Systemprogrammierung

Grundlage von Betriebssystemen

Teil B – VII.2 Betriebsarten: Dialog- und Echtzeitverarbeitung

Wolfgang Schröder-Preikschat

8. Juli 2015

Agenda

Gliederung

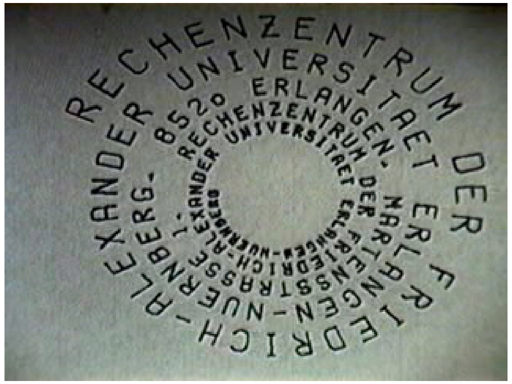
Lehrstoff

- weiterhin ist das Ziel, „zwei Fliegen mit einer Klappe zu schlagen“:
  - i einen Einblick in **Betriebssystemgeschichte** zu geben und
  - ii damit gleichfalls **Betriebsarten** von Rechensystemen zu erklären
- im Vordergrund stehen die Entwicklungsstufen im **Dialogbetrieb**, der Dialogprozesse einführt, d.h., Prozesse:
  - die an der Konkurrenz um gemeinsame Betriebsmittel teilnehmen
  - die Benutzer/innen an einer Dienstleistung teilhaben lassen
- kennzeichnend ist, Programmausführung **interaktiv** zu gestalten
  - mitlaufend (*on-line*) den Prozessfortschritt beobachten und überwachen
  - dazu spezielle Schutzvorkehrungen und eine effektive Speicherverwaltung

> *Viele dieser Techniken — wenn nicht sogar alle — sind auch heute noch in einem **Universalbetriebssystem** auffindbar.*

- des Weiteren erfolgt ein kurzer Einblick in den **Echtzeitbetrieb**, der an sich quer zu all den betrachteten Betriebsarten liegt

Rechnerbetrieb am RRZE[1] um 1973



[1] http://www.rrze.uni-erlangen.de/wir-ueber-uns/publikationen/das-rrze-der-film.shtml

Gliederung

Dialogbetrieb

- Benutzereingaben und Verarbeitung wechseln sich anhaltend ab
  - E/A-intensive Anwendungsprogramme interagieren mit Benutzer/inne/n
  - Zugang über **Dialogstationen** (*interactive terminals*)
    - **Datensichtgerät** und **Tastatur** (seit 1950er Jahren, Whirlwind/SAGE)
    - später die ***Maus*** Engelbart/English, SRI, 1963/64; vorgestellt 1968)
- **dynamische Einplanung** (*dynamic scheduling*, *on-line*) hält Einzug, bevorzugt **interaktive** (E/A-intensive) **Prozesse**
  - Beendigung von Ein-/Ausgabe führt zur „prompten“ Neueinplanung
  - Benutzer/innen erfahren eine schnelle Reaktion insb. auf Eingaben
- **Problem**:
  - Zusatz (*add-on*) zum Stapelbetrieb, Monopolisierung der CPU, Sicherheit

Anekdote (*add-on*: aus eigener Erfahrung im Studium)

Unter den Studierenden hatte sich schnell herumgesprochen, mittels Tastatureingaben die Dringlichkeit ihrer im Hintergrund ablaufenden Programmausführung anheben zu können.

Dialogororientiertes Monitorsystem

- Prozesse „im Vordergrund“ starten und „im Hintergrund“ vollziehen
  - in **Konversation** Aufträge annehmen, ausführen und dabei überwachen
    - d.h. Prozesse starten, stoppen, fortführen und ggf. abbrechen
  - zur selben Zeit laufen im Rechensystem mehrere Programme parallel ab
  - mehrere Aufgaben (*task*) werden „gleichzeitig“ bearbeitet (*multitasking*)
- in weiterer Konsequenz lässt sich so **Mischbetrieb** unterstützen, z.B.:

Vordergrund	echtzeitabhängige Prozesse $\leadsto$ Echtzeitbetrieb	(Realzeit)
Mittelgrund	E/A-intensive Prozesse $\leadsto$ Dialogbetrieb	(Antwortzeit)
Hintergrund	CPU-intensive Prozesse $\leadsto$ Stapelbetrieb	(Rechenzeit)

- **Problem**:
  - Hauptspeicher(größe)

Hinweis (Mischbetrieb)

*Zeit ist ein wichtiger Aspekt, jedoch ist dabei das **Bezugssystem** zu beachten: Antwort-/Rechenzeit hat nur das Rechensystem, Echtzeit jedoch vor allem die (phys.) Umgebung als Bezugspunkt.*

Dialog mit teilnehmenden Prozessen

time sharing

- eigene Dialogprozesse werden interaktiv gestartet und konkurrieren mit anderen (Dialog-) Prozessen um gemeinsame Betriebsmittel
  - um CPU-Monopolisierung vorzubeugen, werden CPU-Stöße partitioniert, indem Prozesse nur eine **Zeitscheibe** (*time slice*) lang „laufend“ sind
  - CPU-Zeit ist damit eine Art **konsumierbares Betriebsmittel**, um das wiederverwendbare Betriebsmittel „CPU“ beanspruchen zu können
  - ist die Zeitscheibe abgelaufen, muss ein Prozess die CPU abgeben, bevor er eine neue Zeitscheibe zum Verbrauch erhalten kann
  - jeder Dialogprozess „nimmt teil“ an der **Konkurrenz** um Betriebsmittel
- technische Grundlage liefert ein **Zeitgeber** (*timer*), der für **zyklische Unterbrechungen** (*timer interrupt*) sorgt
  - der unterbrochene Prozess wird neu eingeplant: „laufend“ $\mapsto$ „bereit“
    - ihm wird die CPU zu Gunsten eines anderen Prozesses entzogen
    - er erfährt die **Verdrängung** (*preemption*) von „seinem“ Prozessor
  - aber nur, sofern es einen anderen Prozess im Zustand „bereit“ gibt
- **Problem**:
  - Hauptspeicher, Einplanung, Einlastung, Ein-/Ausgabe, Sicherheit

Bahnbrecher und Wegbereiter I

- **CTSS** (*Compatible Time-Sharing System* [1], MIT, 1961)
  - Pionierarbeit zu interaktiven Systemen und zur Prozessverwaltung
    - **partielle Virtualisierung**: Prozessinkarnation als virtueller Prozessor
    - **mehrstufige Einplanung** (*multi-level scheduling*) von Prozessen
    - zeilenorientierte Verarbeitung von Kommandos (u.a. `printf` [1, S. 340])
  - vier Benutzer gleichzeitig: drei im Vordergrund, einen im Hintergrund
- **ITS** (*Incompatible Time-sharing System* [5], MIT, 1969)
  - Pionierarbeit zur Ein-/Ausgabe und Prozessverwaltung:
    - **geräteunabhängige Ausgabe** auf Grafikbildschirme, virtuelle Geräte
    - netzwerktransparenter Dateizugriff (über ARPANET [23])
    - **Prozesshierarchien**, Kontrolle untergeordneter Prozesse (`^Z` [5, S. 13])
  - „Seitenhieb“ auf CTSS und Multics, wegen der eingeschlagenen Richtung

Hinweis (Zeitteilverfahren)

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. [21]

Dialog mit teilhabenden Prozessen

transaction mode

- ein von mehreren Dialogstationen aus gemeinsam benutzter, zentraler Dialogprozess führt die abgesetzten Kommandos aus
  - mehrere Benutzer „haben Teil“ an der Dienstleistung eines Prozesses, die Bedienung regelt ein einzelnes Programm
  - gleichartige, bekannte und festverdrahtete (*hard-wired*) Aktionen können von verschiedenen Benutzern zugleich ausgelöst werden
- das den Dialogprozess vorgebende **Dienstprogramm** steht für einen **Endbenutzerdienst** mit festem, definiertem Funktionsangebot
  - Kassen, Bankschalter, Auskunft-/Buchungssysteme, ...
  - allgemein: **Transaktionssysteme**
- **Problem**:
  - Antwortverhalten (weiche/feste Echtzeit), Durchsatz

Hinweis

*Heute bekannt als Klient/Anbieter-System (client/server system), in dem **Dienstnehmer** (service user) mit einem **Dienstgeber** (service provider) interagieren.*

Gliederung

Dialog mit Echtzeitprozessen

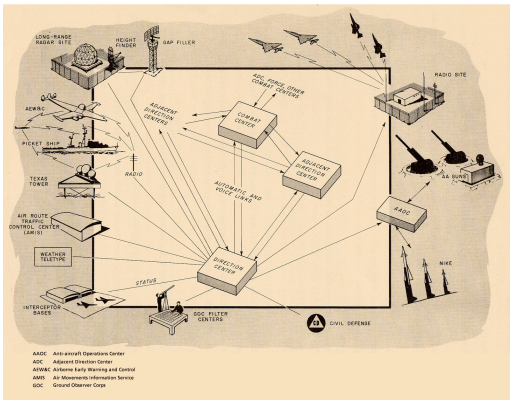
Definition (Echtzeitbetrieb, in Anlehnung an DIN 44300 [4])

[. . .] 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 werden nach einer zeitlich zufälligen Verteilung (ereignisgesteuert) oder zu vorbestimmten Zeitpunkten (zeitgesteuert) verwendet.

- kennzeichnend ist, dass die Zustandsänderung von Prozessen durch eine Funktion der **realen Zeit** [13] definiert ist
  - das korrekte 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
- **interne Prozesse** des Rechensystems müssen **externe Prozesse** der phyikalischen Umgebung des Rechensystems steuern/überwachen

Bahnbrecher und Wegbereiter II

- **SAGE** (*semi-automatic ground environment*, 1958–1983)
  - Whirlwind (MIT, 1951), AN/FSQ-7 (Whirlwind II, IBM, 1957)
- erstes Echtzeitrechensystem — eine Schöpfung des „Kalten Krieges“:



- 27 Installationen über die USA verteilt
  - *Nonstop*-Betrieb
  - 25 Jahre
- durch Datenfernleitungen miteinander gekoppelt
  - Telefonleitungen
  - Vorläufer des Internets
- pro Installation:
  - 100 Konsolen
  - 500 KLOC Assembler

Echtzeitfähigkeit bedeutet Rechtzeitigkeit

- im Vordergrund steht die **zuverlässige Reaktion** des Rechensystems auf Ereignisse in der Umgebung des Rechensytsems
  - interne Prozesse erhalten jeweils einen **Termin** (*deadline*) vorgegeben, bis zu dem ein Berechnungsergebnis abzuliefern ist
  - die **Terminvorgaben** sind weich (*soft*), fest (*firm*) oder hart (*hard*), sie sind durch die externen Prozessen bestimmt
- 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**

Hinweis (Determiniertheit und Determinismus)

Einerseits sind bei ein und derselben Eingabe verschiedene Abläufe zulässig, die dann jedoch stets das gleiche Resultat liefern müssen. Andererseits muss zu jedem Zeitpunkt im Rechensystem bestimmt sein, wie weitergefahren wird.

Terminvorgaben

- externe (physikalische) Prozesse definieren, was genau bei einer nicht termingerecht geleisteten Berechnung zu geschehen hat:

  weich (*soft*) auch „schwach“
  - das Ergebnis ist weiterhin von Nutzen, verliert jedoch mit jedem weiteren Zeitverzug des internen Prozesses zunehmend an Wert
  - die Terminverletzung ist tolerierbar

  fest (*firm*) auch „stark“
  - das Ergebnis ist wertlos, wird verworfen, der interne Prozess wird abgebrochen und erneut bereitgestellt
  - die Terminverletzung ist tolerierbar

  hart (*hard*) auch „strikt“
  - Verspätung der Ergebnislieferung kann zur „Katastrophe“ führen, dem internen Prozess wird eine **Ausnahmesituation** zugestellt
  - Terminverletzung ist keinesfalls tolerierbar — aber möglich. . .
- **Problem**:
  - Termineinhaltung unter allen Last- und Fehlerbedingungen

Terminvorgaben: fest $\Longleftrightarrow$ hart

- eine Terminverletzung bedeutet grundsätzlich die Ausnahmesituation, deren Behandlung jedoch auf verschiedenen Ebenen erfolgt
  - im Betriebssystem (fest) oder im Maschinenprogramm (hart)
  - im „harten Fall“ also im Anwendungsprogramm des späten Prozesses
- das Betriebssystem erkennt die Verletzung, die Anwendung muss aber plangemäß weiterarbeiten (fest) | den sicheren Zustand finden (hart)
  - das Betriebssystem bricht die Berechnung ab
  - die nächste Berechnung wird gestartet
  - transparent für die Anwendung
  - das Betriebssystem löst eine Ausnahmesituation aus
  - die Ausnahmebehandlung führt zum sicheren Zustand
  - **intransparent** für die Anwendung

Hinweis (Terminverletzung)

Auch wenn Ablaufplan und Betriebssystem in Theorie „am Reißbrett“ deterministisch sind, kann in Praxis das Rechensystem Störeinflüssen unterworfen sein und so Termine verpassen.

Gliederung

Symmetrische Simultanverarbeitung

Definition (SMP, *symmetric multiprocessing*)

Zwei oder mehr gleiche (identische) Prozessoren, eng gekoppelt über ein gemeinsames Verbindungssystem.

- erfordert ganz bestimmte **architektonische Merkmale** sowohl von der Befehlssatz- als auch von der Maschinenprogrammebene
  - jeder Prozessor hat gleichberechtigten Zugriff auf den Hauptspeicher (*shared-memory acccess*) und die Peripherie
  - der Zugriff auf den Hauptspeicher ist für alle Prozessoren gleichförmig (*uniform memory access*, UMA)
    - bedingt in nichtfunktionaler Hinsicht, sofern nämlich die **Zyklenanzahl pro Speicherzugriff** betrachtet wird
    - unbedingt aber im funktionalen Sinn bezogen auf die Maschinenbefehle
  - die Prozessoren stellen ein **homogenes System** dar und sie werden von demselben Betriebssystem verwaltet
- **Problem**:
  - Synchronisation, Skalierbarkeit

Speichergekoppelter Multiprozessor

Definition (SMP, *shared-memory processor*)

Ein Parellelrechnersystem, in dem alle Prozessoren den Hauptspeicher mitbenutzen, ohne jedoch einen gleichberechtigten/-förmigen Zugriff darauf haben zu müssen.

- **architektonische Merkmale** der Befehlssatzebene geben bestimmte Freiheitsgrade für die Simultanverarbeitung vor

  asymmetrisch
  - hardwarebedingter, zwingender asymmetrischer Betrieb
  - Programme sind ggf. prozessorgebunden
  - ↪ *asymmetric multiprocessing*

  symmetrisch
  - **anwendungsorientierter Betrieb** wird ermöglicht
  - das Betriebssystem legt die Multiprozessorbetriebsart fest
  - *symmetric/asymmetric multiprocessing*
- die Maschinenprogrammebene kann ein **heterogenes System** bilden, in funktionaler und nichtfunktionaler Hinsicht
- **Problem**:
  - Synchronisation, Skalierbarkeit, Anpassbarkeit

Parallelverarbeitung

parallel processing

- N Prozessoren können...
  - N verschiedene oder identische Programme,
  - N Fäden dieser Programme oder
  - N Fäden ein und desselben Programms

  ...**echt parallel** ausführen
- jeder dieser N Prozessoren kann...
  - M verschiedene oder identische Programme,
  - M Fäden dieser Programme oder
  - M Fäden ein und desselben Programms

  ...**pseudo/quasi parallel** im Multiplexbetrieb ausführen
- all diese bis zu $N \times M$ Ausführungsstränge finden **nebenläufig** statt

Hinweis (Synchronisation)

Koordination der Kooperation und Konkurrenz gleichzeitiger Prozesse ist nur bedingt davon abhängig, dass Rechensystembetrieb echt oder pseudo/quasi parallel geschieht.

Sicherheit

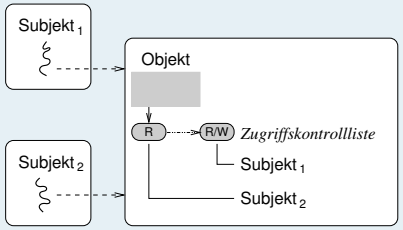
security

- **Schutz** (*protection*) vor unautorisierten Zugriffen durch Prozesse, der in Körnigkeit und Funktion sehr unterschiedlich ausgelegt sein kann
  - i jeden Prozessadressraum in **Isolation** betreiben
    - Schutz durch Eingrenzung oder Segmentierung
    - Zugriffsfehler führen zum Abbruch der Programmausführung
    - i.A. keine selektive Zugriffskontrolle möglich und sehr grobkörnig
  - ii Prozessen eine **Befähigung** (*capability* [3, 28, 6]) zum Zugriff erteilen
    - den verschiedenen **Subjekten** (Prozesse) individuelle Zugriffsrechte geben, z.B., ausführen, lesen, schreiben oder ändern dürfen
    - und zwar auf dasselbe von ihnen mitbenutzte **Objekt** (Datum, Datei, Gerät, Prozedur, Prozess)
  - iii Objekten eine **Zugriffskontrollliste** (*access control list*, ACL [25]) geben
    - ein Listeneintrag legt das Zugriffsrecht eines Subjekts auf das Objekt fest
    - vereinfacht auch in „Besitzer/in-Gruppe-Welt"-Form (*user/group/world*)
- **Problem**:
  - verdeckter Kanal (*covered channel*) bzw. Seitenkanal

Gegenüberstellung

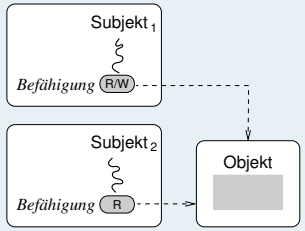
- feinkörniger Schutz durch **selektive Autorisierung** der Zugriffe:

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

access matrix [14, 9]

- in diesem allgemeinen Modell spezifiziert jeder Eintrag in der Matrix das individuelle Zugriffsrecht eines Subjekts auf ein Objekt:

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

- *read* R
- *write* W
- *execute* X

- je nach **Abspeicherung** und Verwendung der in der Matrix kodierten Information ergeben sich verschiedene Implementierungsoptionen:
  - Totalsicht ☹
    - in Form einer systembezogenen **Zugriffstabelle**
    - 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

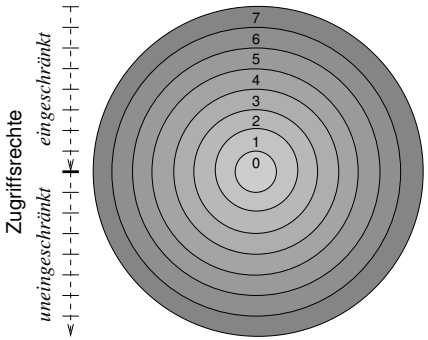
Bahnbrecher und Wegbereiter III

- **Multics** (*Multiplexed Information and Computing Service* [20], 1965)
  - setzt den 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 [26, 24]
  - **segmentbasierter Ringschutz** (*ring-protected paged segmentation*)
    - das ursprüngliche Konzept (für den GE 645) sah 64 Ringe vor, letztendlich bot die Hardware (Honeywell 6180) Unterstützung für acht Ringe
    - nicht in Hardware implementierte Ringe wurden durch Software emuliert
  - eng mit dem Segmentkonzept verbunden war **dynamisches Binden**:
    - jede Art von Information, ob Programmtext oder -daten, war ein Segment
    - Segmente konnten bei Bedarf (*on demand*) geladen werden
    - Zugriff auf ein ungeladenes Segment führte zum **Bindefehler** (*link trap*)
    - in Folge machte eine **Bindelader** (*linking loader*) das Segment verfügbar
- **Problem**:
  - Hardwareunterstützung

Schutzringe: Multics

Honeywell 6180

- 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



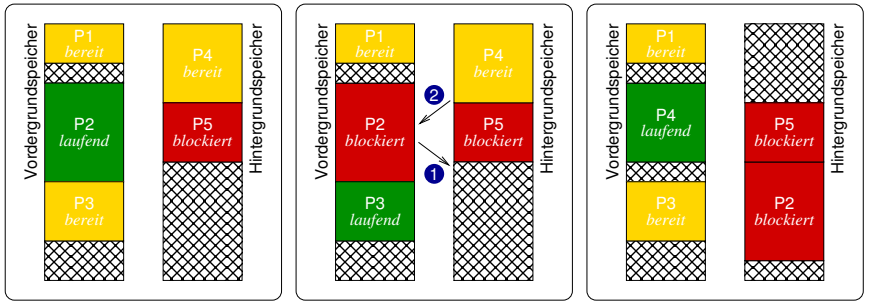
- je nach Prozessattribut/-aktion sind **Ringfehler** (*ring fault*) möglich
  - Folge ist die Teilinterpretation der Operation auf Ring 0 (*supervisor*)
  - unautorisierte Operationen führen zum **Schutzfehler** (*protection fault*)
- **Problem**:
  - Schichtenstruktur, Ringzuordnung: **funktionale Hierarchie** [11]

Grad an Mehrprogrammbetrieb

- die selektive Überlagerung des Hauptspeichers durch programmiertes dynamisches Laden (*overlay*) hat seine Grenzen
  - Anzahl × Größe hauptspeicherresidenter Text-/Datenbereiche begrenzt die Anzahl der gleichzeitig zur Ausführung vorgehaltenen Programme
  - variabler Wert, abhängig von Struktur/Organisation der Programme und den Fähigkeiten der Programmierer/innen
- **Umlagerung** der Speicherbereiche gegenwärtig nicht ausführbereiter Programme (*swapping*) verschiebt die Grenze nach hinten
  - schafft Platz für ein oder mehrere andere (zusätzliche) Programme
  - lässt mehr Programme zu, als insgesamt in den Hauptsspeicher passt
- Berücksichtigung solcher Bereiche der sich in Ausführung befindlichen Programme (*paging*, *segmentation*) gibt weiteren Spielraum [2]
  - im Unterschied zu vorher werden nur Teile eines Programms umgelagert
  - Programme liegen nur scheinbar („virtuell“) komplett im Hauptspeicher
- Prozesse belegen **Arbeitsspeicher**, nämlich den zu einem bestimmten Zeitpunkt beanspruchten Verbund von **Haupt- und Ablagespeicher**

Umlagerung nicht ausführbereiter Programme

- Funktion der mittelfristigen Einplanung (*medium-term scheduling*)



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

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

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

- **Problem**:
  - Fragmentierung, Verdichtung, Körnigkeit

Umlagerung ausführbereiter Programme

- Prozesse schreiten voran, obwohl die sie kontrollierenden Programme nicht komplett im Hauptspeicher vorliegen: **virtueller Speicher** [10]
  - die von einem Prozess zu einem Zeitpunkt scheinbar nicht benötigten Programmteile liegen im Hintergrund, im Ablagespeicher
    - sie werden erst bei Bedarf (*on demand*) nachgeladen
    - ggf. sind als Folge andere Programmteile vorher zu verdrängen
  - Zugriffe auf ausgelagerte Programmteile unterbrechen die Prozesse und werden durch **partielle Interpretation** ausgeführt
    - logisch bleibt der unterbrochene Prozess weiter in Ausführung
    - physisch wird er jedoch im Zuge der Einlagerung (E/A) blockieren
  - Aus- und Einlagerung wechseln sich mehr oder wenig intensiv ab
- **Problem**:
  - Lade- und Ersetzungsstrategien, Arbeitsmenge (*working set*)

Hinweis

Der Platzbedarf der scheinbar (virtuell) komplett im Hauptspeicher liegenden und laufenden Programme kann die Größe des wirklichen (realen) Hauptspeichers weit überschreiten.

Granularität der Umlagerungseinheiten

- Programmteile, die ein-, aus- und/oder überlagert werden können, sind **Seiten** oder **Segmente**:

  Seitennummerierung (*paging*) Atlas [7]
  - Einheiten (von Bytes) fester Größe
  - **Problem**: interne Fragmentierung ⇝ „*false positive*“ (Adresse)

  Segmentierung (*segmentation*) B 5000 [19]
  - Einheiten (von Bytes) variabler Größe
  - **Problem**: externe Fragmentierung ⇝ „*false negative*“ (Bruchstücke)

  seitennummerierte Segmentierung (*paged segmentation*) GE 635 [8]
  - Kombination beider Verfahren: Segmente aber in Seiten untergliedern
  - **Problem**: interne Fragmentierung (wie oben)
- sie werden abgebildet auf gleich große Einheiten des Hauptspeichers (eingelagert) oder Ablagespeichers (ausgelagert)
- **Problem**:
  - Fragmentierung (des Arbeitsspeichers) ⇝ Verschnitt

Automatische Überlagerung

Partielle Interpretation

- seiten- und/oder segmentbasierte Umlagerung zeigt Ähnlichkeiten zur **Überlagerungstechnik** (*overlay*), jedoch:
  - die Seiten-/Segmentanforderungen sind nicht im Maschinenprogramm zu finden, stattdessen im Betriebssystem (*pager, segment handler*)
    - die Anforderungen stellt stellvertretend ein Systemprogramm
    - Ladeanweisungen sind so vor dem Maschinenprogramm verborgen
  - Zugriffe auf ausgelagerte Seiten/Segmente fängt die Befehlssatzebene ab, die sie dann ans Betriebssystem weiterleitet (*trap*)
    - das Maschinenprogramm wird von CPU bzw. MMU unterbrochen
    - der gescheiterte Zugriff wird vom Betriebssystem partiell interpretiert
- des Weiteren fällt die **Wiederholung** des unterbrochenen Befehls an, die vom Betriebssystem zu veranlassen ist
  - der Speicherzugriff scheiterte beim Befehls- oder Operandenabruf
  - die CPU konnte die Operation noch nicht vollständig ausführen (*rerun*)
- **Problem**:
  - Komplexität, Determiniertheit

Universalbetriebssystem

(*general purpose operating system*)

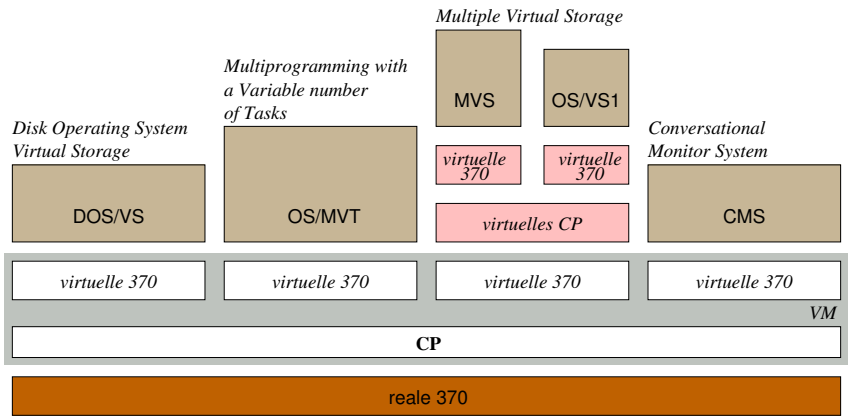
- bei einem Betriebssystem handelt es sich um Software, die zwischen Baum und Borke steckt, womit sich ein **Dilemma** ergibt:
  - *Clearly, the operating system design must be strongly influenced by the type of use for which the machine is intended.*
  - *Unfortunately it is often the case with ‘general 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“ [18])
- ein **Allzweckbetriebssystem** ist geprägt von Kompromissen, die sich quer durch die Implementierung ziehen
  - damit Echtzeitbetrieb aber ausschließen, der kompromisslos sein muss!
- Ansätze für verbesserte Akzeptanz sind Virtualisierung einerseits und „Konzentration auf das Wesentliche“ andererseits
  - auch damit bleibt ein Betriebssystem **domänenspezifische Software**

Virtualisierung

VM 370 (IBM, 1968)

- Spezialisierung durch virtuelle Maschinen:



CP
- Abk. für *control program*: **Hypervisor**, VMM
- **Selbstvirtualisierung** (para/voll, [12, S. 30]) des realen System/370

Konzentration auf das Wesentliche

„Lotta hat einen Unixtag“, Astrid Lindgren [15, 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“.

- UNIX [22, 17, 16]
  - ein Betriebssystemkern von 10^4 Zeilen C und nicht 10^6 Zeilen PL/I

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

- ITS nicht zu vergessen (S. 10)

Hinweis (UNIX $\overset{?}{\mapsto}$ Unglück)

Vom ursprünglichen Ansatz eines nur wesentliche Dinge enthaltenes, schlankes Betriebssystem ist heute wenig zu spüren.

Gliederung

Stand der Kunst

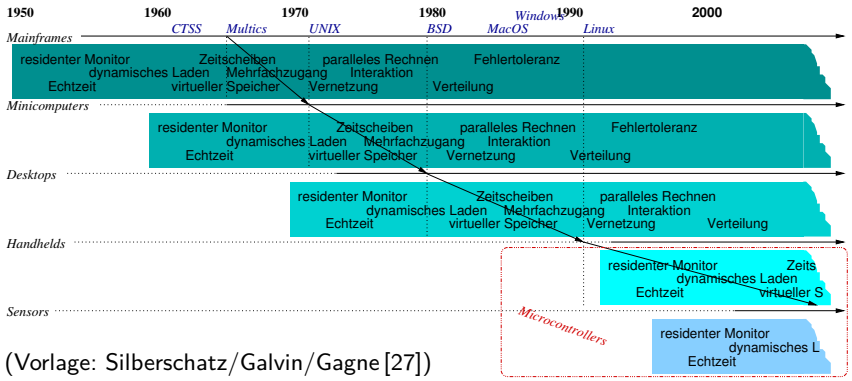
Viel Getöse für altbackene Technik?

- **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

Hinweis („Des Kaisers neue Kleider“)

Funktionsumfang wie auch Repräsentation vermeintlich moderner Betriebssysteme, lässt den Schluss zu, dass so einige Male das Rad neu erfunden wurde.

Migration von Betriebssystemkonzepten



(Vorlage: Silberschatz/Galvin/Gagne [27])

- Fähigkeit zur „Wanderung“ zu anderen, kleineren Gefilden fällt nicht vom Himmel, sondern bedarf sorgfältiger **Konzeptumsetzung**
- Voraussetzung dafür ist eine **Domänenanalyse**, um gemeinsame und variable Konzeptanteile zu identifizieren

Resümee

. . . die eierlegende Wollmilchsau gibt es nicht!

- **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
- **Echtzeitbetrieb** muss kompromisslos sein, da das Zeitverhalten des Rechensystems sonst unvorhersehbar ist
  - 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
- wichtige **Systemmerkmale** insbesondere für Mehrzugangsbetrieb:
  - Parallelverarbeitung durch (speichergekoppelte) Multiprozessoren
  - über bloße Adressraumisolation hinausgehende Schutzvorkehrungen
  - auf Programm(teil)umlagerung ausgerichtete Speicherverwaltung
- **Allzweckbetriebssysteme** sind universal, indem sie Fähigkeiten für die verschiedensten Bereiche umfassen — aber nicht für alle. . .

Literaturverzeichnis I

[1] Corbató, F. J. ; Merwin-Daggett, M. ; Dalex, R. C.:
An Experimental Time-Sharing System.
In: *Proceedings of the AIEE-IRE '62 Spring Joint Computer Conference*, ACM, 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 II

[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

Literaturverzeichnis III

[10] Güntsch, F.-R. :
Logischer Entwurf eines digitalen Rechengeräts mit mehreren asynchron laufenden Trommeln und automatischem Schnellspeicherbetrieb, Technische Universität Berlin, Diss., März 1957

[11] 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

[12] Kleinöder, J. ; Schröder-Preikschat, W. :
Virtuelle Maschinen.
In: Lehrstuhl Informatik 4 (Hrsg.): *Systemprogrammierung*.
FAU Erlangen-Nürnberg, 2015 (Vorlesungsfolien), Kapitel 5.1

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

Literaturverzeichnis IV

[14] 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

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

[16] 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

[17] 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

Literaturverzeichnis V

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

[19] 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

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

[21] Pouzon, L. :
The Origin of the Shell.
In: *Multics Home*.
Multicians, 2000, Kapitel `http://www.multicians.org/shell.html`

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

Literaturverzeichnis VI

[23] 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

[24] 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

[25] 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

[26] 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

Literaturverzeichnis VII

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

[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