>	<b>Installation eines Linux-Betriebssystems am Beispiel von Raspberry Pi OS</b> .....	54
2.2.1	Was genau ist Raspberry Pi OS, und was genau ist Debian? .....	54
2.2.2	Raspberry Pi OS aus dem Internet herunterladen .....	55
2.2.3	Das Raspberry Pi OS installieren .....	57
2.2.4	Den Pi mit der Grundkonfiguration starten .....	60
2.2.5	Wie oft benötigen Sie Updates, und wie gehen Sie vor? .....	61
<b>2.3</b>	<b>Die Konfiguration des Raspberry Pi im Detail</b> .....	63
<b>2.4</b>	<b>Das Tool raspi-config</b> .....	63
2.4.1	Das Tool raspi-config im Detail .....	64

## **3 Das erste Programm** 69

---

<b>3.1</b>	<b>Was leistet der Editor?</b> .....	70
<b>3.2</b>	<b>Was leistet der Compiler?</b> .....	70
<b>3.3</b>	<b>Was sind Include-Dateien, und wozu benötigt man sie?</b> .....	71
<b>3.4</b>	<b>Das erste Programm – die Ausgabe eines Textes in der Konsole</b> .....	71
<b>3.5</b>	<b>Das Hallo-Welt-Programm im Detail</b> .....	72
<b>3.6</b>	<b>Wie kompiliert man ein Programm und führt es anschließend aus?</b> .....	75

## **4 Grundlagen** 77

---

<b>4.1</b>	<b>Symbole</b> .....	77
4.1.1	Bezeichner .....	77
4.1.2	Schlüsselwörter .....	78
4.1.3	Literale .....	78
4.1.4	Einfache Begrenzer .....	80
4.1.5	Kommentare .....	81
<b>4.2</b>	<b>Zeichensätze</b> .....	84
4.2.1	Basis-Zeichensatz .....	84
4.2.2	Ausführungszeichensatz (Steuerzeichen) .....	85
4.2.3	Trigraph-Zeichen .....	88
<b>4.3</b>	<b>Strings</b> .....	89
<b>4.4</b>	<b>Das Einführungsbeispiel mit Strings</b> .....	90

<b>5</b>	<b>Basisdatentypen</b>	91
5.1	Was ist ein Byte, und wie werden Zahlen in C gespeichert?	91
5.2	Was ist eine Variable, und was ist ein Variablentyp?	92
5.3	Was ist ein Operand?	93
5.4	Was ist ein Parameter?	93
5.5	Wie speichert ein Prozessor Variablen, Parameter und Operanden?	93
5.6	Was sind Speicheradressen, und warum sind sie wichtig für C?	93
5.7	Deklaration und Definition von Variablen	94
5.8	Array-Datentypen (Datenfelder)	96
5.9	Standard-Datentypen	96
5.9.1	Der Datentyp »int« (Integer), Standardgröße 2 Byte	97
5.9.2	Verwendung des Datentyps »int«	98
5.9.3	Der Datentyp »long«, Standardgröße 4 Byte	101
5.9.4	Der Datentyp »long long«, Standardgröße 8 Byte	102
5.9.5	Der Datentyp »short«, Größe 2 Byte	102
5.10	Ganzzahlige Typen mit vorgegebener Breite – <stdint.h>	103
5.10.1	<inttypes.h> (ab C99)	105
5.11	Die Gleitpunkttypen »float« und »double«	107
5.11.1	Gleitpunkttypen im Detail	109
5.11.2	»float« im Detail	109
5.11.3	»double« im Detail	110
5.11.4	»long double«	110
5.11.5	long long double	111
5.11.6	Einiges zu n-stelliger Genauigkeit	111
5.11.7	Numerische Gleitpunktprobleme	113
5.12	Komplexe Gleitpunkttypen – <complex.h>	115
5.13	Der Datentyp »char«, Standardgröße 1 Byte	117
5.14	Nationale contra internationale Zeichensätze	123
5.14.1	Der Breitzichen-Typ »wchar_t«, Größe 2 Byte	125
5.14.2	Multibyte-Zeichen, Größe 2, 4 oder 8 Byte	126
5.15	Boolesche Werte – <stdbool.h>	127
5.16	Der Unterschied zwischen vorzeichenlosen und vorzeichenbehafteten Zahlen	128
5.17	Limits für Ganzzahl- und Gleitpunktdatentypen	130
5.18	Der Typ »void«	134

<b>5.19 Numerische Konstanten</b> .....	134
5.19.1 Ganzzahlkonstanten .....	134
5.19.2 Gleitpunktkonstanten .....	135
5.19.3 Zeichenkonstanten .....	135
5.19.4 Stringlitterale (Stringkonstanten) .....	136
<b>5.20 Umwandlungsvorgaben für formatierte Ein-/Ausgabe</b> .....	136

## **6 Formatierte Ein-/Ausgabe mit »scanf()« und »printf()«** 145

---

<b>6.1 Formatierte Eingabe mit »scanf()«</b> .....	145
6.1.1 Der Adressoperator »&« .....	147
6.1.2 Probleme und ihre Behandlung mit »scanf()« .....	149
6.1.3 Überprüfen auf das richtige Format .....	152
6.1.4 Zusammenfassung zu »scanf()« .....	154
<b>6.2 Formatierte Ausgabe mit »printf()«</b> .....	155

## **7 Operatoren** 159

---

<b>7.1 Was sind Operatoren, und wozu benötigt man sie in C?</b> .....	159
<b>7.2 Arithmetische Operatoren</b> .....	160
7.2.1 Dividieren von Ganzzahlen .....	162
<b>7.3 Erweiterte Darstellung arithmetischer Operatoren</b> .....	164
<b>7.4 Inkrement- und Dekrement-Operatoren</b> .....	165
<b>7.5 Bit-Operatoren</b> .....	167
7.5.1 Bitweises UND .....	167
7.5.2 Bitweises ODER .....	169
7.5.3 Bitweises XOR .....	170
7.5.4 Bitweises Komplement .....	171
7.5.5 Linksverschiebung .....	171
7.5.6 Rechtsverschiebung .....	173
7.5.7 Linksrotation (erst ab C23) .....	173
7.5.8 Rechtsrotation (erst ab C23) .....	173
7.5.9 Rezept für Fortgeschrittene .....	173
<b>7.6 Makros für logische Operatoren und Bit-Operatoren – &lt;iso646.h&gt;</b> .....	175
<b>7.7 Der »sizeof«-Operator</b> .....	176
7.7.1 C versus C++ .....	178

<b>8</b>	<b>Typumwandlung</b>	179
<b>8.1</b>	<b>Automatische implizite Datentypumwandlung durch den C-Compiler</b> ....	179
8.1.1	Implizites »char« nach »int« .....	180
8.1.2	Implizites »float« nach »double« .....	180
8.1.3	Implizite Umwandlung in einen komplexen Gleitpunkttyp .....	180
8.1.4	Übliche arithmetische Datentypumwandlung .....	181
<b>8.2</b>	<b>Wozu benötigt man das benutzerdefinierte explizite Typcasting?</b> .....	184
<b>8.3</b>	<b>Explizite Datentypumwandlung mit Typcasting an Beispielen</b> .....	184
<b>9</b>	<b>Kontrollstrukturen</b>	187
<b>9.1</b>	<b>Verzweigungen mit der »if«-Bedingung</b> .....	188
9.1.1	Anweisungsblöcke .....	188
<b>9.2</b>	<b>Die Verzweigung mit »else if«</b> .....	192
<b>9.3</b>	<b>Die Verzweigung mit »else«</b> .....	194
<b>9.4</b>	<b>Der !-Operator (logischer NOT-Operator)</b> .....	198
<b>9.5</b>	<b>Logisches UND (&amp;&amp;) – logisches ODER (  )</b> .....	200
<b>9.6</b>	<b>Der Bedingungsoperator »?:«</b> .....	203
<b>9.7</b>	<b>Fallunterscheidung: die »switch«-Verzweigung</b> .....	205
9.7.1	default .....	208
<b>9.8</b>	<b>Die »while«-Schleife</b> .....	210
9.8.1	Endlosschleifen mit »while« .....	212
9.8.2	Fehlervermeidung bei »while«-Schleifen .....	213
<b>9.9</b>	<b>Die »do while«-Schleife</b> .....	214
<b>9.10</b>	<b>Die »for«-Schleife</b> .....	219
9.10.1	Einsatzmöglichkeiten der »for«-Schleife .....	222
<b>9.11</b>	<b>Kontrollierter Ausstieg aus Schleifen mit »break«</b> .....	227
9.11.1	continue .....	227
9.11.2	break .....	229
<b>9.12</b>	<b>Direkte Sprünge mit »goto«</b> .....	229
<b>9.13</b>	<b>Einige Anmerkungen zum Notationsstil, ehe Sie weiterlesen</b> .....	229
<b>9.14</b>	<b>Einige Anmerkungen zu einem guten Programmierstil</b> .....	230

---

<b>10 Funktionen</b>	233
<b>10.1 Was sind Funktionen?</b>	233
<b>10.2 Wozu dienen Funktionen?</b>	234
<b>10.3 Definition von Funktionen</b>	234
<b>10.4 Funktionsaufruf</b>	235
<b>10.5 Funktionsdeklaration</b>	237
<b>10.6 Lokale Variablen</b>	239
<b>10.7 Globale Variablen</b>	241
<b>10.8 Statische Variablen</b>	243
<b>10.9 Schlüsselwörter für Variablen – Speicherklassen</b>	244
10.9.1 auto	244
10.9.2 extern	245
10.9.3 register	245
10.9.4 static	245
<b>10.10 Typ-Qualifizierer</b>	245
10.10.1 volatile	246
10.10.2 const	246
<b>10.11 Geltungsbereich von Variablen</b>	246
<b>10.12 Speicherklassen-Spezifizierer für Funktionen</b>	248
10.12.1 extern	248
10.12.2 static	248
10.12.3 volatile	248
<b>10.13 Datenaustausch zwischen Funktionen</b>	249
<b>10.14 Wertübergabe an Funktionen (Call-by-Value)</b>	250
<b>10.15 Der Rückgabewert von Funktionen</b>	254
<b>10.16 Die Hauptfunktion »main()«</b>	256
<b>10.17 Was bedeutet der Rückgabewert beim Beenden eines Programms?</b>	258
10.17.1 Programmende auswerten	259
<b>10.18 Funktionen der Laufzeitbibliothek</b>	262
<b>10.19 Getrenntes Kompilieren von Quelldateien</b>	263
<b>10.20 Rekursive Funktionen (Rekursion)</b>	265
10.20.1 Exkurs: Stack	265
10.20.2 Rekursionen und der Stack	266
10.20.3 Fakultät	272

10.20.4	Fibonacci-Zahlen .....	272
10.20.5	Größter gemeinsamer Teiler (ggT) .....	273
<b>10.21</b>	<b>»inline«-Funktionen .....</b>	<b>278</b>
<b>11</b>	<b>Präprozessor-Direktiven .....</b>	<b>281</b>
<hr/>		
<b>11.1</b>	<b>Mit »#include« Dateien einbinden .....</b>	<b>282</b>
<b>11.2</b>	<b>Wichtige vordefinierte Headerdateien für C .....</b>	<b>283</b>
<b>11.3</b>	<b>Makros und Konstanten – »#define« .....</b>	<b>285</b>
11.3.1	Symbolische Konstanten mit »#define« .....	285
11.3.2	Makros mit »#define« .....	290
<b>11.4</b>	<b>Bedingte Kompilierung mit »#ifdef« .....</b>	<b>294</b>
<b>11.5</b>	<b>Vordefinierte Präprozessor-Direktiven (ANSI C) .....</b>	<b>300</b>
<b>11.6</b>	<b>Einen Makroparameter durch einen String ersetzen .....</b>	<b>302</b>
<b>11.7</b>	<b>»#undef« – Makronamen wieder aufheben .....</b>	<b>304</b>
<b>11.8</b>	<b>Fehlermeldungen ausgeben mit »#error« .....</b>	<b>304</b>
<b>11.9</b>	<b>»#pragma« (gilt nur für Windows) .....</b>	<b>306</b>
<b>12</b>	<b>Arrays .....</b>	<b>307</b>
<hr/>		
<b>12.1</b>	<b>Arrays deklarieren .....</b>	<b>308</b>
<b>12.2</b>	<b>Initialisierung und Zugriff auf Arrays .....</b>	<b>309</b>
12.2.1	Gültigkeitsbereich von Arrays .....	315
<b>12.3</b>	<b>Arrays vergleichen .....</b>	<b>316</b>
<b>12.4</b>	<b>Anzahl der Elemente eines Arrays ermitteln .....</b>	<b>318</b>
<b>12.5</b>	<b>Übergabe von Arrays an Funktionen .....</b>	<b>319</b>
<b>12.6</b>	<b>Arrays aus Funktionen zurückgeben .....</b>	<b>322</b>
<b>12.7</b>	<b>Programmbeispiel zu Arrays .....</b>	<b>323</b>
<b>12.8</b>	<b>Array-Werte über die Tastatur einlesen .....</b>	<b>327</b>
<b>12.9</b>	<b>Mehrdimensionale Arrays .....</b>	<b>328</b>
12.9.1	Mehrdimensionale Arrays initialisieren .....	328
12.9.2	Übergabe von zwei- bzw. mehrdimensionalen Arrays an Funktionen .....	341



<b>12.10 Arrays in eine Tabellenkalkulation einlesen (CSV-Dateien)</b> .....	343
<b>12.11 Strings/Zeichenketten (»char«-Array)</b> .....	345
12.11.1 Vom String zur Binärzahl .....	349
<b>12.12 Einlesen von Strings</b> .....	352
<b>12.13 Die Standard-Bibliothek &lt;string.h&gt;</b> .....	355
12.13.1 »strcat()« – Strings aneinanderhängen .....	355
12.13.2 »strchr()« – ein Zeichen im String suchen .....	356
12.13.3 »strcmp()« – Strings vergleichen .....	357
12.13.4 »strcpy()« – einen String kopieren .....	358
12.13.5 »strcspn()« – einen Teilstring ermitteln .....	359
12.13.6 »strlen()« – Länge eines Strings ermitteln .....	360
12.13.7 »strncat()« – String mit n Zeichen aneinanderhängen .....	361
12.13.8 »strncmp()« – n Zeichen von zwei Strings miteinander vergleichen	361
12.13.9 »strncpy()« – String mit n Zeichen kopieren .....	362
12.13.10 »strpbrk()« – nach dem Auftreten bestimmter Zeichen suchen .....	363
12.13.11 »strrchr()« – das letzte Auftreten eines bestimmten Zeichens im String suchen .....	364
12.13.12 »strspn()« – das erste Auftreten eines Zeichens, das nicht vorkommt .....	364
12.13.13 »strstr()« – einen String nach dem Auftreten eines Teilstrings durchsuchen .....	365
12.13.14 »strtok()« – einen String anhand bestimmter Zeichen zerlegen .....	366

## **13 Zeiger (Pointer)** 369

---

<b>13.1 Der Unterschied zwischen einer normalen Variablen und einem Zeiger</b> .....	369
<b>13.2 Zeiger deklarieren</b> .....	370
<b>13.3 Zeiger initialisieren</b> .....	371
13.3.1 Speichergröße von Zeigern .....	383
<b>13.4 Zeigerarithmetik</b> .....	384
<b>13.5 Zeiger, die auf andere Zeiger verweisen</b> .....	385
13.5.1 Subtraktion zweier Zeiger .....	387
<b>13.6 Typensicherung bei der Dereferenzierung</b> .....	388
<b>13.7 Zeiger als Funktionsparameter (Call-by-Reference)</b> .....	389
13.7.1 Zeiger als Rückgabewert .....	392
<b>13.8 Array und Zeiger</b> .....	396

<b>13.9 Zeiger auf Strings</b> .....	403
13.9.1 Zeiger auf konstante Objekte (Read-only-Zeiger) .....	404
<b>13.10 Zeiger auf Zeiger und Stringtabellen</b> .....	405
13.10.1 Stringtabellen .....	407
<b>13.11 Zeiger auf Funktionen</b> .....	415
<b>13.12 void-Zeiger</b> .....	420
<b>13.13 Äquivalenz zwischen Zeigern und Arrays</b> .....	424
<b>13.14 Der »restrict«-Zeiger</b> .....	426
<b>14 Kommandozeilenargumente</b> .....	429
<b>14.1 Argumente an die Hauptfunktion übergeben</b> .....	430
<b>14.2 Argumente aus der Kommandozeile auswerten</b> .....	435
<b>15 Dynamische Speicherverwaltung</b> .....	441
<b>15.1 Das Speicherkonzept</b> .....	442
15.1.1 Codespeicher .....	442
15.1.2 Daten-Speicher .....	443
15.1.3 Stack-Speicher .....	443
15.1.4 Heap-Speicher .....	443
<b>15.2 Speicherallokation mit »malloc()«</b> .....	444
<b>15.3 Das NULL-Mysterium</b> .....	447
15.3.1 NULL für Fortgeschrittene .....	448
15.3.2 Was jetzt – NULL, 0 oder \0 ...? .....	449
15.3.3 Zusammengefasst .....	450
<b>15.4 Speicherreservierung und ihre Probleme</b> .....	451
<b>15.5 »free()« – Speicher wieder freigeben</b> .....	452
15.5.1 Einige ergänzende Punkte zur Freispeicherverwaltung .....	456
15.5.2 Prozessinterne Freispeicherverwaltung .....	457
<b>15.6 Dynamische Arrays</b> .....	460
<b>15.7 Speicher dynamisch reservieren mit »realloc()« und »calloc()«</b> .....	464
<b>15.8 Speicher vom Stack anfordern mit »alloca()« (nicht ANSI C)</b> .....	468

<b>15.9</b>	<b>Ergänzende Anmerkungen zu »free()«</b> .....	468
<b>15.10</b>	<b>Zweidimensionale dynamische Arrays</b> .....	469
<b>15.11</b>	<b>Was muss man tun, wenn die Speicherallokation fehlschlägt?</b> .....	472
15.11.1	Speicheranforderung reduzieren .....	473
15.11.2	Speicheranforderungen aufteilen .....	474
15.11.3	Einen Puffer konstanter Größe verwenden .....	476
15.11.4	Vor der Allokation auf eine Festplatte zwischenspeichern .....	476
15.11.5	Nur so viel Speicher anfordern wie nötig .....	476

## **16 Strukturen** 479

---

<b>16.1</b>	<b>Benutzerdefinierte Typdefinitionen mit »typedef«</b> .....	479
<b>16.2</b>	<b>Was ist ein strukturierter Datentyp, und wozu benötigt man ihn in C?</b> .....	479
<b>16.3</b>	<b>Strukturen mit »typedef struct« deklarieren</b> .....	480
<b>16.4</b>	<b>Initialisierung und Zugriff auf Strukturen</b> .....	482
16.4.1	Bestimmte Elemente initialisieren .....	487
<b>16.5</b>	<b>Strukturen als Wertübergabe an eine Funktion benutzen</b> .....	491
<b>16.6</b>	<b>Strukturen als Rückgabewert einer Funktion benutzen</b> .....	493
<b>16.7</b>	<b>Strukturen miteinander vergleichen</b> .....	495
<b>16.8</b>	<b>Arrays von Strukturen erstellen</b> .....	497
<b>16.9</b>	<b>Strukturen in Strukturen (Nested Structures)</b> .....	504
<b>16.10</b>	<b>Weitere spezielle Datentypen</b> .....	514
16.10.1	Unions .....	515
16.10.2	Der Aufzählungstyp »enum« .....	521
16.10.3	Weitere Typdefinitionen mit »typedef« .....	523
<b>16.11</b>	<b>Attribute von Strukturen verändern</b> .....	524
<b>16.12</b>	<b>Bitfelder</b> .....	528
<b>16.13</b>	<b>Das »offsetof«-Makro</b> .....	535

## **17 Ein-/Ausgabe-Funktionen** 537

---

<b>17.1</b>	<b>Was ist eine Datei aus Sicht der Programmiersprache C?</b> .....	537
<b>17.2</b>	<b>Formatierte und unformatierte Ein-/Ausgabe</b> .....	538

<b>17.3 Höhere Ein-/Ausgabe-Funktionen mit &lt;stdio.h&gt;</b> .....	538
17.3.1 Eine Datei mit »fopen« öffnen .....	540
17.3.2 Zeichenweise lesen und schreiben – »getchar()« und »putchar()« .....	546
17.3.3 Zeichenweise lesen und schreiben – »putc()«/»fputc()« und »getc()«/»fgetc()« .....	551
17.3.4 Eine geöffnete Datei mit »fclose()« wieder schließen (Wichtig!) .....	557
17.3.5 Formatierte Ein-/Ausgaben mit »fprintf()« und »fscanf()« .....	561
<b>17.4 Streams (Ein-/Ausgabe-Datenströme)</b> .....	566
17.4.1 Was unterscheidet einen Stream von einer Datei? .....	566
17.4.2 Standard-Streams in C .....	567
17.4.3 Fehlerbehandlung von Streams – »feof()«, »ferror()« und »clearerr()« .....	570
17.4.4 Gelesene Zeichen in die Eingabe zurückschieben – »ungetc()« .....	572
17.4.5 Den Tastaturpuffer leeren – »fflush()« .....	574
17.4.6 Einen Stream positionieren – »fseek()«, »rewind()« und »ftell()« .....	575
17.4.7 Einen Stream positionieren – »fsetpos()«, »fgetpos()« .....	579
17.4.8 Zeilenweise Ein-/Ausgabe in Streams .....	581
<b>17.5 Spezielle Ein-/Ausgabe-Funktionen für Streams</b> .....	595
17.5.1 Blockweise lesen und schreiben – »fread()« und »fwrite()« .....	596
17.5.2 Eine Datei erneut öffnen – »freopen()« .....	606
17.5.3 Eine Datei löschen oder umbenennen – »remove()« und »rename()« .....	608
17.5.4 Pufferung einstellen – »setbuf()« und »setvbuf()« .....	611
17.5.5 Temporäre Dateien erzeugen – »tmpfile()« und »tmpnam()« .....	618
17.5.6 Fehler bei der Ein-/Ausgabe abfangen und behandeln .....	623
17.5.7 Formatiert in einen String schreiben und formatiert aus einem String lesen – »sscanf()« und »sprintf()« .....	627
17.5.8 Ein fortgeschrittenes Thema .....	633
<b>17.6 Low-Level-Datei-I/O-Funktionen (nicht ANSI C)</b> .....	641
17.6.1 Dateien öffnen mit »open()« .....	642
17.6.2 Dateien schließen mit »close()« .....	649
17.6.3 Datei erzeugen – »creat()« .....	650
17.6.4 Schreiben und lesen – »write()« und »read()« .....	651
17.6.5 Den File-Deskriptor mit »lseek()« positionieren .....	662
17.6.6 Einen File-Deskriptor von einem Stream mit »fileno()« ableiten .....	663
17.6.7 Stream von File-Deskriptor mit »fdopen()« ableiten .....	664
17.6.8 Gerätedateien unter Linux – Zugriff auf Maus und Framebuffer .....	666

---

## 18 Attribute von Dateien und das Arbeiten mit Verzeichnissen (nicht ANSI C) 673

---

<b>18.1 Die Attribute einer Datei mit »stat()« ermitteln</b> .....	673
18.1.1 »stat()« – »st_mode« .....	674
18.1.2 »stat()« – »st_size« .....	680
18.1.3 »stat()« – »st_atime«, »st_mtime« und »st_ctime« .....	682
18.1.4 »stat()« – »st_gid« und »st_uid« .....	686
18.1.5 »stat()« – »st_nlink«, »st_ino« .....	687
18.1.6 »stat()« – »st_dev«, »st_rdev« .....	688
<b>18.2 Prüfen des Zugriffsrechts mit »access()«</b> .....	691
<b>18.3 Verzeichnisfunktionen</b> .....	693
18.3.1 Verzeichnis erstellen, löschen und wechseln mit »mkdir()«, »rmdir« und »chdir« .....	693
18.3.2 In das Arbeitsverzeichnis wechseln mit »getcwd()« .....	699
18.3.3 Verzeichnisse öffnen, lesen und schließen – »opendir()«, »readdir()« und »closedir()« .....	701

---

## 19 Arbeiten mit variabel langen Argumentlisten – <stdarg.h> 705

---

<b>19.1 Was ist eine Ellipse, und wie werden mit ihr Parameter übergeben?</b> .....	705
<b>19.2 Wie kann man mit &lt;stdarg.h&gt; die Parameterliste abfragen?</b> .....	706
<b>19.3 Makros in &lt;stdarg.h&gt; – »va_list«, »va_arg«, »va_start« und »va_end«</b> ....	706
<b>19.4 Die Argumentliste am Anfang oder Ende kennzeichnen</b> .....	707
<b>19.5 »vprintf()«, »vsprintf()«, »vfprintf()« und »vsnprintf()«</b> .....	712
<b>19.6 Variadic Makros – __VA_ARGS__</b> .....	716

---

## 20 Zeitroutinen 721

---

<b>20.1 Die Headerdatei &lt;time.h&gt;</b> .....	721
20.1.1 Konstanten in der Headerdatei <time.h> .....	723
20.1.2 Datums- und Zeitfunktionen in <time.h> .....	723

<b>20.2 Laufzeitmessung (Profiling)</b> .....	734
<b>20.3 Besonderheiten beim Raspberry Pi</b> .....	735

## **21 Weitere Headerdateien und ihre Funktionen (ANSI C)** 737

---

<b>21.1 Testmöglichkeiten und Fehlersuche – &lt;assert.h&gt;</b> .....	738
<b>21.2 Zeichenklassifizierung und Umwandlung – &lt;ctype.h&gt;</b> .....	739
<b>21.3 Mathematische Funktionen – &lt;math.h&gt;, &lt;tgmath.h&gt; und &lt;complex.h&gt;</b> .....	744
21.3.1 Funktionen für reelle und komplexe Gleitpunkttypen .....	746
21.3.2 Funktionen nur für reelle Gleitpunkttypen .....	747
21.3.3 Funktionen nur für komplexe Gleitpunkttypen .....	749
21.3.4 Typengenerische Makros – <tgmath.h> .....	751
21.3.5 Gleitpunktwerte klassifizieren .....	752
21.3.6 Makro zum Vergleichen von reellen Zahlen .....	753
21.3.7 Zugriff auf die Gleitpunktumgebung – <fenv.h> .....	754
<b>21.4 Einige nützliche Funktionen in &lt;stdlib.h&gt;</b> .....	758
21.4.1 Programmbeendigung – »exit()«, »_exit()«, »atexit()« und »abort()« .....	758
21.4.2 Strings in numerische Werte konvertieren .....	762
21.4.3 Bessere Alternative – Strings in numerische Werte konvertieren .....	764
21.4.4 Zufallszahlen .....	769
21.4.5 Absolutwerte, der Quotient und der Rest von Divisionen .....	771
21.4.6 Suchen und Sortieren – »qsort()« und »bsearch()« .....	773
21.4.7 »system()« .....	776
<b>21.5 Länderspezifische Eigenheiten – &lt;locale.h&gt;</b> .....	778
<b>21.6 Nichtlokale Sprünge – &lt;setjmp.h&gt;</b> .....	782
<b>21.7 Einige nützliche Funktionen in &lt;signal.h&gt;</b> .....	786
<b>21.8 Die »mem...«-Funktionen zur Speicher manipulation – &lt;string.h&gt;</b> .....	791
21.8.1 »memchr()« – Suche nach einzelnen Zeichen .....	792
21.8.2 »memcmp()« – bestimmte Anzahl von Bytes vergleichen .....	792
21.8.3 »memcpy()« – bestimmte Anzahl von Bytes kopieren .....	793
21.8.4 »memmove()« – bestimmte Anzahl von Bytes kopieren .....	794
21.8.5 »memset()« – Speicherbereich mit bestimmten Zeichen auffüllen .....	795

## **22 Dynamische Datenstrukturen** 797

---

<b>22.1 Lineare Listen (einfach verkettete Listen)</b> .....	797
22.1.1 Erstes Element der Liste löschen .....	806
22.1.2 Ein beliebiges Element in der Liste löschen .....	807
22.1.3 Elemente der Liste ausgeben .....	810
22.1.4 Eine vollständige Liste auf einmal löschen .....	816
22.1.5 Element in die Liste einfügen .....	818
<b>22.2 Doppelt verkettete Listen</b> .....	832
<b>22.3 Stacks nach dem LIFO-(Last-in-first-out-)Prinzip</b> .....	849
<b>22.4 Queues nach dem FIFO-Prinzip</b> .....	870
<b>22.5 Dynamisches Array mit flexiblen Elementen</b> .....	879

## **23 Algorithmen** 881

---

<b>23.1 Was sind Algorithmen?</b> .....	881
<b>23.2 Einige einfache Beispiele für Algorithmen</b> .....	883
23.2.1 Sortieralgorithmen .....	883
23.2.2 Suchalgorithmen .....	911
23.2.3 Pattern Matching .....	941
23.2.4 Pattern Matching durch reguläre Ausdrücke .....	952
23.2.5 Backtracking .....	959
23.2.6 Der Weg durch den Irrgarten .....	959
<b>23.3 Kryptografische Algorithmen</b> .....	969
23.3.1 Kryptografisches Hashing .....	970
23.3.2 Lineare Verschlüsselungsalgorithmen .....	982
23.3.3 Public-Key-Verfahren .....	1010

## **24 MySQL und C** 1027

---

<b>24.1 Aufbau eines Datenbanksystems</b> .....	1027
24.1.1 Warum wurden Datenbanksysteme (DBS) entwickelt? .....	1027
24.1.2 Das Datenbank-Management-System (DBMS) .....	1028
24.1.3 Relationale Datenbanken .....	1031
24.1.4 Mit C eigene Clients für SQL entwickeln mithilfe der ODBC-API .....	1032

---

<b>24.2</b>	<b>MySQL installieren</b> .....	1033
24.2.1	Linux .....	1033
24.2.2	Windows .....	1034
24.2.3	Den Client »mysql« starten .....	1035
<b>24.3</b>	<b>Crashkurs SQL</b> .....	1037
24.3.1	Was ist SQL? .....	1037
24.3.2	Die Datentypen von (My)SQL .....	1037
24.3.3	Eine Datenbank erzeugen .....	1040
24.3.4	Eine Datenbank löschen .....	1041
24.3.5	Die Datenbank wechseln .....	1042
24.3.6	Eine Tabelle erstellen .....	1042
24.3.7	Eine Tabelle anzeigen .....	1043
24.3.8	Tabellendefinition überprüfen .....	1043
24.3.9	Tabellen löschen .....	1043
24.3.10	Ein Backup erstellen .....	1044
24.3.11	Die Struktur einer Tabelle ändern .....	1044
24.3.12	Einzelne Datensätze hinzufügen .....	1045
24.3.13	Bestimmte Datensätze auswählen .....	1045
24.3.14	Ein fortgeschrittenes Szenario .....	1046
24.3.15	Datensätze löschen .....	1048
24.3.16	Datensatz ändern .....	1048
24.3.17	Zusätzliche Zugriffsrechte in MySQL .....	1049
24.3.18	Übersicht über einige SQL-Kommandos .....	1050
<b>24.4</b>	<b>Die MySQL-C-API</b> .....	1052
24.4.1	Grundlagen zur Programmierung eines MySQL-Clients .....	1052
24.4.2	Client-Programm mit dem GCC unter Linux und dem Cygwin-GCC-Compiler unter Windows .....	1053
24.4.3	MySQL Client-Programme mit dem VC++ Compiler und dem Borland Freeware Compiler .....	1054
24.4.4	Pelles C .....	1055
24.4.5	Troubleshooting .....	1056
24.4.6	Das erste Client-Programm – Verbindung mit dem MySQL-Server herstellen .....	1056
24.4.7	MySQL-Kommandozeilen-Optionen .....	1061
24.4.8	Anfrage an den Server .....	1064
<b>24.5</b>	<b>MySQL und C mit CGI</b> .....	1083
24.5.1	HTML-Eingabeformular .....	1083
24.5.2	Die CGI-Anwendung »add_db.cgi« .....	1085
24.5.3	Die CGI-Anwendung »search_db.cgi« .....	1092



<b>24.6 Funktionsübersicht</b> .....	1101
<b>24.7 Datentypenübersicht der C-API</b> .....	1105
<b>24.8 Weiterführende Literatur zu Datenbanken</b> .....	1105

## **25 Netzwerkprogrammierung und Cross-Plattform-Entwicklung** 1107

---

<b>25.1 Begriffe zur Netzwerktechnik</b> .....	1107
25.1.1 Ethernet-Frames .....	1108
25.1.2 IP-Adressen .....	1109
25.1.3 Portnummern .....	1111
25.1.4 Host- und Domainnamen .....	1112
25.1.5 Nameserver .....	1112
25.1.6 Das IP-Protokoll .....	1113
25.1.7 TCP und UDP .....	1113
<b>25.2 Was sind Sockets und wie legt man sie in C an?</b> .....	1114
<b>25.3 Headerdateien für die Netzwerkprogrammierung</b> .....	1115
25.3.1 Linux .....	1115
25.3.2 Windows .....	1115
<b>25.4 Das Client-Server-Prinzip</b> .....	1118
<b>25.5 Erstellen einer Client-Anwendung</b> .....	1120
25.5.1 »socket()« – Erzeugen eines Kommunikationsendpunktes .....	1120
25.5.2 »connect()« – ein Client stellt eine Verbindung zum Server her .....	1122
25.5.3 Senden und Empfangen von Daten .....	1127
25.5.4 »close()« und »closesocket()« .....	1130
<b>25.6 Erstellen einer Server-Anwendung</b> .....	1130
25.6.1 »bind()« – Festlegen einer Adresse aus dem Namensraum .....	1130
25.6.2 »listen()« – Warteschlange für eingehende Verbindungen einrichten .....	1132
25.6.3 »accept()« und die Server-Hauptschleife .....	1133
<b>25.7 Ein einfacher TCP-Echo-Server</b> .....	1135
25.7.1 Der Client .....	1136
25.7.2 Der Server .....	1139

<b>25.8 Cross-Plattform-Development</b> .....	1143
25.8.1 Abstraction Layer .....	1143
25.8.2 Headerdatei für Linux .....	1144
25.8.3 Linux-Quellcodedatei .....	1145
25.8.4 Headerdatei für Windows .....	1148
25.8.5 Windows-Quellcodedatei .....	1149
25.8.6 All together – die »main()«-Funktionen .....	1153
25.8.7 Ein UDP-Beispiel .....	1156
25.8.8 Mehrere Clients gleichzeitig behandeln .....	1159
<b>25.9 Weitere Anmerkungen zur Netzwerkprogrammierung und Literaturempfehlungen</b> .....	1166
25.9.1 Das Datenformat .....	1166
25.9.2 Der Puffer .....	1167
25.9.3 Portabilität .....	1168
25.9.4 Von IPv4 nach IPv6 .....	1168
25.9.5 RFC-Dokumente (Request for Comments) .....	1170
25.9.6 Sicherheit .....	1170
25.9.7 Literaturempfehlungen .....	1171
 <b>26 Paralleles Rechnen</b> .....	 1173
<b>26.1 Was ist Multitasking und wie wird es realisiert?</b> .....	1173
<b>26.2 Braucht man spezielle Prozessoren für Multitasking?</b> .....	1174
26.2.1 Single-Prozessor-Systeme .....	1175
26.2.2 Hyperthreading .....	1176
<b>26.3 Braucht man spezielle Multitasking-Betriebssysteme?</b> .....	1177
<b>26.4 Programmiertechniken der Parallelisierung</b> .....	1177
26.4.1 Automatische Parallelisierung .....	1177
26.4.2 Halbautomatische Parallelisierung .....	1178
26.4.3 Echte Parallelisierung .....	1178
<b>26.5 Vom Prozess zum Thread</b> .....	1179
<b>26.6 Mit POSIX-Threads programmieren</b> .....	1182
26.6.1 Ein serielles Beispiel .....	1182
26.6.2 Das Grundgerüst für ein Programm mit mehreren Threads .....	1184
26.6.3 Zusammenfassung .....	1190

**27 Sicheres Programmieren** 1193

---

**27.1 Buffer Overflow (Speicherüberlauf)** ..... 1194

    27.1.1 Was verursacht Buffer Overflows? ..... 1194

**27.2 Warum sind Buffer Overflows kritisch für die Sicherheit?** ..... 1195

    27.2.1 Speicherverwaltung von Programmen ..... 1196

    27.2.2 Der Stack Frame ..... 1197

    27.2.3 Manipulation der Rücksprungadresse ..... 1198

**27.3 Wie man Buffer Overflows vermeidet** ..... 1202

    27.3.1 Unsicheres Einlesen von Eingabe-Streams ..... 1202

    27.3.2 Unsichere Funktionen zur Stringbearbeitung ..... 1203

    27.3.3 Unsichere Funktionen zur Bildschirmausgabe ..... 1203

    27.3.4 Weitere unsichere Funktionen im Überblick ..... 1204

**27.4 Gegenmaßnahmen zum Buffer Overflow, wenn das Programm fertig ist** ..... 1205

    27.4.1 Programme und Tools zum Buffer Overflow ..... 1208

    27.4.2 Ausblick ..... 1209

**27.5 Stack Overflow (Stapelüberlauf)** ..... 1210

**27.6 Was verursacht Stack Overflows?** ..... 1210

**27.7 Warum ist ein Stapelüberlauf kritisch für die Sicherheit?** ..... 1210

**27.8 Wie man Stack Overflows verhindert** ..... 1211

**27.9 Memory Leaks (Speicherlecks)** ..... 1212

**27.10 Bibliotheken und Tools zu Memory Leaks** ..... 1216

    27.10.1 ccmalloc ..... 1216

    27.10.2 dbmalloc ..... 1216

    27.10.3 mpatrol ..... 1216

**27.11 Tipps zu Sicherheitsproblemen** ..... 1217

**28 Wie geht's jetzt weiter?** 1219

---

**28.1 GUI-Programmierung – grafische Oberflächen** ..... 1220

    28.1.1 Low-Level-Grafikprogrammierung ..... 1220

    28.1.2 High-Level-Grafikprogrammierung ..... 1221

    28.1.3 RAD-Tools ..... 1222

    28.1.4 Multimedia-Grafikprogrammierung ..... 1223

---

<b>28.2</b>	<b>Grafikprogrammierung unter Windows</b> .....	1223
28.2.1	In die Windows-Konsole zeichnen .....	1224
28.2.2	Die Methode BitBlt .....	1227
28.2.3	Grafikprogrammierung unter Linux (Beispiel Raspberry Pi) .....	1231
28.2.4	Die Framebuffer-Geräte-datei .....	1231
28.2.5	In den Framebuffer zeichnen .....	1232
28.2.6	Den Bildschirminhalt schnell auffrischen .....	1233
28.2.7	Ein paar Grafikprimitive .....	1236
28.2.8	Weiterführende Literatur zum Thema Grafikprogrammierung .....	1239

---

## Anhang 1241

---

<b>A</b>	<b>Operatoren</b> .....	1241
A.1	Rangfolge der Operatoren .....	1241
A.2	ASCII-Code-Tabelle .....	1243
A.3	Reservierte Schlüsselwörter in C .....	1244
A.4	Standard-Headerdateien der ANSI-C-Bibliothek .....	1245
Index .....		1247