Systemprogrammierung

Betriebsarten: Dialogverarbeitung

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

2. Juli 2014

Gliederung

Gleichzeitiger, interaktiver Zugang für mehrere Benutzer

Benutzereingaben und deren Verarbeitung wechseln sich anhaltend ab

- E/A-intensive Anwendungsprogramme interagieren mit Benutzern
- Zugang über Dialogstationen (engl. *interactive terminals*)
  - Datensichtgerät und Tastatur (seit 1950er Jahren, Whirlwind/SAGE)
  - später die *Maus* (Engelbart/English, SRI, 1963/64; vorgestellt 1968)

Einplanung bevorzugt interaktive (E/A-intensive) Prozesse

- Beendigung von Ein-/Ausgabe führt zur „prompten“ Neueinplanung
  - im Falle von E/A-Operationen, die sich blockierend auswirken
- Benutzer erfahren eine schnelle Reaktion insb. auf Eingaben
  - sofern auch die Einlastung von Prozessen „prompt“ geschieht

Dialogbetrieb (engl. *conversational mode*)

- Zusatz zum Stapelbetrieb, Monopolisierung der CPU, Sicherheit

Dialogorientiertes Monitorsystem

„Graue Eminenz“ gleich im Hintergrund die Fäden (engl. *threads*) ziehen

Programme „im Vordergrund“ starten und „im Hintergrund“ verarbeiten

- interaktiv Aufträge annehmen, ausführen und dabei überwachen
  - d.h. starten, stoppen, fortführen und ggf. abbrechen
- zur selben Zeit sind mehrere Programme nebenläufig/parallel aktiv
- im Ergebnis werden mehrere Aufgaben „gleichzeitig“ bearbeitet

Mischbetrieb

- E/A-intensive Programme im Vordergrund ausführen
- CPU-intensive Programme im Hintergrund ausführen

Vordergrund	Hintergrund
Echtzeitbetrieb	Dialogbetrieb
Dialogbetrieb	Stapelbetrieb

Problem

- Hauptspeicher(größe)

Dialogstationen können eigene Dialogprozesse absetzen

Zeitscheibe (engl. *time slice*) als „Betriebsmittel“

- jeder Prozess bekommt die CPU nur für eine Zeitspanne zugeteilt
  time sharing (engl.) Prozesse berechtigen, die CPU für einige Zeit zur Abarbeitung der ihnen zugeordneten Programme zu nutzen
- endet die Zeitspanne, werden die Prozesse neu eingeplant
  - ggf. Verdrängung (engl. *preemption*) des laufenden Prozesses
  - d.h. ihm wird die CPU zu Gunsten eines anderen Prozesses entzogen

Zeitgeber (engl. *timer*) sorgen für **zyklische Programmunterbrechungen**

- das Intervall ist durch die Zeitscheibenlänge vorgegeben
- Monopolisierung der CPU ist nicht mehr möglich: **CPU-Schutz**

Problem

- Arbeitsspeicher, Einplanung, Einlastung, Ein-/Ausgabe, Sicherheit

Bahnbrecher und Wegbereiter

CTSS (*Compatible Time-Sharing System* [1], MIT, 1961)

Time-sharing introduced the engineering constraint that the interactive needs of users [were] just as important as the efficiency of the equipment. (Fernando J. Corbató)

Time-sharing was a misnomer. While it did allow the sharing of a central computer, its success derives from the ability to share other resources: data, programs, concepts. (Louis Pouzin)

ITS (*Incompatible Time-sharing System* [5], MIT, 1969)

- Pionierarbeit zur Ein-/Ausgabe und Prozessverwaltung:
  - geräteunabhängige Ausgabe auf Grafikbildschirme, virtuelle Geräte
  - netzwerktransparenter Dateizugriff (über ARPANET [19])
  - Prozesshierarchien, Kontrolle untergeordneter Prozesse
- „Seitenhieb“ auf CTSS und Multics, wegen der eingeschlagenen Richtung

Dialogstationen teilen sich einen Dialogprozess

Bedienung mehrerer Benutzer durch ein Programm (bzw. einen Prozess)

- gleichartige, bekannte Vorgänge an vielen Daten-/Dialogstationen
- Wiederverwendung derselben residenten Dialogprozessinkarnation

Endbenutzerdienst mit festem, definiertem Funktionsangebot

- Kassen, Bankschalter, Auskunft-/Buchungssysteme, ...
- **Transaktionssysteme**

Klient/Anbieter System (engl. *client/server system*)

- sehr viele Dienstnehmer (engl. *service user*)
- einen/wenige Dienstgeber (engl. *service provider*)

Problem

- Antwortverhalten (weiche/feste Echtzeit), Durchsatz

Gliederung

SMP: Abk. engl. *symmetric multiprocessing*

Symmetrisch aufgebauter Multiprozessor

Zwei oder mehr gleiche (bzw. identische) Prozessoren, die über ein gemeinsames Verbindungssystem (meist ein Bus, aber z.B. auch die Kreuzschiene (engl. crossbar)) gekoppelt sind.

- jeder Prozessor hat gleichberechtigten Zugriff auf Hauptspeicher (engl. *shared-memory acccess*) und E/A-Geräte
- der Zugriff auf alle Speicherzellen ist für alle Prozessoren gleichförmig (engl. *uniform memory access*, UMA)
- die Prozessoren bilden ein **homogenes System** und werden von demselben Betriebssystem verwaltet

Problem

- Koordination, Skalierbarkeit

SMP: Abk. engl. *shared-memory processor*

Speichergekoppelter Multiprozessor

Ein Parellelrechnersystem, bei dem sich alle Prozessoren denselben Arbeitsspeicher teilen, nicht jedoch gleichberechtigten Zugriff darauf haben müssen — sie können zusammen ein **heterogenes System** *bilden.*

asymmetrischer SMP ∼ *asymmetric multiprocessing*

- hardwarebedingter asymmetrischer Multiprozessorbetrieb
- Programme/Betriebssystem sind ggf. prozessorgebunden

symmetrischer SMP ∼ *symmetric multiprocessing*

- Anwendungen geben die gewünschte Art des Parallelbetriebs vor
- das Betriebssystem legt die Multiprozessorbetriebsart fest

Problem

- Koordination, Skalierbarkeit, Anpassbarkeit

Parallelverarbeitung (engl. *parallel processing*)

Eines oder mehrerer Programme...

N Prozessoren können...

- *N* verschiedene Programme
- *N* Prozesse verschiedener Programme
- *N* Prozesse desselben Programms

In einem Programm können mehrere **Fäden** (engl. *threads*) gleichzeitig aktiv sein.

...parallel ausführen und jeder der *N* Prozessoren kann...

- *M* verschiedene Programme
- *M* Prozesse verschiedener Programme
- *M* Fäden desselben Programms

...pseudo-parallel (nebenläufig) im Multiplexbetrieb ausführen

Sicherheit (engl. *security*)

Schutz (engl. *protection*) vor nicht authorisierten Zugriffen

Adressraumisolation: Schutz ⇝ Eingrenzung/Segmentierung

- Zugriffsfehler führen zum Abbruch der Programmausführung
  - die CPU (bzw. MMU) fängt, das Betriebssystem bricht ab
- i.A. keine selektive Zugriffskontrolle möglich und sehr grobkörnig

Prozessen eine Befähigung (engl. *capability* [3]) zum Zugriff erteilen:

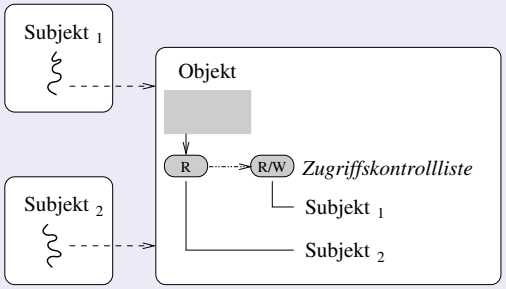
Verschiedenen **Subjekten** *unterschiedliche Zugriffsrechte (ausführen, lesen, schreiben, ändern) auf dasselbe* **Objekt** *(Datum, Datei, Gerät, Prozedur, Prozess) einräumen. [25, 6]*

Alternative: Zugriffskontrollliste (engl. *access control list*, ACL [21])

Problem

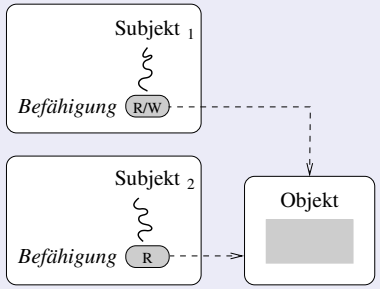
- verdeckter Kanal (engl. *covered channel*)

Zugriffskontrollliste



- Rechtevergabe einfach (lokal)
- Rechterücknahme: einfach (lokal)
- Rechteüberprüfung: aufwändig (Suche)
- dito Subjektrechtebestimmung (entfernt)
- Objektsicht-Rechtebestimmung: einfach
- Kontrollinformation: zentral gespeichert

Befähigungen



- aufwändig (entfernt)
- aufwändig (entfernt)
- einfach (lokal)
- einfach (Zugriff)
- aufwändig (Sammelruf)
- dezentral gespeichert

Methodologie: Zugriffsmatrix (engl. *access matrix* [12, 9])

Einträge spezifizieren individuelle Zugriffsrechte der Subjekte auf Objekte:

Subjekte	Objekte		
	Cyan	Grau	Blau
1	R/X	R/W	–
2	–	R	–

R Abk. engl. *read*
W Abk. engl. *write*
X Abk. engl. *execute*

Implementierungsoptionen

Totalsicht ☹
- in der dem Modell entsprechenden Tabellenform
- ineffizient, wegen der i.d.R. dünn besetzten Matrix

Objektsicht ☺
- in Form einer objektbezogenen **Zugriffskontrollliste**
- spaltenweise Speicherung der Zugriffsmatrix

Subjektsicht ☺
- in Form subjektbezogener **Befähigungen**
- zeilenweise Speicherung der Zugriffsmatrix

Ringschutz (engl. *ring-protected paged segmentation*)

Multics [18, 26], Maßstab in Bezug auf Adressraum-/Speicherverwaltung:

1. *jede* im System gespeicherte abrufbare Information ist direkt von einem Prozessor adressierbar und jeder Berechnung referenzierbar
2. *jede* Referenzierung unterliegt einer durch **Hardwareschutzringe** implementierten mehrstufigen Zugriffskontrolle [22, 20]

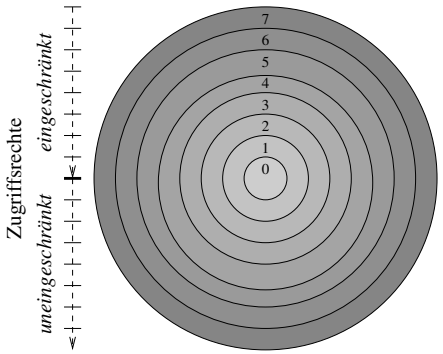
„*trap on use*“, ausnahmebedingtes dynamisches Binden/Laden:

- Textsegmente werden bei Bedarf automatisch nachgeladen
- mögliche synchrone Programmunterbrechung beim Prozeduraufruf

Problem

- hochkomplexes System (Hardware/Software)

Schutzringe: Multics



Verwendung der Schutzringe:

0–3 Betriebssystem
- 0–1 Hauptsteuerprogramm
- 2–3 Dienstprogramme

4–7 Anwendungssystem
- 4–5 Benutzerprogramme
- 6–7 Subsysteme

Ringwechsel, Zugriffe

- kontrolliert durch die Hardware
- ggf. Teilinterpretation

Problem

- Schichtenstruktur, Ringzuordnung: **funktionale Hierarchie** [10]

Arbeitsspeicher und der Grad an Mehrprogrammbetrieb

Überlagerung durch programmiertes dynamisches Laden hat seine Grenze

Anzahl × Größe arbeitsspeicherresidenter Text-/Datenbereiche begrenzt die Anzahl der gleichzeitig zur Ausführung vorgehaltenen Programme

- variabler Wert, abhängig von Struktur/Organisation der Programme
- eine Variable auch in Bezug auf die Fähigkeiten der Programmierer

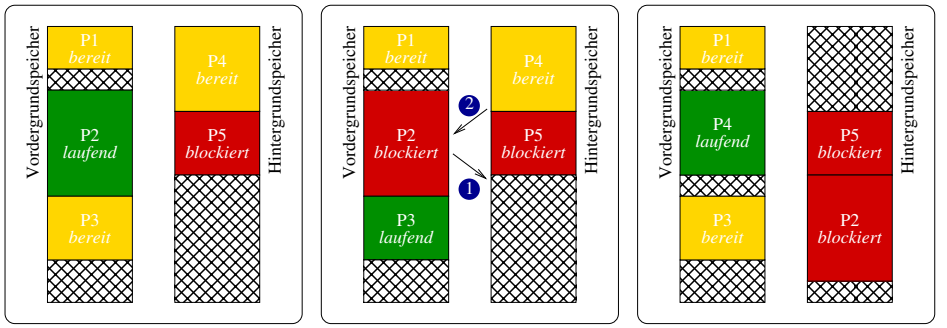
Umlagerung von solchen Bereichen gegenwärtig nicht ausführbereiter Programme (engl. *swapping*) verschiebt die Grenze nach hinten

- schafft Platz für ein oder mehrere andere (zusätzliche) Programme
- lässt mehr Programme zu, als insgesamt in den Arbeitsspeicher passt

Auslagerung von Bereichen sich in Ausführung befindlicher Programme (engl. *paging*, *segmentation*) liefert weiteren Spielraum [2]

Umlagerung nicht ausführbereiter Programme

Funktion der mittelfristigen Einplanung (engl. *medium-term scheduling*) von Prozessen



Ausgangssituation:
- P[1-3] im RAM
- P2 belegt die CPU

Umlagerung P2 & P4:
1. *swap out*
2. *swap in*

Resultat:
- P[134] im RAM
- P4 belegt die CPU

Problem
- Fragmentierung, Verdichtung, Körnigkeit

Umlagerung ausführbereiter Programme

Prozesse laufen trotz Hauptspeichermangel ab: **virtueller Speicher**
- nicht benötigte Programmteile liegen im Hintergrundspeicher
  - sie werden erst bei Bedarf (engl. *on demand*) nachgeladen
  - ggf. sind als Folge andere Programmteile vorher zu verdrängen
- Zugriff auf ausgelagerte Teile bedeutet **partielle Interpretation**
  - logisch bleibt das unterbrochene Programm weiter in Ausführung
  - physisch wird es jedoch im Zuge der Einlagerung (E/A) blockieren
- Aus- und Einlagerung wechseln sich mehr oder wenig intensiv ab

> *Der Arbeitsspeicherbedarf eines/mehrerer scheinbar (virtuell) komplett im Arbeitsspeicher liegender und laufender/laufbereiter Programme kann die Größe des wirklichen (physikalischen) Arbeitsspeichers weit überschreiten.*

Problem
- Lade- und Ersetzungsstrategien

Körnigkeit (engl. *granularity*) der Umlagerungseinheiten

Programmteile, die ein-, aus- und/oder überlagert werden können, sind **Seiten** (engl. *pages*) oder **Segmente** (engl. *segments*):

Seitenüberlagerung (engl. *paging*) Atlas [7]
Adressraumteile fester Größe (Seiten) werden auf entsprechend große Teile (Seitenrahmen) des Haupt-/Hintergrundspeichers abgebildet

Segmentierung (engl. *segmentation*) B 5000 [17]
Adressraumteile variabler Größe (Segmente) werden auf entsprechend große Teile des Haupt-/Hintergrundspeichers abgebildet

seitennummerierte Segmentierung (*paged segmentation*) GE 635 [8]
Kombination beider Verfahren: Segmente in Seiten untergliedern

Problem
- interne Fragmentierung (Seiten), externe Fragmentierung (Segmente)

Automatische Überlagerung durch partielle Interpretation

Ähnlichkeit zur Überlagerungstechnik, jedoch...

- Seiten-/Segmentanforderungen sind nicht in den Anwendungen programmiert, stattdessen im Betriebssystem
  - die Anforderungen stellt stellvertretend ein Systemprogramm
  - Ladeanweisungen sind so vor dem Anwendungsprogramm verborgen
- Zugriffe auf ausgelagerte Seiten/Segmente werden von der Befehlssatzebene abgefangen und ans Betriebssystem weitergeleitet
  - das Anwendungsprogramm wird von CPU bzw. MMU unterbrochen
  - der Zugriff wird vom Betriebssystem partiell interpretiert
- Wiederholungen des unterbrochenen Maschinenbefehls sind in Betracht zu ziehen und vom Betriebssystem zu veranlassen

Problem

- Komplexität, Determiniertheit

Universalbetriebssystem — „Allgemeinzweck“

(engl. *general purpose operating system*)

Betriebssystemdilemma — Software zwischen Baum und Borke

- *Clearly, the operating system design must be strongly influenced by the type of use for which the machine is intended.*
- *Unfortunately it is often the case with 'general purpose machines' that the type of use cannot easily be identified;*
- *a common criticism of many systems is that, in attempting to be all things to all individuals, they end up being totally satisfactory to no-one.*

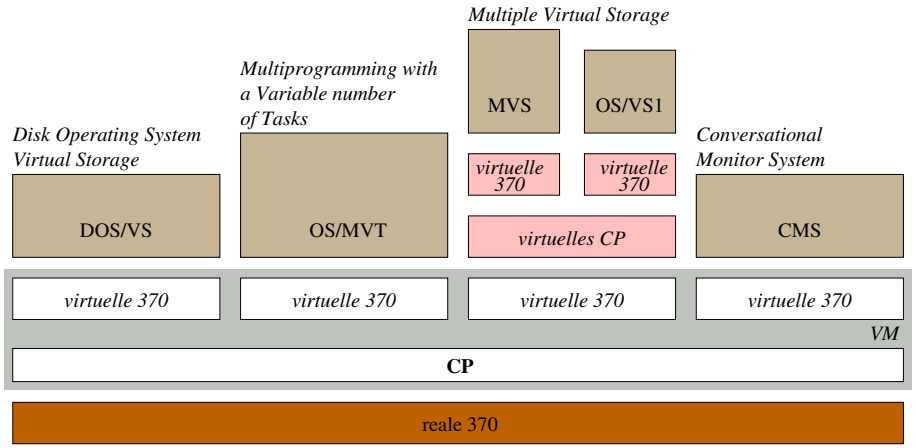
(Lister, „Fundamentals of Operating Systems“ [16])

Verbesserung von Benutzerzufriedenheit:

- Selbstvirtualisierung (virtuelle Maschinen)
- Konzentration auf das Wesentliche

Spezialisierung durch virtuelle Maschinen

VM 370 (IBM, 1968)



Konzentration auf das Wesentliche

UNIX [24, 15, 14] — Ein Betriebssystemkern von 10^4 Zeilen C und nicht 10^6 Zeilen PL/I

Multics		**UNICS**
Multiplexed	$\Longleftrightarrow$	Uniplexed
Information and Computing System		

- ITS nicht zu vergessen (S. 6)

„Lotta hat einen Unixtag“, Astrid Lindgren [13, S. 81–89]

Die drei Jahre alte Lotta ist die kleine Schwester der Erzählerin. Läuft am Tag vieles schief bei ihr, sagt sie „Unixtag“, meint aber „Unglückstag“.

Gliederung

Verarbeitung von Programmen in Echtzeit

Zustandsänderung von Programmen ist eine Funktion der **realen Zeit** [11]

- korrektes Verhalten des Systems hängt nicht nur von den logischen Ergebnissen von Berechnungen ab
- zusätzlicher Aspekt ist der **physikalische Zeitpunkt** der Erzeugung und Verwendung der Berechnungsergebnisse

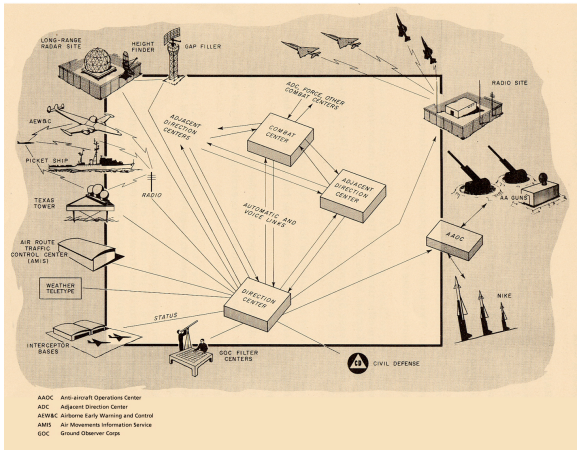
DIN 44300 [4]

Echtzeitbetrieb ist ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorbestimmten Zeitpunkten anfallen.

- Whirlwind (MIT, 1951), AN/FSQ-7 (Whirlwind II, IBM, 1957)
- SAGE (*semi-automatic ground environment*, 1958–1983)

SAGE und AN/FSQ-7 „Whirlwind II“

Erstes Echtzeitrechensystem als Schöpfung des „Kalten Krieges“



- 27 Installationen über die USA verteilt
  - *Nonstop*-Betrieb
  - 25 Jahre
- durch Datenfernleitungen miteinander gekoppelt
  - Telefonleitungen
  - Internet-„Mutter“
- pro Installation. . .
  - 100 Konsolen
  - 500 KLOC Ass.

Echtzeitfähigkeit bedeutet Rechtzeitigkeit

Zuverlässige Reaktion des Rechensystems auf Umgebungsereignisse

Geschwindigkeit liefert keine Garantie, um rechtzeitig Ergebnisse von Berechnungen abliefern und Reaktionen darauf auslösen zu können

- die im Rechensystem verwendete Zeitskala muss mit der durch die Umgebung vorgegebenen identisch sein
- **„Zeit“ ist keine intrinsiche Eigenschaft des Rechensystems**

Determiniertheit und Determinismus

- bei ein und derselben Eingabe sind verschiedene Abläufe zulässig, alle Abläufe liefern jedoch stets das gleiche Resultat
- zu jedem Zeitpunkt ist bestimmt, wie weitergefahren wird

- die **Terminvorgaben** gelten als weich, fest oder hart

Terminvorgaben

Festgelegt durch die Umgebung („externe Prozesse“) des Rechensystems

weich (engl. *soft*) auch „schwach“
- das Ergebnis einer zu einem vorgegebenen Termin nicht geleisteten Arbeit ist weiterhin von Nutzen, verliert aber an Wert
- Terminverletzung ist tolerierbar (z.B. Ton-/Bildrate)

fest (engl. *firm*) auch „stark“
- das Ergebnis einer zu einem vorgegebenen Termin nicht geleisteten Arbeit ist wertlos und wird verworfen (z.B. Übungsaufgabe)
- Terminverletzung ist tolerierbar, führt zum Arbeitsabbruch

hart (engl. *hard*) auch „strikt“
- das Versäumnis eines fest vorgegebenen Termins kann eine „Katastrophe“ hervorrufen
- Terminverletzung ist keinesfalls tolerierbar (z.B. Airbag)

Problem
- Termineinhaltung unter allen Last- und Fehlerbedingungen

Terminvorgaben: Fest $\Longleftrightarrow$ Hart

Terminverletzung bedingt als Ausnahmesituation begreifen müssen

fest/hart $\mapsto$ Terminverletzung ist nicht ausgeschlossen[1]
- die Terminverletzung wird vom Betriebssystem erkannt

fest $\rightsquigarrow$ plangemäß weiterarbeiten
- das Betriebssystem bricht den Arbeitsauftrag ab
- der nächste Arbeitsauftrag wird gestartet
- ist transparent für die Anwendung

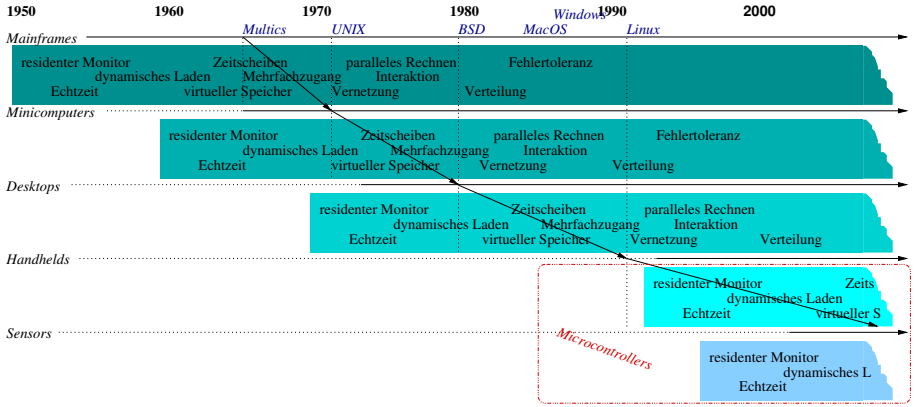
hart $\rightsquigarrow$ sicheren Zustand finden
- das Betriebssystem löst eine **Ausnahmesituation** aus
- die Ausnahmebehandlung führt zum sicheren Zustand
- ist **intransparent für die Anwendung**

[1] Auch wenn Ablaufplan und Betriebssystem theoretisch Determinismus zeigen, kann das im Feld eingesetzte technische System von Störeinflüssen betroffen sein, die ggf. die Verletzung auch eines harten Termins nach sich ziehen.

Gliederung

Migration von Betriebssystemkonzepten

Fortschritt in der Hardwareentwicklung kommt Betriebssystemen zu Gute



(Vorlage: Silberschatz/Galvin/Gagne [23])

Stand der Kunst — „Quietschbuntes Brimborium“

Viel Getöse für altbackene Technik?

Linux „*yet another UNIX-like operating system*“, aber was soll's...

- Entwicklungsprozess und -modell sind der eigentliche „Kick“
- 70er-Jahre Technologie — ohne Multics (funktional) zu erreichen

Windows „*new technology*“, wirklich?

- vor WNT entwickelte Cuttler VMS (DEC): $WNT = VMS + 1$
- mit 94 % Marktführer im PC-Sektor — für 2 % des Prozessormarktes

MacOS X ein vergleichsweise echter Fortschritt

- solides UNIX (Free**BSD**) auf solider Mikrokernbasis (**Mach**)
- Apple bringt PC-Technologie erneut voran — bei $\leq 4\,\%$ Marktanteil

Resümee

- **Mehrzugangsbetrieb** ermöglicht Arbeit und Umgang mit einem Rechensystem über mehrere Dialogstationen
  - im Teilnehmerbetrieb setzen Dialogstationen eigene Dialogprozesse ab
  - im Teilhaberbetrieb teilen sich Dialogstationen einen Dialogprozess
- wichtige **Systemmerkmale** zur Unterstützung dieser Betriebsart
  - Parallelverarbeitung durch (speichergekoppelte) Multiprozessoren
  - über bloße Adressraumisolation hinausgehende Schutzvorkehrungen
  - auf Programm(teil)umlagerung ausgerichtete Speicherverwaltung
  - dem Allgemeinzweck gewidmete Universalbetriebssysteme

- **Echtzeitbetrieb** — querschneidende Betriebsart des Rechensystems
  - Zustandsänderung von Programmen wird zur Funktion der realen Zeit
    - „Zeit“ ist keine intrinsische Eigenschaft des Rechensystems mehr
  - „externe Prozesse“ definieren Terminvorgaben, die einzuhalten sind
    - die Echtzeitbedingungen dabei gelten als weich, fest oder hart

Literaturverzeichnis

[1] Corbató, F. J. ; Daggett, M. M. ; Daley, R. C.:
An Experimental Time-Sharing System.
In: *1962 Proceedings of the Spring Joint Computer Conference, May 1–3, 1962, San Francisco, USA* American Federation of Information Processing Societies, AFIPS Press, 1962, S. 335–344

[2] Denning, P. J.:
Virtual Memory.
In: *Computing Surveys* 2 (1970), Sept., Nr. 3, S. 153–189

[3] Dennis, J. B. ; Horn, E. C. V.:
Programming Semantics for Multiprogrammed Computations.
In: *Communications of the ACM* 9 (1966), März, Nr. 3, S. 143–155

[4] Deutsches Institut für Normung:
Informationsverarbeitung — Begriffe.
Berlin, Köln, 1985 (DIN 44300)

[5] Eastlake, D. E. ; Greenblatt, R. D. ; Holloway, J. T. ; Knight, T. F. ; Nelson, S. :
ITS 1.5 Reference Manual / MIT.
Cambridge, MA, USA, Jul. 1969 (AIM-161A). – Forschungsbericht

Literaturverzeichnis (Forts.)

[6] Fabry, R. S.:
Capability-Based Addressing.
In: *Communications of the ACM* 17 (1974), Jul., Nr. 7, S. 403–412

[7] Fotheringham, J. :
Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store.
In: *Communications of the ACM* 4 (1961), Okt., Nr. 10, S. 435–436

[8] General Electric Company (Hrsg.):
GE-625/635 Programming Reference Manual.
CPB-1004A.
Phoenix, AZ, USA: General Electric Company, Jul. 1964

[9] Graham, G. S. ; Denning, P. J.:
Protection—Principles and Practice.
In: *1972 Proceedings of the Spring Joint Computer Conference, May 6–8, 1972, Atlantic City, USA* American Federation of Information Processing Societies, AFIPS Press, 1972, S. 417–429

[10] 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.)

[11] Kopetz, H. :
Real-Time Systems: Design Principles for Distributed Embedded Applications.
Kluwer Academic Publishers, 1997. –
ISBN 0–7923–9894–7

[12] Lampson, B. W.:
Protection.
In: *Proceedings of the Fifth Annual Princeton Conference on Information Sciences and Systems*.
New Jersey, USA : Department of Electrical Engineering, Princeton University, März 1971, S. 437–443

[13] *Kapitel* Lotta hat einen Unixtag.
In: Lindgren, A. :
Die Kinder aus der Krachmacherstraße.
Oettinger-Verlag, 1957. –
ISBN 3–7891–4118–6, S. 81–89

[14] Lions, J. :
A Commentary on the Sixth Edition UNIX Operating System.
The University of New South Wales, Department of Computer Science, Australia :
http://www.lemis.com/grog/Documentation/Lions, 1977

Literaturverzeichnis (Forts.)

[15] Lions, J. :
UNIX Operating System Source Code, Level Six.
The University of New South Wales, Department of Computer Science, Australia :
http://v6.cuzuco.com, Jun. 1977

[16] Lister, A. M. ; Eager, R. D.:
Fundamentals of Operating Systems.
The Macmillan Press Ltd., 1993. –
ISBN 0–333–59848–2

[17] Mayer, A. J. W.:
The Architecture of the Burroughs B5000: 20 Years Later and Still Ahead of the Times?
In: *ACM SIGARCH Computer Architecture News* 10 (1982), Jun., Nr. 4, S. 3–10

[18] Organick, E. I.:
The Multics System: An Examination of its Structure.
MIT Press, 1972. –
ISBN 0–262–15012–3

Literaturverzeichnis (Forts.)

[19] Roberts, L. G.:
Multiple Computer Networks and Intercomputer Communication.
In: Gosden, J. (Hrsg.) ; Randell, B. (Hrsg.): *Proceedings of the First ACM Symposium on Operating System Principles (SOSP '67), October 1–4, 1967, Gatlinburg, TN, USA*, ACM, 1967, S. 3.1–3.6

[20] Saltzer, J. H.:
Protection and the Control of Information Sharing in Multics.
In: *Communications of the ACM* 17 (1974), Jul., Nr. 7, S. 388–402

[21] Saltzer, J. H. ; Schroeder, M. D.:
The Protection of Information in Computer Systems.
In: *Proceedings of the IEEE* 63 (1975), Sept., Nr. 9, S. 1278–1308

[22] Schroeder, M. D. ; Saltzer, J. H.:
A Hardware Architecture for Implementing Protection Rings.
In: *Proceedings of the Third ACM Symposium on Operating System Principles (SOSP 1971), October 18–20, 1971, Palo Alto, California, USA*, ACM, 1971, S. 42–54

[23] Silberschatz, A. ; Galvin, P. B. ; Gagne, G. :
Operating System Concepts.
John Wiley & Sons, Inc., 2001. –
ISBN 0–471–41743–2

Literaturverzeichnis (Forts.)

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

[25] 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

[26] *Multics Home*.
http://www.multicians.org, Nov. 2010