Betriebssystemtechnik

Lehrveranstaltungskonzept: Einleitung

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

17. April 2012

Gliederung

1. Systemsoftware
   - Synergie
   - Historie
2. Fallstudie
   - Vorhaben
   - Variabilität
   - Ausführungsstränge
3. Schichtenstruktur
   - Betriebssystemausprägung
   - Variantenvielfalt
4. Zusammenfassung

Hinter der Kulisse: BST in aller Kürze...

- Betriebssystemtechnik
  - Betriebssysteme
    - Federgewichtsprozesse
    - Synchronisation
    - Gastsystem
  - Softwaretechnik
    - Variabilität
    - Programmfamilie
    - Wiederverwendbarkeit

Softwaretechnik ↔ Betriebsysteme

Schichtenstruktur	1968	Dijkstra [5]	↪ THE
?	1969	Ritchie *et al.* [25]	↪ Unix
Botschaft	1970	Hansen [11]	↪ RC 4000
Abstraktion	1971	Liskov [17]	↪ Venus
C	1971	Ritchie *et al.* [26]	↩ Unix
Monitor	1972	Hansen [10]	↩ RC 4000
Geheimnisprinzip	1972	Parnas [20]	
Betriebssystemfamilie	1973	Parnas *et al.* [23]	
Objekt	1974	Wulf *et al.* [28]	↪ HYDRA
abstrakter Datentyp	1974	Liskov *et al.* [18]	↩ Venus
Parallelprogrammierung	1975	Hansen [12]	↩ RC 4000
Transparenz	1975	Parnas *et al.* [24]	
Benutztbeziehung	1976	Parnas [22]	
funktionale Hierarchie	1976	Habermann *et al.* [9]	↪ FAMOS
Programmfamilie	1976	Parnas [21]	

Gliederung

1 Systemsoftware
- Synergie
- Historie

2 Fallstudie
- Vorhaben
- Variabilität
- Ausführungsstränge

3 Schichtenstruktur
- Betriebssystemausprägung
- Variantenvielfalt

4 Zusammenfassung

Fadenbaugruppe (engl. *threads package*)

Operationsprinzip [8]

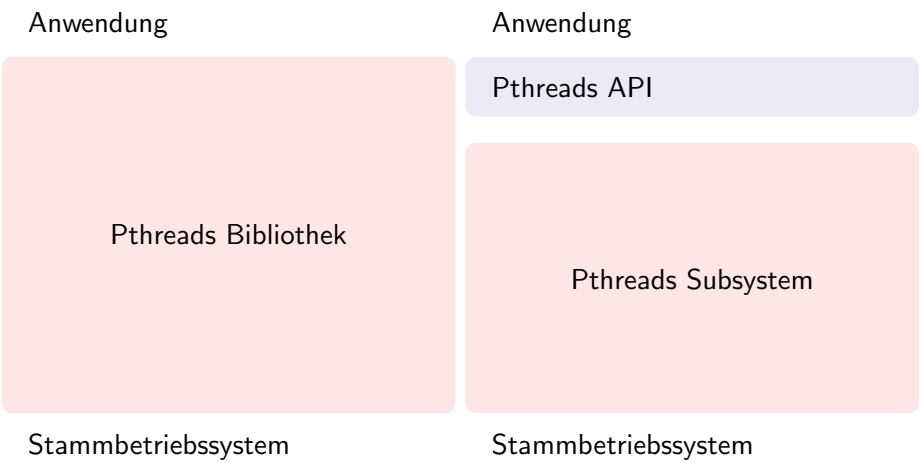
- statisch konfigurierbar und ggf. dynamisch änderbar in Abhängigkeit vom jeweiligen Anwendungsfall auslegen

Informationsstruktur $\mapsto$ Fäden, Fortsetzung
- invariant zu haltender Prozessorstatus bei Ausführungspausen
  - kontextabhängig $\rightsquigarrow$ mehrere Gewichtsklassen
  - min. Befehlszähler (PC), max. kompletter Registersatz
- problemspezifisch ausgeprägte Aktivitätsträger(spezialisierungen)
  - Makro, Prozedur, Methode, Koroutine, Faden
  - gemeinsam bzw. allein benutzter Laufzeitstapel

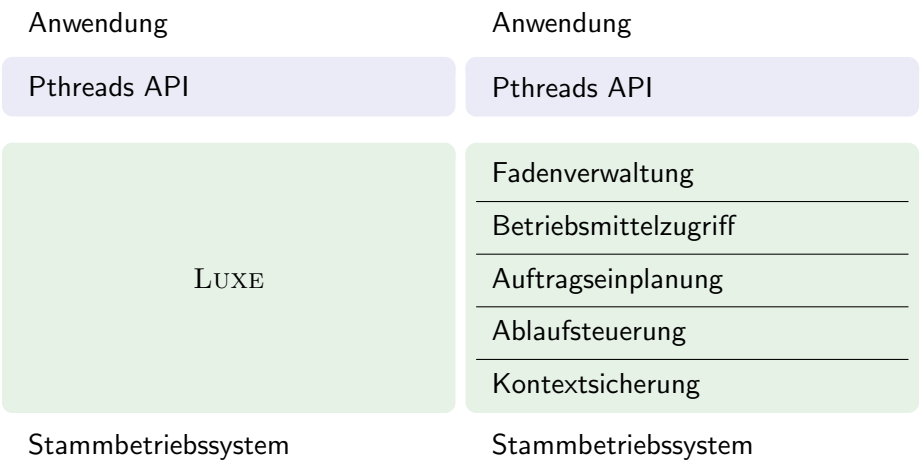
Kontrollstruktur $\mapsto$ Einplanung, Koordinierung
- nicht verdrängend, ggf. aber unterbrechend arbeitend
- verdrängend arbeitend: verzögert, unverzögert

- Fadenfamilie eigentümlicher (nicht-) funktionaler Eigenschaften

Pthreads [16]



Pthreads *de* LUXE



Nomen est omen — Der Name ist ein Zeichen...

Bausatzausstattung folgerichtiger[1] Betriebssystemanbauteile $\models$ LUCSE

logical (dt. folgerichtig)
unit (dt. Anbauteil)
construction-set (dt. Bausatz)
environment (dt. Ausstattung)

(CS $\mapsto$ X) $\rightsquigarrow$ LUXE

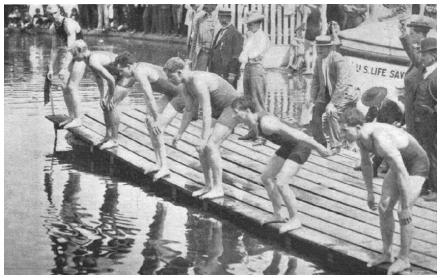
- in bester Tradition mit Multics und Unix: (Multi $\mapsto$ Uni) $\cup$ (cs $\mapsto$ x)

[1] Als Synonym für „durchdacht“.

Konkurrenz — als Triebfeder

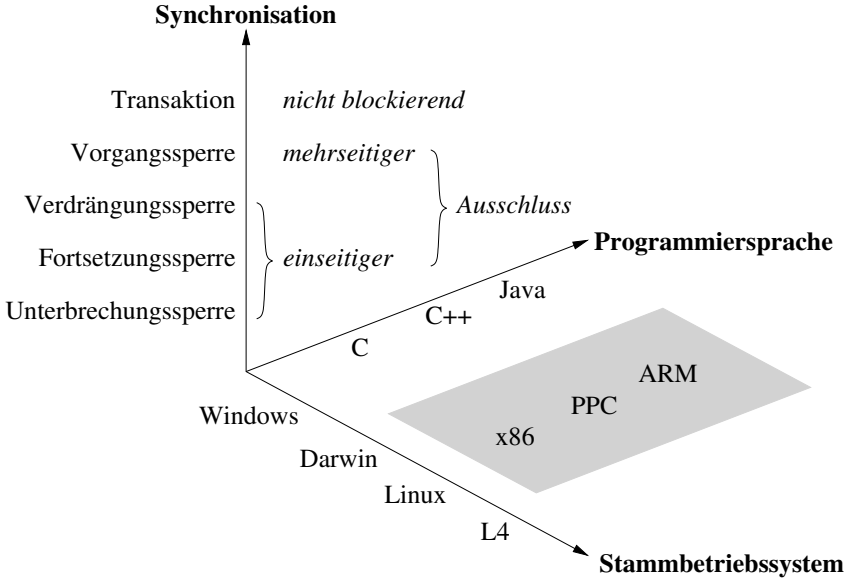
Wettbewerb zwischen den Arbeitsgruppen in Hinblick auf das Ziel, das beste Fädenangebot (engl. *threads package*) zu liefern:

- Betriebsmittelbedarf
  - Laufzeit
  - Speicher
  - ggf. Energie
- Skalierbarkeit
  - n-fädiger Betrieb
  - n-kerniger Betrieb
  - $n = 1, 2, \ldots, N$



- anwendungsfallspezifische, spezialisierte Subsystemvarianten

Dimensionen untersuchter Variabilität



Fadenfamilie $\leftrightarrow$ Betriebssystem

Betriebssystem

Parallelarbeit { nebeneinander / übereinander } verschränkter gleichzeitiger Prozesse

nebeneinander verschränkt $\mapsto$ synchroner Kontrollflusswechsel
- „freiwillige“ Prozessorübergabe (Pausierung/Blockierung)
  - Verdrängung ist der Zwang zur Pausierung !!!
- direkt/indirekt durch den laufenden Programmfaden

übereinander verschränkt $\mapsto$ asynchrone Programmunterbrechung
- „unfreiwillige“ Prozessorübergabe (Unterbrechungsbehandlung)
  - kann Verdrängung zur Folge haben !!!
- durch den vom *Interrupt* betroffenen Programmfaden

Fadenfamilie

- bestimmte Schichten eines Betriebssystems umfassender Komplex

Programmfäden durchziehen Betriebssysteme

Simultanbetrieb geht (schon immer) einher mit der Fähigkeit, *im* Betriebssystem gleichzeitige Prozesse [13, 14] stattfinden zu lassen

- jeder Prozess ist als eigenständiges Kontrollflussexemplar realisiert[2]
- so können im Betriebssystem viele Programmfäden zugleich agieren
  - insb. sich gegenseitig verdrängen, auf Betriebsmittelzuteilung warten

Mehrfädigkeit eines Betriebssystems muss deshalb aber noch längst nicht an der Systemaufrufschnittstelle sichtbar sein

- erst Thoth [3, 4] brachte Fäden in die Anwendungsprogramme [7]
- Standardbetriebssysteme haben (zu) lange darauf warten lassen

- Fadenkonzepte gibt es heute inner-/außerhalb von Betriebssystemen

[2] Ausnahmen wie z.B. TinyOS [15], Contiki [6], BCC-Varianten von OSEK [19] oder Betriebssysteme mit stapelbasierter Einplanung [1, 2] bestätigen die Regel.

Gliederung

1 Systemsoftware
- Synergie
- Historie

2 Fallstudie
- Vorhaben
- Variabilität
- Ausführungsstränge

3 Schichtenstruktur
- Betriebssystemausprägung
- Variantenvielfalt

4 Zusammenfassung

Programmfäden und Betriebssysteme: Orthogonalität

Stammbetriebssystem $\mapsto$ bodenständig (engl. *native*)

- Programm der Ebene $_2$, auf Befehlssatzebene aufsetzend
  - wobei Ebene $_2$ eine reale oder virtuelle Maschine darstellt
- implementiert die Maschinenprogrammebene, Ebene $_3$
  - virtuelle Maschine, interpretiert Maschinenprogramme partiell [27]
- kann Fäden auf Kernebene (engl. *kernel-level threads*) bieten

Gastbetriebssystem $\mapsto$ nicht bodenständig

- Programm der Ebene $_3$, auf Maschinenprogrammebene aufsetzend
  - stellt damit selbst ein (gewöhnliches) Maschinenprogramm dar
- tritt als Bibliotheksbetriebssystem in Erscheinung
  - virtuelle Maschine, privilegierte Operationen mittels Systemaufrufe
- bietet ggf. Fäden auf Benutzerebene (engl. *user-level threads*)

Programmfaden $\mapsto$ Konzept einer virtuellen Maschine [27]

- existiert auf Benutzerebene (eines BS) ebenso wie auf Kernebene

Fallbeispiel Stammbetriebssystem

Schicht	Funktion	Konzepte
12	Programmverwaltung	Text, Daten, Überlagerung
11	Dateiverwaltung	Dateisystem; Verzeichnis, Verknüpfung
10	Prozessverwaltung	Aktivitätsträger, Kontext, Stapel
9	Adressraumverwaltung	Arbeitsspeicher, Segment, Seite
8	Informationsaustausch	Paket, Nachricht, Kanal, Portal
7	Geräteprogrammierung	Kern; Signal, Zeichen, Block, Datenstrom
6	Elementplatzierung	Hauptspeicher; Seitenrahmen, Segment/Fragment
5	Zugriffskontrolle	Subjekt, Objekt, Domäne, Befähigung
4	Betriebsmittelzugriff	Verdrängungs-/Vorgangssperre
3	Auftragseinplanung	Ereignis, Priorität, Zeitscheibe, Energie
2	Ablaufsteuerung	Unterbrechungs-/Fortsetzungssperre, Transaktion
1	Kontextsicherung	Koroutine, Unterbrechung, Fortsetzung, Region
0	Stammprozessorabstraktion	**ADT**, Stammsystem
−1	Peripherie	MMU, (A)PIC, DMA, UART, ATA, SCSI, USB, ...
−2	Zentraleinheit	ARM, AVR, PowerPC, SPARC, x86, ...

Fallbeispiel Gastbetriebssystem

Schicht	Funktion	Konzepte
12	Programmverwaltung	Text, Daten, Überlagerung
11	Dateiverwaltung	Dateisystem; Verzeichnis, Verknüpfung
10	Prozessverwaltung	Aktivitätsträger, Kontext, Stapel
9	Adressraumverwaltung	Arbeitsspeicher, Segment, Seite
8	Informationsaustausch	Paket, Nachricht, Kanal, Portal
7	Geräteprogrammierung	Kern; Signal, Zeichen, Block, Datenstrom
6	Elementplatzierung	Hauptspeicher; Seitenrahmen, Segment/Fragment
5	Zugriffskontrolle	Subjekt, Objekt, Domäne, Befähigung
4	Betriebsmittelzugriff	Verdrängungs-/Vorgangssperre
3	Auftragseinplanung	Ereignis, Priorität, Zeitscheibe, Energie
2	Ablaufsteuerung	Unterbrechungs-/Fortsetzungssperre, Transaktion
1	Kontextsicherung	Koroutine, Unterbrechung, Fortsetzung, Region
0	Stammprozessorabstraktion	**ADT**, Gastsystem
−1	Betriebssystem	Darwin, L4, Linux, Solaris, Windows (Cygwin), ...
−2	Zentraleinheit	ARM, AVR, PowerPC, SPARC, x86, ...

Fallbeispiel Gastbetriebssystem *de* LUXE

Schicht	Funktion	Konzepte
12	*leer*	*keine*
11	*leer*	*keine*
10	Prozessverwaltung	Aktivitätsträger, Kontext, Stapel
9	*leer*	*keine*
8	Informationsaustausch	Paket, Nachricht; `socket(2)`
7	Geräteprogrammierung	Kern, `sched_set_affinity(2)`; `signal(3)`
6	Elementplatzierung	Hauptspeicher, `sbrk(2)`, `malloc(3)`
5	*leer*	*keine*
4	Betriebsmittelzugriff	Verdrängungs-/Vorgangssperre
3	Auftragseinplanung	Ereignis, Priorität, Zeitscheibe
2	Ablaufsteuerung	Unterbrechungs-/Fortsetzungssperre, Transaktion
1	Kontextsicherung	Koroutine, Unterbrechung, Fortsetzung, Region
0	Stammprozessorabstraktion	**ADT**, Gastsystem
−1	Betriebssystem	Darwin, L4, Linux, Windows (Cygwin)
−2	Zentraleinheit	PowerPC, x86

Schicht (engl. *layer*, *tier*) als logisches Gebilde

Anwendungen legen die in einem Betriebssystem wesentlich/fallweise realisierten funktionalen und nichtfunktionalen Eigenschaften fest

- wirkliches Vorhandensein einer Schicht ist niemals Dogma
- Schichtenstrukturen definieren sich über eine Benutztbeziehung [22]
- nicht benutzte (gebrauchte) Funktionen sind nicht real vorhanden

Universalbetriebssysteme sind typisch für eine komplett umgesetzte Schichtenstruktur — so sie überhaupt schichtenstrukturiert sind

- wenngleich auch mit Optionen in den Diensten einzelner Schichten

Spezialbetriebssysteme sind typisch für eine nur teilweise umgesetzte Schichtenstruktur — sofern eine solche definiert ist

- ausgewählte Schichten sind komplett „leer“ d.h. nicht realisiert

Beachte

- der Systementwurf ist komplett, die Umsetzung nicht unbedingt

Fadenfamilie ↔ Betriebssystem *de* LUXE

Schicht... Ausführung von Programmfäden	*wesentlich*
10 Prozessverwaltung	
3 Auftragseinplanung	
1 Kontextsicherung	

Schicht... Koordinierung gleichzeitiger Prozesse	*fallweise*
4 Betriebsmittelzugriff	
2 Ablaufsteuerung	

Schicht... Anschluss an die Außenwelt	*wesentlich*
7 Geräteprogrammierung	
0 Stammprozessorabstraktion	

Fadenfamilie ↔ Betriebssystem *de* LUXE (Forts.)

Schicht... Stammsystemspezifisch *bodenständig*

10 Prozessverwaltung

4 Betriebsmittelzugriff

3 Auftragseinplanung

2 Ablaufsteuerung
- Unterbrechungssperre: prozessorabhängig
- Transaktion: CPU abhängig

1 Kontextsicherung
- Koroutine/Unterbrechung: prozessorabhängig
- Region: betriebsartenabhängig

Schicht... Gastsystemspezifisch *nicht bodenständig*

7 Geräteprogrammierung

0 Stammprozessorabstraktion

Fadenfamilie ↔ Variabilität

Funktional *gleiche*, nicht-funktional *ungleiche* Implementierungen
- Koroutine: Varianten zur Stapelanbindung
  - von einer Koroutine allein oder mehreren gemeinsam benutzter Stapel
- Programmfaden: Varianten der Zustandssicherung
  - alle, nur die nicht-flüchtigen oder gar keine Arbeitsregister sichern
- Sperre: Varianten in Bezug auf die Zielbereiche
  - Unterbrechungen, Fortsetzungen, Verdrängungen, Vorgänge sperren
- kritischer Abschnitt: Varianten von Schutzverfahren
  - Ereignis sperrende/zulassende Synchronisation gleichzeitiger Prozesse

Querschneidende Belange
- gemeinsam/allein von Koroutinen bzw. Fäden benutzter Stapel
- verdrängende und verzögerte/unverzögerte Fadeneinplanung
- Wettstreitigkeiten gegenüber tolerante/intolerate Koordinierung

Gliederung

1 Systemsoftware
- Synergie
- Historie

2 Fallstudie
- Vorhaben
- Variabilität
- Ausführungsstränge

3 Schichtenstruktur
- Betriebssystemausprägung
- Variantenvielfalt

4 Zusammenfassung

Resümee

Einordnung $\mapsto$ *logical unit construction-set environment* $\models$ LUXE
- Untersuchung von Variabiliät in der Systemsoftware
- Synchronisation, Stammbetriebssystem, ggf. Programmiersprache

Schichtenstruktur $\mapsto$ Fadenfamilie im Betriebssystemkontext
- Stammbetriebssystem $\models$ Gastbetriebssystem $\supset$ LUXE
- Schichten 1–4, 10: Fädenangebot im Stamm-/Gastbetriebssystem

Variantenvielfalt $\mapsto$ Betriebsart bzw. Architektur eines Rechensystems
- fallweise, Reaktion darauf ist von querschneidendem Belang

  Verdrängung $\Rightarrow$ Koordinierung durch Schicht$_4$ $\supset$ Schicht$_2$

  Unterbrechung $\Rightarrow$ Koordinierung durch Schicht$_2$
- wesentlich, wenngleich anwendungsfallabhängige Funktionen
  - Prozessverwaltung, Auftragseinplanung, Kontextsicherung
  - Geräteprogrammierung, Stammprozessorabstraktion

Literaturverzeichnis

[1] Baker, T. P.:
A Stack-Based Resource Allocation Policy for Realtime Processes.
In: *Proceedings of the 11th IEEE Real-Time Systems Symposium (RTSS 1990)*.
Washington, DC, USA : IEEE Computer Society, 1990. –
ISBN 0–8186–2112–5, S. 191–200

[2] Baker, T. P.:
Stack-Based Scheduling of Realtime Processes.
In: *Real-Time Systems* 3 (1991), März, Nr. 1, S. 67–99

[3] Cheriton, D. R.:
Multi-Process Structuring and the Thoth Operating System.
Ontario, Canada, University of Waterloo, Diss., 1978

[4] Cheriton, D. R. ; Malcolm, M. A. ; Melen, L. S.:
Thoth, a Portable Real-Time Operating System.
In: *Communications of the ACM* 22 (1979), Febr., Nr. 2, S. 105–115

[5] Dijkstra, E. W.:
The Structure of the THE-Multiprogramming System.
In: *Communications of the ACM* 11 (1968), Mai, Nr. 5, S. 341–346

Literaturverzeichnis (Forts.)

[6] Dunkels, A. ; Grönvall, B. ; Voigt, T. :
Contiki — A Lightweight and Flexible Operating System for Tiny Networked Sensors.
In: Estrin, D. (Hrsg.) ; Bulusu, N. (Hrsg.) ; Jha, S. (Hrsg.): *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (EmNetS-I)*.
Washington, DC, USA : IEEE Computer Society, 2004 (29th Annual IEEE International Conference on Local Computer Networks (LCN'04)). –
ISBN 0–7695–2260–2, S. 455–462

[7] Gentleman, W. M.:
Message Passing Between Sequential Processes: The Reply Primitive and the Administrator Concept.
In: *Software—Practice and Experience* 11 (1981), Nr. 5, S. 435–466

[8] Giloi, W. K.:
Rechnerarchitektur.
Berlin Heidelberg : Springer-Verlag, 1993. –
ISBN 3–540–56355–5

[9] Habermann, A. N. ; Flon, L. ; Cooprider, L. W.:
Modularization and Hierarchy in a Family of Operating Systems.
In: *Communications of the ACM* 19 (1976), Mai, Nr. 5, S. 266–272

Literaturverzeichnis (Forts.)

[10] Hansen, P. B.:
Structured Multiprogramming.
In: *Communications of the ACM* 15 (1972), Jul., Nr. 7, S. 574–578

[11] Hansen, P. B.:
The Nucleus of a Multiprogramming System.
In: *Communications of the ACM* 13 (1970), Apr., Nr. 4, S. 238–241/250

[12] Hansen, P. B.:
The Programming Language Concurrent Pascal.
In: *IEEE Transactions on Software Engineering* 1 (1975), Jun., Nr. 2, S. 199–207

[13] Hansen, P. B.:
The Solo Operating System: A Concurrent Pascal Program.
In: *Software—Practice and Experience* 6 (1976), Apr.–Jun., Nr. 2, S. 141–149

[14] Hansen, P. B.:
The Architecture of Concurrent Programs.
Englewood Cliffs, New Jersey : Prentice-Hall, Inc., 1977. –
ISBN 0–13–044628–9

Literaturverzeichnis (Forts.)

[15] Hill, J. L. ; Szewczyk, R. ; Woo, A. ; Hollar, S. ; Culler, D. E. ; Pister, K. S. J.:
System Architecture Directions for Networked Sensors.
In: *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)* Bd. 34.
New York, NY, USA : ACM Press, 2000 (ACM SIGOPS Operating Systems Review 5), S. 93–104

[16] Institute of Electrical and Electronics Engineers, Inc.:
IEEE Std 1003.1c-1995 Thread Extensions.
New York, NY, USA, 1995

[17] Liskov, B. J. H.:
The Design of the Venus Operating System.
In: *Communications of the ACM* 15 (1972), März, Nr. 3, S. 144–149

[18] Liskov, B. J. H. ; Zilles, S. N.:
Programming with Abstract Data Types.
In: Leavenworth, B. (Hrsg.): *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages* Bd. 9.
New York, NY, USA : ACM, Apr. 1974 (ACM SIGPLAN Notices 4), S. 50–59

Literaturverzeichnis (Forts.)

[19] OSEK/VDX Organisation:
OSEK VDX Portal.
http://www.osek-vdx.org/,

[20] Parnas, D. L.:
On the Criteria to be used in Decomposing Systems into Modules.
In: *Communications of the ACM* 15 (1972), Dez., Nr. 12, S. 1053–1058

[21] Parnas, D. L.:
On the Design and Development of Program Families.
In: *IEEE Transactions on Software Engineering* SE-2 (1976), März, Nr. 1, S. 1–9

[22] Parnas, D. L.:
Some Hypothesis About the "Uses" Hierarchy for Operating Systems / TH Darmstadt, Fachbereich Informatik.
1976 (BSI 76/1). –
Forschungsbericht

[23] Parnas, D. L. ; Price, W. R.:
The Design of the Virtual Memory Aspects of a Virtual Machine.
In: Gagliardi, U. O. (Hrsg.) ; Bolliet, L. (Hrsg.) ; Goldberg, R. P. (Hrsg.):
Proceedings of the Workshop on Virtual Computer Systems.
New York, NY, USA : ACM, 1973, S. 184–190

Literaturverzeichnis (Forts.)

[24] Parnas, D. L. ; Siewiorek, D. P.:
Use of the Concept of Transparency in the Design of Hierarchically Structured Systems.
In: *Communications of the ACM* 18 (1975), Jul., Nr. 7, S. 401–408

[25] Ritchie, D. M. ; Thompson, K. :
The UNIX Timesharing System.
In: *Communications of the ACM* 17 (1974), Jul., Nr. 7, S. 365–374

[26] Ritchie, D. M. ; Thompson, K. ; Johnson, S. C. ; Lesk, M. E.:
The C Programming Language.
In: *Bell System Technical Journal* 57 (1978), Jul./Aug., Nr. 6

[27] Tanenbaum, A. S.:
Multilevel Machines.
In: *Structured Computer Organization*.
Prentice-Hall, Inc., 1979. –
ISBN 0–130–95990–1, Kapitel 7, S. 344–386

[28] Wulf, W. A. ; Cohen, E. S. ; Corwin, W. M. ; Jones, A. K. ; Levin, R. ; Pierson, C. ; Pollack, F. J.:
HYDRA: The Kernel of a Multiprocessor Operating System.
In: *Communications of the ACM* 17 (1974), Jun., Nr. 6, S. 337–345