



Inhaltsverzeichnis

Carsten Vogt

Nebenläufige Programmierung

Ein Arbeitsbuch mit UNIX/Linux und Java

ISBN: 978-3-446-42755-6

Weitere Informationen oder Bestellungen unter

<http://www.hanser.de/978-3-446-42755-6>

sowie im Buchhandel.

Inhalt

1 Einführung	13
2 Basistechniken	19
2.1 Formen der Nebenläufigkeit	20
2.1.1 Hard- und Software – eine Kurzeinführung	20
2.1.1.1 Computer-Hardware	20
2.1.1.2 Computer-Software	21
2.1.2 Nebenläufigkeit in Hardware	22
2.1.3 Nebenläufigkeit in Software	24
2.2 Die Rolle des Betriebssystems	25
2.2.1 Systemarchitekturen	25
2.2.1.1 Aufgaben und Schnittstellen	25
2.2.1.2 Virtualisierung	28
2.2.1.3 Netzdienste und verteilte Systeme	30
2.2.2 Betriebsarten	32
2.2.3 Prozesse und Threads	34
2.2.3.1 Prozesse	34
2.2.3.2 Threads	36
2.2.3.3 Der Lebenszyklus	37
2.2.4 Implementierungsaspekte	39
2.2.4.1 Buchführung	39
2.2.4.2 Dispatching	40
2.2.4.3 Scheduling	40
2.3 Prozesse und Threads in UNIX/Linux	42
2.3.1 Kommandos der Benutzerschnittstelle	42
2.3.2 Grundlegende API-Funktionen für Prozesse	45
2.3.2.1 Die Funktion fork()	46
2.3.2.2 Weitere Funktionen	48
2.3.2.3 Programmbeispiele	50
2.3.3 Grundlegende API-Funktionen für Threads	55
2.3.3.1 Pthreads: pthread_create(), pthread_exit()	55
2.3.3.2 Pthreads: pthread_join(), pthread_cancel()	58
2.3.3.3 vfork() und clone()	59

2.4 Threads in Java.....	60
2.4.1 Die Klasse Thread.....	60
2.4.1.1 run() und start()	61
2.4.1.2 join()	63
2.4.1.3 Weitere Methoden	64
2.4.2 Grundlegende Programmiertechniken	64
2.4.2.1 Zugriff auf gemeinsame Variablen	64
2.4.2.2 Beenden von Threads	65
2.5 Zusammenfassung und Ausblick	67
2A Basistechniken: Aufgaben	69
2A.1 Wissens- und Verständnisfragen.....	69
2A.2 Sprachunabhängige Anwendungsaufgaben	71
2A.3 Programmierung unter UNIX/Linux.....	73
2A.4 Programmierung in Java	75
3 Synchronisation.....	79
3.1 Synchronisationsbedingungen	79
3.1.1 Elementare Bedingungen	79
3.1.1.1 Wechselseitiger Ausschluss	80
3.1.1.2 Reihenfolgebedingung	81
3.1.2 Komplexere Probleme.....	82
3.1.2.1 Erzeuger-Verbraucher-Problem	83
3.1.2.2 Leser-Schreiber-Problem	83
3.1.2.3 Philosophenproblem.....	84
3.2 Einfache Synchronisationsmechanismen	84
3.2.1 Grundlegende Eigenschaften	85
3.2.2 Interruptsperrern.....	85
3.2.3 Spinlocks	86
3.2.4 Signale und Events.....	90
3.3 Synchronisation durch Semaphore.....	91
3.3.1 Arbeitsprinzip von Semaphoren.....	91
3.3.1.1 Datenstrukturen und Operationen	91
3.3.1.2 Semaphoroperationen in Bild und Notation.....	93
3.3.1.3 Varianten und Erweiterungen.....	95
3.3.2 Einsatz bei Standardproblemen.....	96
3.3.2.1 Wechselseitiger Ausschluss	96

3.3.2.2 Reihenfolgebedingung	97
3.3.2.3 Erzeuger-Verbraucher-Problem	98
3.3.2.4 Leser-Schreiber-Problem	99
3.3.2.5 Philosophenproblem	101
3.3.3 Systematische Lösung von Problemen	102
3.3.4 Fehlerquellen	105
3.3.4.1 Deadlocks: Problematik	106
3.3.4.2 Deadlocks: Lösungen	107
3.3.4.3 Missachtung der Atomarität	108
3.3.4.4 Einsatz von sleep()	109
3.3.4.5 Mangelnde Fairness	109
3.4 Synchronisation durch Monitore	110
3.4.1 Grundprinzip von Monitoren	110
3.4.1.1 Definition des Monitorbegriffs	110
3.4.1.2 Beispiel: Einfacher Ringpuffer mit Überschreiben	111
3.4.2 Bedingungsvariablen	112
3.4.2.1 Zweck und Einsatz	112
3.4.2.2 Beispiel: Ringpuffer für Erzeuger/Verbraucher	114
3.4.3 Lösung weiterer Standardprobleme	115
3.4.3.1 Reihenfolgebedingung	115
3.4.3.2 Leser-Schreiber-Problem	116
3.4.3.3 Philosophenproblem	117
3.5 Mechanismen in UNIX/Linux	118
3.5.1 Signale	118
3.5.2 Lock-Dateien	120
3.5.3 Semaphore	121
3.5.3.1 Erzeugen von Semaphorgruppen	121
3.5.3.2 Initialisieren und Löschen	123
3.5.3.3 P- und V-Operationen	125
3.5.3.4 Programmstrukturen und -beispiele	127
3.5.4 Mutexe mit Bedingungsvariablen	132
3.5.4.1 Mutexe	132
3.5.4.2 Bedingungsvariablen	132
3.5.4.3 Beispiel: Erzeuger-Verbraucher mit Ringpuffer	133
3.6 Mechanismen in Java	135
3.6.1 Atomare Operationen	135
3.6.1.1 Basistypen	135
3.6.1.2 Collections	136

3.6.2 Semaphore.....	136
3.6.2.1 Die Klasse Semaphore	136
3.6.2.2 Beispiel: Reihenfolgebeziehung.....	137
3.6.3 Monitore.....	138
3.6.3.1 synchronized.....	138
3.6.3.2 wait() und notify()	140
3.6.3.3 Die Interfaces Lock und Condition	142
3.6.4 Weitere Mechanismen.....	143
3.7 Zusammenfassung und Ausblick	144
3A Synchronisation: Aufgaben.....	147
3A.1 Wissens- und Verständnisfragen.....	147
3A.2 Sprachunabhängige Anwendungsaufgaben	150
3A.3 Programmierung unter UNIX/Linux.....	155
3A.4 Programmierung in Java	158
4 Kommunikation	161
4.1 Grundlegende Begriffe.....	161
4.1.1 Arten der Kommunikation	161
4.1.2 Sender-Empfänger-Beziehungen	163
4.1.2.1 Ein oder mehrere Sender und Empfänger	163
4.1.2.2 Direkte vs. indirekte Kommunikation.....	164
4.1.2.3 Enge vs. lose zeitliche Kopplung	165
4.1.3 Kommunikation in Rechnernetzen.....	166
4.1.3.1 Schnittstellen: Sockets.....	166
4.1.3.2 Protokolle und Protokollstacks.....	167
4.1.3.3 Der Protokollstack des Internets	169
4.2 Techniken in UNIX/Linux	170
4.2.1 Shared Memory.....	171
4.2.1.1 API-Funktionen.....	171
4.2.1.2 Programmbeispiel: Erzeuger-Verbraucher-System.....	173
4.2.2 Pipes.....	174
4.2.2.1 Benannte Pipes	175
4.2.2.2 Unbenannte Pipes.....	176
4.2.3 Message Queues.....	177
4.2.3.1 API-Funktionen: Erzeugen und Löschen	177
4.2.3.2 API-Funktionen: Senden und Empfangen.....	178
4.2.3.3 Programmbeispiel: Erzeuger-Verbraucher-System.....	181

4.2.4 Sockets	182
4.2.4.1 Domains und Typen	182
4.2.4.2 API-Funktionen: Übersicht	183
4.2.4.3 API-Funktionen: Erzeugen und Schließen	185
4.2.4.4 API-Funktionen: Verbinden und Kommunizieren	187
4.2.4.5 Programmbeispiel: Stream-Sockets	188
4.2.4.6 Programmbeispiel: Datagram-Sockets	191
4.3 Techniken in Java	193
4.3.1 Übersicht	193
4.3.2 Piped Streams	194
4.3.3 Sockets	196
4.3.3.1 Stream-Sockets	197
4.3.3.2 Datagram-Sockets	200
4.4 Zusammenfassung und Ausblick	202
4A Kommunikation: Aufgaben	205
4A.1 Wissens- und Verständnisfragen	205
4A.2 Sprachunabhängige Anwendungsaufgaben	208
4A.3 Programmierung unter UNIX/Linux	209
4A.4 Programmierung in Java	213
5 Kooperation	217
5.1 Modelle und Techniken	217
5.1.1 Das Client-Server-Modell	217
5.1.1.1 Grundlegende Struktur	218
5.1.1.2 Zeitliche Abläufe	218
5.1.1.3 Implementierungsaspekte	220
5.1.2 Das Peer-to-Peer-Modell	221
5.1.3 Programmiertechniken	222
5.1.3.1 Prozedurorientierte Kooperation	222
5.1.3.2 Objektorientierte Kooperation	224
5.1.3.3 Webbasierte Kooperation	225
5.2 Techniken in UNIX/Linux	226
5.2.1 Kooperation über Sockets	226
5.2.2 Remote Procedure Call (RPC)	228
5.2.2.1 Komponenten und ihr Zusammenspiel	228
5.2.2.2 Schritte der Programmierung	230

5.3 Techniken in Java.....	234
5.3.1 Remote Method Invocation (RMI)	234
5.3.1.1 Komponenten und ihr Zusammenspiel	234
5.3.1.2 Schritte der Programmierung	235
5.3.2 Dynamische Webseiten.....	238
5.3.2.1 Applets	238
5.3.2.2 Servlets und Java Server Pages	240
5.3.3 Web Services.....	241
5.4 Zusammenfassung.....	243
5A Kooperation: Aufgaben	245
5A.1 Wissens- und Verständnisfragen.....	245
5A.2 Sprachunabhängige Anwendungsaufgaben	247
5A.3 Programmierung unter UNIX/Linux.....	249
5A.4 Programmierung in Java	250
Literatur und Internet.....	253
Index.....	255