— BS //—

Erscheinungsformen

Betriebssysteme, ©Wolfgang Schröder-Preikschat

Einteilung

- nach der **Entstehungsgeschichte** 2
- nach der **Betriebsart** 10
- nach dem **Zweck** 16

Einteilung nach der Entstehungsgeschichte

- manuelle Bestückung des Rechners („am Anfang war das Feuer“) 3
- automatische Bestückung des Rechners 4
- *off-line* Verarbeitung 5
- überlappte Ein-/Ausgabe 6
- überlappte Job-Verarbeitung 7
- Mehrprogrammbetrieb 8
- interaktive Job-Verarbeitung 9

Manuelle Bestückung des Rechners

- Operateure/Programmierer haben die volle Kontrolle über den Rechner
  1. Programmkarten in den Rechner einspeisen . . . . . *Lochkarten einlegen*
  2. Lochkartenleseprogramm eingeben ({Binär,Oktal,Hex}code) und starten
  3. Übersetzer (sofern erforderlich) starten
  4. Datenkarten in den Rechner einspeisen . . . . . *Lochkarten einlegen*
  5. übersetztes Programm starten
  6. Ergebnisse vom Lochkartenstanzer/Drucker abholen
- der bedingt erforderliche ({Fortran,Cobol}-)Übersetzer und sämtliche E/A-Prozeduren sind Bestandteil des Programmkartenstapels
- Schwachpunkt: ***Bedienung***, ***Mensch***

Automatische Bestückung des Rechners

- Durchsatzerhöhung durch Einschränkung/Reduzierung der manuellen Eingriffe
  1. Programmkarten in den Rechner einspeisen ..... *Lochkarten einlegen*
  2. Lochkartenleseprogramm starten
  3. Ergebnisse vom Lochkartenstanzer/Drucker abholen
- ein speicherresidentes Kontrollprogramm agiert als Kommandointerpreter
  - Kontrollkarten regeln den Ablauf..... *job control language*
  - das **embryonale Betriebssystem** besteht aus Lochkartenleseprogramm, Kommandointerpreter und den E/A-Prozeduren
- Schwachpunkte: ***langsame Peripherie***, ***sequentielle E/A***

Off-Line Betrieb

- E/A erfolgt von/auf **Magnetbänder** unterstützt durch **Satellitenrechner**
  - die unterschiedlichen Datenträger ziehen Konvertierungsabläufe nach sich:

Eingabe	→	Lochkarten-zu-Magnetband
Ausgabe	→	Magnetband-zu-Lochkarten
Drucken	→	Lochkarten-zu-Papier

  - der Magnetbandtransfer zwischen Rechner und Satellit erfolgt manuell
- das „embryonale Betriebssystem“ erhält ein neues internes Erscheinungsbild
  - Kontrolle von Bandmaschinen anstatt von Kartenlesern/-stanzern, Druckern
- Schwachpunkte: ***sequentieller Bandzugriff***, ***feste Jobfolge***

Überlappte Ein-/Ausgabe

- E/A-Geräte verfügen über „*direct memory access*“ (DMA)
  - d.h. unabhängig von der CPU arbeitende E/A-Kanäle
- eine asynchrone Unterbrechung (*interrupt*) meldet die E/A-Bereitschaft
  - E/A-Geräte zwingen die CPU zum Kontextwechsel:
    1. Sicherung des PC und Verzweigung an eine feste Speicheradresse
    2. Unterbrechungsbehandlung
    3. Rückkehr zum unterbrochenen Programm und gesicherten PC laden
  - im (embryonalen) Betriebssystem erwächst Synchronisationsbedarf
- Schwachpunkt: ***Lehrlauf*** beim Jobwechsel

Überlappte Job-Verarbeitung

- sequentielle Abarbeitung der Jobs während weitere eingelesen werden
  - die Trommel/Festplatte ersetzt das Magnetband
- das Betriebssystem erhält wahlfreien Zugriff auf die Jobs
  - die Vergabe/Zuteilung der CPU erfolgt nach einem (dynamischen) Fahrplan
  - die Einplanung (*scheduling*) der Jobs unterliegt einer bestimmten Strategie
- Schwachpunkt: ***Monopolisierung*** der CPU, ***Lehrlauf*** bei E/A

Mehrprogrammbetrieb

- die Fahrplanstrategie erfolgt streng betriebsmittelorientiert
  - Speicherplatzbedarf
  - Anzahl benötigter E/A-Kanäle
  - erwartete Laufzeit
- eine Duchsatzoptimierung (# Jobs pro Zeiteinheit) wird praktiziert
  - sofern möglich erfolgen Job-Wechsel bei E/A (d.h. bei Job-Wartephasen)
  - die CPU-Wartezeiten sind i.A. (weit) kürzer als die Job-Wartezeiten
- Schwachpunkt: ***Interaktionslosigkeit*** (*single-stream batch monitor*)

Interaktive Job-Verarbeitung

- Benutzern wird die Illusion einer eigenen (auch virtuellen) Maschine gegeben
  - den Programmen werden Zeitscheiben zur Ausführung zugeteilt
  - ein Zeitscheibenablauf bedeutet (ggf.) den Programmwechsel
  - die zyklische Zeitscheibenvergabe (*scheduling*) führt zum CPU-Multiplexing
- mehrere Benutzer können „gleichzeitig“ am Rechner arbeiten
  - ein Terminal ermöglicht die Interaktion mit dem laufenden Job bzw. Prozess
  - *multi-access*
- Schwachpunkte: . . . . . ***ffs.***

Einteilung nach der Betriebsart

- Stapelbetrieb (*batch processing*) 11
  - Routine
- Zeitscheibenbetrieb (*timesharing*) 12
  - Interaktion
- Echtzeitbetrieb (*real-time processing*) 13
  - Rechtzeitigkeit

Stapelbetrieb

- ist geprägt durch interaktionslose Programmabläufe (→ p. 8):
  1. Einspeisung des Auftrags (*job*)
  2. „aktive Pause“
  3. Inempfangnahme des Resultats
  4. ggf. Fehler korrigieren und zurück zu ⇒ 1.
- klassicher **Lochkartenbetrieb**, ist jedoch nicht darauf beschränkt (z.B. `sh(1)`)
  - Variante ist die dezentrale Job-Verarbeitung: *remote job entry* (RJE)
- geeignet zur Durchführung von Routine- und/oder rechenintensiven Aufgaben

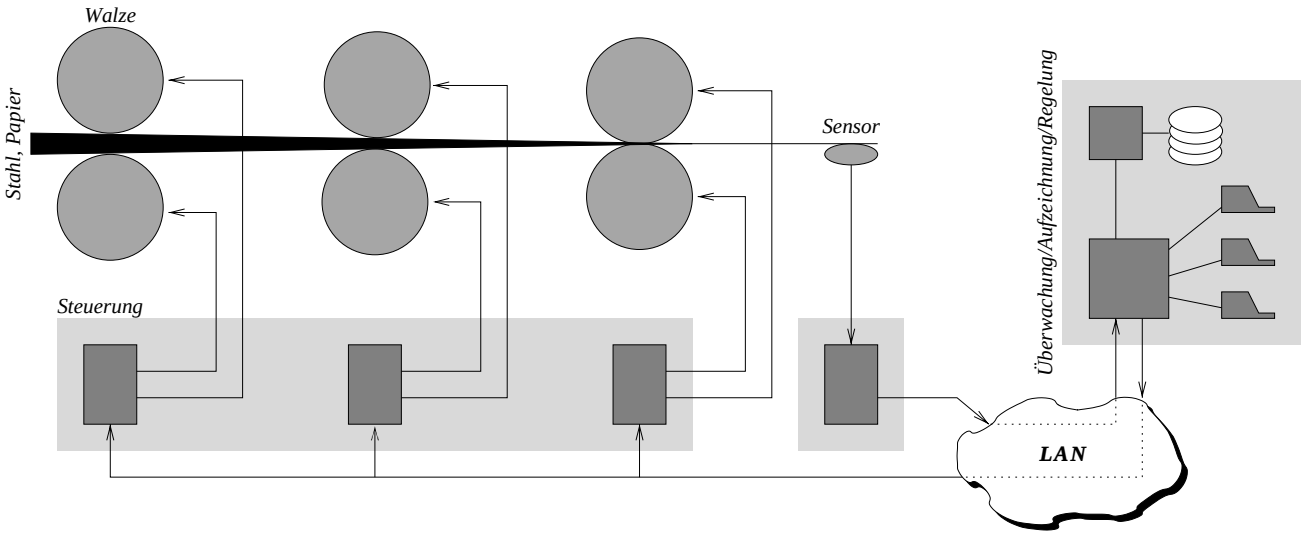
Zeitscheibenbetrieb

- ist geprägt durch interaktive Programmabläufe (→ p. 9)
  - eine pseudo-parallele (d.h. nebenläufige) Programmausführung findet statt
  - Mehrbenutzerfähigkeit ist nicht zwingend, wohl aber Mehrprozessfähigkeit
- die Kombination mit dem Stapelbetrieb ist durchaus üblich
  - Einspeisung von Aufträgen vom Arbeitsplatz aus
  - Verwendung von `sh(1)`-Skripten, z.B., anstatt von Lochkarten
- geeignet für's tägliche Geschäft (Softwareentwicklung, Textverarbeitung, . . .)

Echtzeitbetrieb

- ist geprägt durch rechtzeitige (vorhersagbare) Programmabläufe [1]
  - das Betriebssystem zeigt ein deterministisches Laufzeitverhalten
  - Schnelligkeit ist nicht (immer) entscheidend und ggf. sogar kritisch
  - von „externen Prozessen“ vorgegebene Zeitschranken sind einzuhalten
  - typisches Profil für Prozesskontrollsysteme bzw. **eingebettete Systeme**
- Einsatzszenario ist oft ein rückgekoppeltes (Sensor/Aktor-) System:
  1. Signale vom externen Prozess empfangen
  2. Reaktion zur Stabilisierung einleiten — *Regelung*
  3. Signale zum externen Prozess senden — *Steuerung*
- geeignet für die Steuerung externer/industrieller Prozesse

Prozesskontrollsystem — Walzwerk



Eingebettetes System — Automobil

Infotainment Audio, Video, Mobiltelefon, Internet, . . .

Komfortbereich Navigationssystem, Klimaanlage, Fensterheber, Schiebedach, „Luftsack“ (*airbag*), DWA, FIS, GPS, GRA, . . .

Motor- und Fahrwerksteuerung Einspritzanlage, Zündzeitpunktkontrolle, elektronisches Gaspedal, elektronische Kupplung und -Handbremse (*drive by wire*), ABS, APS, ASR, EBV, EDS, ESP, . . .

Sonstiges Lichtanlage, Blinker, Scheibenwischer, Intervallschaltung, . . .

Einteilung nach dem Zweck

- Spezialzwecksystem (*special-purpose system*) 17
  - Einbenutzersystem
  - Prozesskontrollsystem (→ p. 13)
  - Anfragesystem
  - Transaktionssystem

- Allgemeinzwecksystem (*general-purpose system*) 21

Spezialzwecksystem

- ist vorbereitet auf eine sehr eng gefasste Aufgabe — keine Eventualitäten
  - das System ist maßgeschneidert für ein klar spezifiziertes Einsatzgebiet
    - gilt oft nicht nur für die (System-) Software, sondern auch für die Hardware
    - oft bilden Hardware, Software und Umgebung eine geschlossene Einheit
  - die Systembestandteile sind im hohen Maße auf einander abgestimmt

- ist optimiert hinsichtlich des speziellen Anwendungsfalls
  - auf Kosten aller von diesem „Standard“ abweichenden Fälle

Spezialzwecksystem — Einbenutzersystem

- bietet zu einem Zeitpunkt eine virtuelle Maschine für nur einen Benutzer an
  - sinnvoll dort, wo sich die gemeinsame Nutzung der Hardware nicht rechnet
  - ein sehr weit verbreiteter Ansatz in der „PC-Kommune“, trotz Linux
- Einbenutzerfähigkeit bedeutet nicht Ein{prozess,adressraum}fähigkeit
  - das System kann mehrere (logische, virtuelle) Adressräume verwalten
  - pro Adressraum können mehrere Prozesse (Fäden, *threads*) ablaufen
- bekannte Beispiele: MS-DOS, Windows $\leq$ NT, OS/2, UCSD-Pascal, Oberon

Spezialzwecksystem — Anfragesystem

- eine enorm große Datenmenge (Datenbank) ist für Suchanfragen zu verwalten
  - die Antworten auf die Suchanfragen sollen (möglichst) in Echtzeit geschehen
  - Modifikation und Aktualisierung der Datenbank müssen einhergehen können
  - typische Vertreter sind Management oder medizinische Informationssysteme
- Zugriffe müssen ohne Wissen von der internen Organisationsstruktur erfolgen
  - Abstraktion von Softwarestrukturen und Hardwarekomponenten ist gefordert
  - bis hin zu Transparenz(en) in vernetzten/verteilten Umgebungen
- heute oft realisiert als spezielle Schicht oberhalb eines Allgemeinzwecksystems

Spezialzwecksystem Transaktionssystem

- dem Anfragesystem ähnlich, jedoch im Lichte hochfrequenter Modifikationen
  - konkurrenter Zugriff auf die Datenbank durch sehr viele Benutzer
  - vielfache Änderungen in Sekunden(-bruchteilen) möglich
  - jederzeit muss die Konsistenz der Daten gesichert sein
- die nebenläufigen bzw. konkurrenten Zugriffe sind zu koordinieren
  - eine große Herausforderung im Falle verteilter (heterogener) Datenbanken
  - das System muss adäquate Mechanismen zur Koordination bereit stellen
- heute oft realisiert als spezielle Schicht oberhalb eines Allgemeinzwecksystems

Allgemeinzwecksystem

- ist vorbereitet auf alle Eventualitäten — „Eier legende Wollmilchsau“

$$\text{erzwingt z.B.} \left\{ \begin{array}{l} \textit{Scheduling} \\ \textit{Schutz} \\ \textit{Sicherheit} \end{array} \right\} \text{in einer Ein-} \left\{ \begin{array}{l} \text{prozess} \\ \text{programm} \\ \text{benutzer} \end{array} \right\} \text{-umgebung}$$

- ist optimiert hinsichtlich des meist wahrscheinlichen Anwendungsfalls
  - auf Kosten aller von diesem „Standard“ abweichenden Fälle

Nachwort

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**. [2]

Referenzen

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

[2] A. M. Lister and R. D. Eager. *Fundamentals of Operating Systems*. The Macmillan Press Ltd., fifth edition, 1993. ISBN 0-333-59848-2.

[3] W. Schröder-Preikschat. *The Logical Design of Parallel Operating Systems*. Prentice Hall International, 1994. ISBN 0-13-183369-3.