

Inhalt

1	Einführung	25
1.1	Anforderung an den Leser	25
1.2	Das Betriebssystem	26
1.3	Schreibkonventionen	26
1.4	Notationsstil	27
1.5	Weitere Hilfen	28
2	Laufzeitumgebungen	29
2.1	Historisches	29
2.1.1	Von UNIX ...	29
2.1.2	... zu Linux	31
2.2	Distributionen und ihre Unterschiede	33
2.3	Die GNU-Toolchain	34
2.3.1	Das GNU Build System	34
2.3.2	Die GNU Compiler Collection (GCC)	34
2.3.3	GNU Binutils (binutils)	35
2.3.4	GNU Make (gmake)	35
2.3.5	Der GNU Debugger (gdb)	35
2.4	Paketmanagement	36
2.5	Der Compiler GCC – eine kurze Einführung	37
2.5.1	GCC, erhöre uns – der Aufruf	37
2.5.2	Was befehlst du, Meister?	38
2.5.3	Klassifikation der Dateitypen	41
2.6	POSIX, X/OPEN und ANSI C	42
2.6.1	POSIX	42
2.6.2	X/OPEN	44
2.6.3	ANSI C	44
2.6.4	Weitere Standards	45

3	Dynamische Daten in C	47
3.1	Speicher anfordern	47
3.2	Speicher verschieben und löschen	49
3.3	Zeichenketten und -funktionen	49
3.3.1	strdup() und strndup() bzw. strdupa() und strndupa()	49
3.3.2	strcpy(), strncpy(), strcat() und strncat()	50
3.3.3	strchr() und strrchr()	50
3.3.4	strpbrk()	50
3.3.5	strtok() und strtok_r()	51
3.4	Zeichenkodierung	51
3.4.1	Wide Characters	52
3.4.2	UTF-8	52
3.5	Müllsammler, Kanarienvögel und Sicherheit	54
4	E/A-Funktionen	57
4.1	Elementare E/A-Funktionen	57
4.2	Filedeskriptor	58
4.2.1	Verwaltung für offene Deskriptoren	59
4.3	Funktionen, die den Filedeskriptor verwenden	61
4.3.1	Datei öffnen – open()	61
4.3.2	Anlegen einer neuen Datei – creat()	67
4.3.3	Datei schließen – close()	68
4.3.4	Schreiben von Dateien – write()	68
4.3.5	Lesen von Dateien – read()	72
4.3.6	Schreib-/Lesezeiger positionieren – lseek()	74
4.3.7	Duplizieren von Filedeskriptoren – dup() und dup2()	77
4.3.8	Ändern oder Abfragen der Eigenschaften eines Filedeskriptors – fcntl()	79
4.3.9	Record Locking – Sperren von Dateien einrichten	85
4.3.10	Multiplexing E/A – select()	96
4.3.11	Unterschiedliche Operationen – ioctl()	99
4.3.12	Lesen und Schreiben mehrerer Puffer – writev() und readv()	100
4.3.13	Übersicht zu weiteren Funktionen, die den Filedeskriptor verwenden	102
4.4	Standard-E/A-Funktionen	106
4.4.1	Der FILE-Zeiger	106
4.4.2	Öffnen und Schließen von Dateien	107

4.4.3	Formatierte Ausgabe	109
4.4.4	Formatierte Eingabe	110
4.4.5	Binäres Lesen und Schreiben	111
4.4.6	Zeichen- und zeilenweise Ein-/Ausgabe	112
4.4.7	Status der Ein-/Ausgabe überprüfen	113
4.4.8	Stream positionieren	113
4.4.9	Puffer kontrollieren	115
4.4.10	Datei löschen und umbenennen	117
4.4.11	Temporäre Dateien erstellen	117
4.5	Mit Verzeichnissen arbeiten	118
4.5.1	Ein neues Verzeichnis anlegen – mkdir()	119
4.5.2	In ein Verzeichnis wechseln – chdir(), fchdir()	120
4.5.3	Ein leeres Verzeichnis löschen – rmdir()	122
4.5.4	Das Format eines Datei-Eintrags in struct dirent	122
4.5.5	Einen Verzeichnisstream öffnen – opendir()	124
4.5.6	Lesen aus dem DIR-Stream mit opendir() und Schließen des DIR-Streams mit closedir()	126
4.5.7	Positionieren des DIR-Streams	129
4.5.8	Komplettes Verzeichnis einlesen – scandir()	130
4.5.9	Ganze Verzeichnisbäume durchlaufen – nftw()	135
4.6	Fehlerbehandlung	140
4.7	Temporäre Dateien und Verzeichnisse	143
4.8	Ausblick	143
5	Attribute von Dateien und Verzeichnissen	145
5.1	Struktur stat	145
5.1.1	Dateiart und Zugriffsrechte einer Datei erfragen – st_mode	146
5.1.2	User-ID-Bit und Group-ID-Bit – st_uid und st_gid	150
5.1.3	Inode ermitteln – st_ino	151
5.1.4	Linkzähler – st_nlink	151
5.1.5	Größe der Datei – st_size	156
5.1.6	st_atime, st_mtime, st_ctime	158
5.2	Erweiterte Attribute	161
5.2.1	Access Control Lists (Zugriffskontrolllisten)	161
5.2.2	Erweiterte benutzerdefinierte Attribute	167

6	Zugriff auf Systeminformationen	169
<hr/>		
6.1	Das /sys-Dateisystem (sysfs)	169
6.2	Das /proc-Dateisystem (procfs)	171
6.3	Informationen aus dem /proc-Verzeichnis herausziehen	171
6.4	Hardware- bzw. Systeminformationen ermitteln	173
6.4.1	CPU-Informationen – /proc/cpuinfo	174
6.4.2	Geräteinformationen – /proc/devices	175
6.4.3	Speicherauslastung – /proc/meminfo	175
6.4.4	Weitere Hardware-Informationen zusammengefasst	175
6.5	Prozessinformationen	179
6.5.1	/proc/\$pid/cmdline	180
6.5.2	/proc/\$pid/envIRON	180
6.5.3	/proc/self	181
6.5.4	/proc/\$pid/fd/	182
6.5.5	/proc/\$pid/statm	183
6.6	Kernel-Informationen	184
6.6.1	/proc/locks	190
6.6.2	/proc/modules	190
6.7	Filesysteme	191
6.7.1	/proc/mounts	191
6.8	Weiterführendes	192
7	Devices – eine einfache Verbindung zur Hardware	193
<hr/>		
7.1	Die Gerätedateitypen	193
7.2	Die Gerätedateinummern	194
7.3	Historisches	195
7.4	Zugriff auf die Gerätedateien	196
7.5	Gerätenamen	197
7.6	Spezielle Gerätedateien	200
7.7	Gerätedateien in der Praxis einsetzen	201
7.7.1	CD auswerfen und wieder schließen	203
7.7.2	CD-ROM-Fähigkeiten	204

7.7.3	Audio-CD abspielen – komplett und einzelne Tracks – Pause, Fortfahren und Stopp	205
7.7.4	Den aktuellen Status der Audio-CD ermitteln	209
7.7.5	Das komplette Listing	211

8 System- und Benutzerdateien 219

8.1	Die Datei /etc/passwd	219
8.1.1	Die Datei /etc/passwd auswerten	221
8.1.2	getpwuid und getpwnam – einzelne Abfrage von /etc/passwd	221
8.1.3	getpwent, setpwent und endpwent – komplette Abfrage von /etc/passwd	223
8.2	Die Datei /etc/shadow	225
8.2.1	Die Datei /etc/shadow auswerten	227
8.2.2	getspent, setspent und endspent – komplette Abfrage von /etc/shadow	228
8.3	Die Datei /etc/group	231
8.3.1	Die Datei /etc/group auswerten	232
8.3.2	getgrnam und getgrgid – einzelne Einträge aus /etc/group abfragen	233
8.3.3	getgrent, setgrent und endgrent – alle Einträge in /etc/group abfragen	235
8.4	uname – Informationen zum lokalen System erfragen	235
8.5	Das Verzeichnis /etc/skel und der Network Information Service (NIS)	237
8.6	Dateien für Netzwerkinformationen	238
8.7	Folgen für Entwickler	238
8.7.1	Statisches Linken	238
8.7.2	Authentifizierung modernisiert	238

9 Dämonen, Zombies und Prozesse 241

9.1	Was ist ein Prozess?	241
9.2	Prozesskomponente	242
9.2.1	Prozessnummer (PID)	243
9.2.2	Prozessnummer des Vaterprozesses (PPID)	243
9.2.3	Benutzer- und Gruppennummer eines Prozesses (UID, EUID, GID, EGID)	243
9.2.4	Prozessstatus	244
9.2.5	Prozesspriorität	245
9.2.6	Timesharing-Prozesse	246

9.2.7	Echtzeitprozesse	250
9.2.8	Prozessauslagerung	251
9.2.9	Steuerterminal	252
9.3	Prozesse überwachen – ps, top, kpm	252
9.4	Lebenszyklus eines Prozesses	255
9.5	Umgebungsvariablen eines Prozesses	257
9.5.1	Einzelne Umgebungsvariablen abfragen	259
9.5.2	Umgebungsvariable verändern oder hinzufügen – putenv() und setenv()	260
9.5.3	Löschen von Umgebungsvariablen – unsetenv() und clearenv()	263
9.6	Ressourcenlimits eines Prozesses	265
9.6.1	Mehr Sicherheit mit Ressourcenlimits	268
9.6.2	Core-Dateien	269
9.7	Prozesserkennung	269
9.8	Erzeugung von Prozessen – fork()	271
9.8.1	Pufferung	274
9.8.2	Was wird vererbt und was nicht?	279
9.8.3	Einen Prozess mit veränderter Priorität erzeugen	280
9.9	Warten auf einen Prozess	282
9.10	Die exec-Familie	289
9.10.1	execl()	290
9.10.2	execve()	291
9.10.3	execv()	291
9.10.4	execle()	292
9.10.5	execlp()	292
9.10.6	execvp()	293
9.10.7	Kindprozesse mit einem exec-Aufruf überlagern	293
9.11	Kommandoaufrufe aus dem Programm – system()	295
9.12	Dämonen bzw. Hintergrundprozesse	296
9.12.1	Wie ein Prozess zum Dämon wird	297
9.12.2	Dämon, sprich mit uns!	297
9.12.3	Protokollieren von Dämonen – syslog()	298
9.12.4	syslog() in der Praxis	300
9.12.5	Den Dämon, den ich rief	303
9.13	Rund um die Ausführung von Prozessen	307
9.13.1	Einen Dämon beim Booten mit einem init-Skript starten	308
9.13.2	Darf es auch ein bisschen weniger init sein? (Startskript mit systemd)	316
9.13.3	Hintergrundprozesse und Jobkontrolle	317

9.13.4	Prozesse zeitgesteuert ausführen (cron-Jobs)	321
9.14	Zusammenfassung und Ausblick	324
10	Signale	325
<hr/>		
10.1	Grundlagen zu den Signalen	325
10.1.1	Signalmaske	327
10.1.2	Signale und fork()	328
10.1.3	Signale und exec	328
10.1.4	Übersicht über die Signale	328
10.2	Das neue Signalkonzept	331
10.2.1	Wozu ein »neues« Signalkonzept?	332
10.3	Signalmenge initialisieren	332
10.4	Signalmenge hinzufügen oder löschen	333
10.5	Signale einrichten oder erfragen	333
10.5.1	Einen Signalhandler einrichten, der zurückkehrt	338
10.6	Signal an den eigenen Prozess senden – raise()	340
10.7	Signale an andere Prozesse senden – kill()	340
10.8	Zeitschaltuhr einrichten – alarm()	341
10.9	Prozesse stoppen, bis ein Signal eintritt – pause()	342
10.10	Prozesse für eine bestimmte Zeit stoppen – sleep() und usleep()	342
10.11	Signalmaske erfragen oder ändern – sigprocmask()	343
10.12	Prozess während einer Änderung der Signalmaske stoppen – sigsuspend()	344
10.13	Prozesse synchronisieren	344
11	IPC – Interprozesskommunikation	349
<hr/>		
11.1	Unterschiedliche Techniken zur Interprozesskommunikation (IPC) im Überblick	350
11.1.1	(Namenlose) Pipes	350
11.1.2	Benannte Pipes (FIFO-Pipes)	351
11.1.3	Message Queue (Nachrichtenspeicher)	353
11.1.4	Semaphore	353
11.1.5	Shared Memory (gemeinsamer Speicher)	354
11.1.6	STREAMS	354

11.1.7	Sockets	355
11.1.8	Lock Files (Sperrdateien)	356
11.1.9	Dateisperren (Advisory File Locks)	356
11.1.10	Dateisperren (Record Locking)	356
11.2	Neuere Techniken	357
11.2.1	D-Bus und kdbus	357
11.2.2	Dateisystem-Beobachtung	357
11.3	Gründe für IPC	358
11.4	Pipes	359
11.4.1	Eigenschaften von Pipes	359
11.4.2	Pipes einrichten – pipe()	359
11.4.3	Eigenschaften von elementaren E/A-Funktionen bei Pipes	363
11.4.4	Standard-E/A-Funktionen mit pipe	364
11.4.5	Pipes in einen anderen Prozess umleiten	366
11.4.6	Ein Filterprogramm mithilfe einer Pipe erstellen	369
11.4.7	Einrichten einer Pipe zu einem anderen Prozess – popen()	372
11.4.8	Mail versenden mit Pipes und Sendmail	374
11.4.9	Drucken über eine Pipe mit lpr	378
11.4.10	Benannte Pipes – FIFOs	383
11.5	System-V-Interprozesskommunikation	400
11.5.1	Gemeinsamkeiten der System V-UNIX-Mechanismen	400
11.5.2	Ein Objekt einrichten, eine Verbindung herstellen und das Objekt wieder löschen	401
11.5.3	Datenaustausch zwischen nicht verwandten Prozessen	402
11.6	Semaphore	402
11.6.1	Lebenszyklus eines Semaphors	403
11.6.2	Ein Semaphore öffnen oder erstellen – semget()	406
11.6.3	Abfragen, ändern oder löschen der Semaphormenge – semctl()	407
11.6.4	Operationen auf Semaphormengen – semop()	409
11.6.5	Semaphore im Vergleich mit Sperren	411
11.7	Message Queues	411
11.7.1	Eine Message Queue öffnen oder erzeugen – msgget()	412
11.7.2	Nachrichten versenden – msgsnd()	413
11.7.3	Eine Nachricht empfangen – msgrcv()	414
11.7.4	Abfragen, ändern oder löschen einer Message Queue – msgctl()	414
11.8	Shared Memory	424
11.8.1	Ein Shared-Memory-Segment erstellen oder öffnen – shmget()	424
11.8.2	Ein Shared-Memory-Segment abfragen, ändern oder löschen – shmctl()	425
11.8.3	Ein Shared-Memory-Segment anbinden (attach) – shmat()	426

- 11.8.4 Ein Shared-Memory-Segment loslösen – shmdt() 427
- 11.8.5 Client-Server-Beispiel – Shared Memory 427
- 11.9 Das Dateisystem überwachen 435**

12 Threads 439

- 12.1 Unterschiede zwischen Threads und Prozessen 439**
- 12.2 Scheduling und Zustände von Threads 440**
- 12.3 Die grundlegenden Funktionen zur Thread-Programmierung 442**
 - 12.3.1 pthread_create – einen neuen Thread erzeugen 442
 - 12.3.2 pthread_exit – einen Thread beenden 443
 - 12.3.3 pthread_join – auf das Ende eines Threads warten 444
 - 12.3.4 pthread_self – die ID von Threads ermitteln 444
 - 12.3.5 pthread_equal – die ID von zwei Threads vergleichen 450
 - 12.3.6 pthread_detach – einen Thread unabhängig machen 452
- 12.4 Die Attribute von Threads und das Scheduling 453**
- 12.5 Threads synchronisieren 459**
 - 12.5.1 Mutexe 462
 - 12.5.2 Condition-Variablen (Bedingungsvariablen) 471
 - 12.5.3 Semaphore 483
 - 12.5.4 Weitere Synchronisationstechniken im Überblick 486
- 12.6 Threads abbrechen (canceln) 487**
- 12.7 Erzeugen von Thread-spezifischen Daten (TSD-Data) 492**
- 12.8 pthread_once – Codeabschnitt einmal ausführen 495**
- 12.9 Thread-safe (thread-sichere Funktionen) 498**
- 12.10 Threads und Signale 499**
- 12.11 Zusammenfassung und Ausblick 504**

13 Populäre Programmieretechniken unter Linux 505

- 13.1 Reguläre Ausdrücke 505**
 - 13.1.1 Ermittlung der aktuellen Version der SystemRescueCd im Netz 506
 - 13.1.2 Syntax 506
 - 13.1.3 Beispiele 507

13.1.4	Programmieren von regulären Ausdrücken mit pcre	509
13.1.5	In Datenbanken	511
13.2	getopt: Kommandozeilenoptionen auswerten	513
13.3	Capabilities: Wenn root zu mächtig wäre	517
13.3.1	Prozess-Capabilities	518
13.3.2	Datei-Capabilities	522
13.4	Lokalisierung mit gettext	522
13.5	mmap(): Dateien als Speicherbereich	526
13.6	Codedokumentation automatisch erzeugen lassen mit Doxygen	530
14	Netzwerkprogrammierung	533
<hr/>		
14.1	Einführung	533
14.2	Aufbau von Netzwerken	534
14.2.1	ISO/OSI und TCP/IP – die Referenzmodelle	534
14.2.2	Das World Wide Web (Internet)	537
14.3	TCP/IP – Aufbau und Struktur	539
14.3.1	Datenübertragungsschicht	539
14.3.2	Internetschicht	539
14.3.3	Transportschicht (TCP, UDP)	540
14.3.4	Anwendungsschicht	541
14.4	TCP-Sockets	543
14.5	Das Kommunikationsmodell für Sockets	544
14.6	Grundlegende Funktionen zum Zugriff auf die Socket-Schnittstelle	545
14.6.1	Ein Socket anlegen – socket()	545
14.6.2	Verbindungsaufbau – connect()	547
14.6.3	Socket mit einer Adresse verknüpfen – bind()	550
14.6.4	Auf Verbindungen warten – listen() und accept()	550
14.6.5	Senden und Empfangen von Daten (1) – write() und read()	551
14.6.6	Senden und Empfangen von Daten (2) – send() und recv()	552
14.6.7	Verbindung schließen – close()	553
14.7	Aufbau eines Clientprogramms	553
14.7.1	Zusammenfassung Clientanwendung und Quellcode	556
14.8	Aufbau des Serverprogramms	558
14.8.1	Zusammenfassung: Serveranwendung und Quellcode	559

14.9 IP-Adressen konvertieren, manipulieren und extrahieren	563
14.9.1 inet_aton(), inet_pton() und inet_addr()	564
14.9.2 inet_ntoa() und inet_ntop()	565
14.9.3 inet_network()	566
14.9.4 inet_netof()	566
14.9.5 inet_lnaof()	567
14.9.6 inet_makeaddr()	567
14.10 Namen und IP-Adressen umwandeln	570
14.10.1 Nameserver	571
14.10.2 Informationen zum Rechner im Netz – gethostbyname und gethostbyaddr	571
14.10.3 Service-Informationen – getservbyname() und getservbyport()	576
14.11 Der Puffer	580
14.12 Standard-E/A-Funktionen verwenden	581
14.12.1 Pufferung von Standard-E/A-Funktionen	582
14.13 Parallele Server	582
14.14 Synchrones Multiplexing – select()	598
14.15 POSIX-Threads und Netzwerkprogrammierung	620
14.16 Optionen für Sockets setzen bzw. erfragen	625
14.16.1 setsockopt()	625
14.16.2 getsockopt()	626
14.16.3 Socket-Optionen	626
14.17 UDP	630
14.17.1 Clientanwendung	632
14.17.2 Serveranwendung	633
14.17.3 recvfrom() und sendto()	633
14.17.4 bind() verwenden oder weglassen	638
14.18 Unix-Domain-Sockets (IPC)	639
14.18.1 Die Adressstruktur von Unix-Domain-Sockets	639
14.18.2 Lokale Sockets erzeugen – socketpair()	643
14.19 Multicast-Socket	645
14.19.1 Anwendungsgebiete von Multicast-Verbindungen	653
14.20 Nichtblockierende I/O-Sockets	653
14.21 Etwas zu Streams, Raw Socket und TLI und XTI	656
14.21.1 Raw Socket	657
14.21.2 TLI und XTI	657
14.21.3 RPC (Remote Procedure Call)	658

14.22 Netzwerksoftware nach IPv6 portieren	659
14.22.1 Konstanten	659
14.22.2 Strukturen	659
14.22.3 Funktionen	660
14.23 Sicherheit	660
15 Abgesicherte Netzwerkverbindungen	663
<hr/>	
15.1 Grundlagen	663
15.1.1 Schlüsselgenerierung	664
15.1.2 Bibliotheken und Protokollversionen	665
15.1.3 Schlüsselgenerierung in der Praxis	665
15.2 Server	666
15.3 Client	671
15.4 Referenz OpenSSL	675
15.4.1 BIO: Session eröffnen und schließen	675
15.4.2 ERR: Alles über Fehlermeldungen	678
15.4.3 OpenSSL-Konfiguration	679
15.4.4 SSL: Operationen auf SSL-Verbindungen	679
15.4.5 SSL_CTX und andere: Schlüssel und ihre Optionen	682
16 Relationale Datenbanken: MySQL, PostgreSQL und SQLite	687
<hr/>	
16.1 Relationales Datenbankdesign	688
16.2 Datenhaufen und indizierte Listen	690
16.3 SQL-Systeme im Überblick	691
16.4 Vorbereitungen	695
16.4.1 MySQL installieren und starten	696
16.4.2 PostgreSQL installieren und starten	699
16.4.3 SQLite installieren und starten	701
16.5 Beispiele in SQL	701
16.5.1 Tabellen anlegen	701
16.5.2 Schlüsselfelder	702
16.5.3 Indizes	703

16.5.4	Tabellen umbenennen und ändern	704
16.5.5	Daten einfügen, ändern und löschen	708
16.5.6	Daten importieren	711
16.5.7	Daten ausgeben	711
16.5.8	NULL ist 0 oder undefiniert?	713
16.5.9	Unschärfe Suche	714
16.6	MySQL-C-Programmierschnittstelle	714
16.6.1	Verbindung mit dem MySQL-Server aufbauen	716
16.6.2	Aufgetretene Fehler ermitteln – mysql_errno() und mysql_error()	719
16.6.3	Schließt die Verbindung zum Server – mysql_close()	720
16.6.4	Erstes Beispiel	720
16.6.5	Verschiedene Informationen ermitteln	725
16.6.6	Datenbanken, Tabellen und Felder ausgeben (MYSQL_RES)	729
16.6.7	Ergebnismenge zeilenweise bearbeiten (MYSQL_ROW)	731
16.6.8	Ergebnismenge spaltenweise einlesen (und ausgeben) – MYSQL_FIELD	733
16.6.9	Ein Beispiel	738
16.6.10	Ergebnismenge – weitere Funktionen	745
16.6.11	Befehle an den Server – mysql_query() und mysql_real_query()	746
16.6.12	Weitere Funktionen	750
16.6.13	Veraltete Funktionen	756
16.7	Beispiel eines Newssystems mit MySQL	757
16.7.1	Die Headerdatei my CGI.h	758
16.7.2	(Pseudo-)Planung	764
16.7.3	Datenbank und Tabellen anlegen	764
16.7.4	MySQL-Clients mit GUI	786
16.7.5	Randnotiz	786
16.8	Neue SQL-Funktionen für die Shell – MySQL erweitern	787
16.9	MySQL-Funktionen mit der UDF-Schnittstelle entwerfen	788
16.9.1	UDF-Sequenzen	790
16.9.2	UDF_INIT-Struktur	790
16.9.3	UDF_ARGS-Struktur	791
16.9.4	Rückgabewert	793
16.9.5	Benutzerdefinierte Funktionen erstellen	793
16.9.6	Benutzerdefinierte Funktion kompilieren, installieren und ausführen	795
16.10	PostgreSQL – Konfiguration	798
16.10.1	Konfigurationsdateien bei PostgreSQL (postgresql.conf, pg_hba.conf)	798
16.10.2	Crashkurs: PostgreSQL	799
16.10.3	PostgreSQL-C-API – libpg	807
16.10.4	Umgebungsvariablen und Passwortdatei	830
16.10.5	PostgreSQL und Threads	831

16.10.6	Ausblick	832
16.10.7	Funktionsüberblick	833
17	GTK+	839
<hr/>		
17.1	Was ist GTK+?	839
17.1.1	Was sind GDK und Glib?	840
17.1.2	Schnittstellen von GTK+ zu anderen Programmiersprachen	841
17.1.3	GTK+ und GNOME	841
17.1.4	Die GTK+-Versionen 1.2, 2.x und 3.x	841
17.1.5	GTK+ – Aufbau dieses Kapitels	842
17.2	GTK+-Anwendungen übersetzen	843
17.3	Eine Einführung in die Glib-Bibliothek	843
17.3.1	Datentypen	844
17.3.2	Routinen	845
17.3.3	Assertions-Funktionen	847
17.3.4	Speicherverwaltung	849
17.3.5	Stringbearbeitung	852
17.3.6	Selbstverwaltender Stringpuffer	857
17.3.7	Timer	860
17.3.8	Dynamische Arrays	862
17.3.9	Listen, Hashtabellen und binäre Bäume	866
17.3.10	Ausblick zur Glib	867
17.4	Grundlagen der GTK+-Programmierung	867
17.4.1	Die Umgebung initialisieren	868
17.4.2	Widgets erzeugen und gegebenenfalls die Attribute setzen	868
17.4.3	Eine Callback-Funktion einrichten, um Events abzufangen	870
17.4.4	Eine GTK+-Anwendung beenden	873
17.4.5	Die hierarchische Anordnung der Widgets definieren	874
17.4.6	Widgets anzeigen	875
17.4.7	Signale und Events abfangen und bearbeiten – (Events-)Verarbeitungsschleife	876
17.4.8	GTK+ und Umlaute (Zeichenkodierung)	877
17.5	Fenster – GtkWindow	878
17.5.1	Dialogfenster (Dialogboxen)	881
17.5.2	GtkMessageDialog	886
17.6	Anzeige-Elemente	886
17.6.1	Text – GtkLabel	889

17.6.2	Trennlinie – GtkSeparator	892
17.6.3	Grafiken – GtkImage	893
17.6.4	Statusleiste – GtkStatusbar	893
17.6.5	Fortschrittsbalken – GtkProgressBar	894
17.7	Behälter	894
17.7.1	Boxen – GtkBox	894
17.7.2	Aufteilungen, Register und Button-Box	896
17.7.3	Tabellen – GtkTable	903
17.7.4	Ausrichtung – GtkAlignment	907
17.8	Buttons und Toggled-Buttons	908
17.8.1	Buttons allgemein	915
17.8.2	Radio-Buttons (GtkRadioButton)	915
17.8.3	GtkRadioButton, GtkCheckButton und GtkToggleButton	916
17.8.4	Signale für Buttons (GtkButton)	917
17.9	Dateneingabe	917
17.9.1	Textfelder – GtkEntry	924
17.9.2	Schieberegler – GtkScale	925
17.9.3	Zahlenfelder – GtkSpinButton	926
17.9.4	Einstellungen – GtkAdjustment	927
17.9.5	GtkEditable	928
17.10	Menü und Toolbar	929
17.10.1	Menü – GtkItemFactory	934
17.10.2	Toolbar – GtkToolbar	940
17.10.3	Optionsmenü – GtkOptionsMenu	943
17.10.4	Combo-Boxen – GtkCombo	944
17.11	Mehrzeiliger Text	948
17.11.1	Text(editor) – GtkTextView, GtkTextBuffer	956
17.11.2	Scrollendes Fenster – GtkScrolledWindow	960
17.12	Auswählen (Selection)	962
17.12.1	Dateiauswahl – GtkFileSelection	972
17.13	Events	973
17.14	Weitere Widget- und GTK+-Elemente im Überblick	979
17.14.1	Bäume und Listen	980
17.14.2	Lineale	980
17.14.3	Zwischenablage (Clipboard)	980
17.14.4	Drag & Drop	981
17.14.5	Stock Items – Repertoire-Einträge	981
17.14.6	Signale	981
17.14.7	Ressource-Files	981

17.14.8	Widget-Entwicklung	981
17.14.9	GDK	981

18 Werkzeuge für Programmierer 983

18.1	Der Compiler GCC	984
18.1.1	Standardgebrauch des GCC	986
18.1.2	Linken von Programmbibliotheken	986
18.1.3	Dateien, die GCC kennt	988
18.1.4	Ausgabedateien bei jedem einzelnen Schritt der Übersetzung erstellen	988
18.1.5	Noch mehr Optionen	989
18.1.6	Optionen für Warnmeldungen	989
18.1.7	Präprozessor-Optionen	990
18.1.8	Debuggen und Profiling	991
18.1.9	Optimierungsflags	991
18.1.10	Architektur- und Submodell-Flags	992
18.1.11	Statisches vs. dynamisches Linken	993
18.2	Make	994
18.2.1	Make im Zusammenspiel	994
18.2.2	Die Rolle des Makefiles	995
18.2.3	Aufbau des Makefiles	997
18.2.4	Make zur Installation verwenden	1010
18.2.5	Make-Optionen	1011
18.2.6	Ausblick	1011
18.3	Das Makefile portabel machen	1011
18.3.1	Die Autotools	1013
18.4	Bibliotheken erstellen	1019
18.4.1	Allgemeines zu dynamischen Bibliotheken	1024
18.4.2	Dynamisches Nachladen von Bibliotheken	1026
18.5	RPM	1029
18.5.1	Einführung in RPM	1029
18.5.2	Verzeichnisse, die RPM benötigt	1032
18.5.3	Ein eigenes RPM-Paket erstellen	1032
18.5.4	Sources	1032
18.5.5	Die Spec-Datei	1034
18.5.6	Paket erstellen	1037
18.5.7	Das Paket installieren	1040
18.6	Versionskontrollsysteme	1040

- 18.7 Zeitmessung an Programmen** 1042
 - 18.7.1 Einfache Zeitmessung mit TIME – Laufzeit von Prozessen 1042
 - 18.7.2 Profiling mit GPROF – Laufzeit von Funktionen 1043
 - 18.7.3 Analyse mit GCOV 1047
- 18.8 Debuggen mit GDB und DDD** 1050
- 18.9 STRACE – Systemaufrufe verfolgen** 1062
- 18.10 Memory Leaks und unerlaubte Speicherzugriffe** 1064
 - 18.10.1 efence 1065
 - 18.10.2 valgrind 1068
- 18.11 Ausblick** 1072

19 Shell-Programmierung 1075

- 19.1 Was ist eine Shell und was kann sie?** 1075
- 19.2 Was sind Shellskripte und wann ist ihr Einsatz sinnvoll?** 1076
- 19.3 Wann brauche ich keine Shellskripte?** 1077
- 19.4 Welche Schwierigkeiten sind typisch für Shellskripte?** 1078
- 19.5 Verschiedene Shelltypen** 1079
 - 19.5.1 Erweiterungen der Bourne-Shell (sh) 1079
 - 19.5.2 Erweiterung der C-Shell (csh) 1080
 - 19.5.3 Welche Shell sollte man kennen? 1081
- 19.6 Shellskripts ausführen** 1081
 - 19.6.1 Shellskript im Hintergrund ausführen 1082
 - 19.6.2 Die Bedeutung der Subshell 1083
 - 19.6.3 Die ausführende Shell festlegen 1085
 - 19.6.4 Shellskript beenden 1091
- 19.7 Vom Shellskript zum Prozess** 1092
- 19.8 Einen Datenstrom (Stream) umleiten** 1093
 - 19.8.1 Die Standardausgabe umleiten 1093
 - 19.8.2 Die Standardfehlerausgabe umleiten 1094
 - 19.8.3 Die Standardausgabe mit der Standardfehlerausgabe verknüpfen 1095
 - 19.8.4 Die Standardeingabe umleiten 1096
 - 19.8.5 Pipes 1097
 - 19.8.6 Standardausgabe in zwei Richtungen mit tee 1099
 - 19.8.7 Zusammenfassung der verschiedenen Umlenkungen 1100

19.9 Ersatzmuster (Namen-Expansion) zur Suche verwenden	1101
19.9.1 Beliebige Zeichenfolge * (Asterisk)	1101
19.9.2 Beliebiges Zeichen ?	1102
19.9.3 Zeichenbereiche einschränken	1102
19.9.4 Brace Extensions (nur Bash und Korn-Shell)	1104
19.9.5 Tilde-Expansion (nur Bash und Korn-Shell)	1104
19.9.6 Zusammenfassung zu den Ersatzmustern	1104
19.10 Variablen	1105
19.10.1 Zahlen	1109
19.10.2 Zeichenketten	1115
19.10.3 Arrays (nur Bash und Korn-Shell)	1121
19.10.4 Variablen exportieren	1123
19.10.5 Die Umgebungsvariablen (Shellvariablen)	1127
19.10.6 Auto-Variablen der Shell	1128
19.11 Quotings	1130
19.11.1 Single und Double Quotes	1130
19.11.2 Kommandosubstitution (Back Quotes)	1132
19.12 Kommandozeilenargumente	1133
19.12.1 Kommandozeilenparameter \$1 bis \$9	1133
19.12.2 Variable Anzahl von Parametern auswerten	1135
19.12.3 Die Anzahl der Argumente überprüfen	1135
19.12.4 Der Befehl shift	1136
19.12.5 Argumente für die Kommandozeile setzen – set	1137
19.12.6 Kommandozeilenooptionen auswerten	1139
19.13 Kontrollstrukturen	1140
19.13.1 Die if-Anweisung	1141
19.13.2 Die else-Alternative für die if-Verzweigung	1145
19.13.3 Mehrfache Alternative mit elif	1145
19.13.4 Das Kommando test	1146
19.13.5 Dateistatus ermitteln	1151
19.13.6 Ausdrücke logisch verknüpfen	1154
19.13.7 Verzweigungen mit case	1157
19.13.8 Schleifen	1159
19.14 Terminal-Eingabe und -Ausgabe	1166
19.14.1 Ausgabe	1166
19.14.2 Eingabe	1170
19.14.3 Umlenkungen mit exec und File-Deskriptoren erzeugen	1184
19.14.4 Named Pipes	1188
19.14.5 Menü mit select (nur Bash und Korn-Shell)	1189

19.15 Funktionen	1192
19.15.1 Funktionsaufruf	1192
19.15.2 Externe Funktionen verwenden	1193
19.15.3 Parameterübergabe	1193
19.15.4 Rückgabewert aus einer Funktion	1194
19.16 Signale	1197
19.16.1 Signale senden	1197
19.16.2 Signale in einem Shellskript abfangen – trap	1198
19.16.3 Signal-Handler einrichten	1199
19.16.4 Signale ignorieren oder zurücksetzen	1200
19.17 Prozess- und Skriptausführung	1201
19.17.1 Auf Prozesse warten	1201
19.17.2 Hintergrundprozess hervorholen	1201
19.17.3 Jobverwaltung	1202
19.17.4 Explizite Subshell verwenden	1204
19.17.5 Kommunikation zwischen Shellskripten	1205
19.17.6 Skripte zeitgesteuert ausführen	1209
19.18 Ein Shellskript bei der Ausführung	1209
Anhang	1211
<hr/>	
A Sicherheit unter Linux	1213
B Funktionsreferenz	1239
C Linux-Unix-Kommandoreferenz	1321
Index	1405