Teil VI

Intermezzo

Überblick

Lernziele und Lehrinhalte

Grundlagen von Betriebssystemen

Vorgänge innerhalb von Rechensystemen ganzheitlich verstehen

- Zusammenspiel $\left\{ \begin{array}{rcl} \text{Hardware} & \leftrightarrow & \text{Software} \\ & \vdots & \\ \text{Anwendung} & \leftrightarrow & \text{Betriebssystem} \\ & \vdots & \\ \text{Treiber} & \leftrightarrow & \text{Gerät} \end{array} \right\}$ begreifen

Grundzüge der imperativen Systemprogrammierung (in C) selbst erleben

- im Kleinen praktizieren ☞ Dienstprogramme
- im Großen erfahren ☞ Betriebssysteme

Motivation

Rückgrat eines jeden Rechensystems

Betriebssysteme sind unerlässliches Handwerkszeug der Informatik

nicht alle müssen ein solches Handwerkszeug bauen/pflegen können

alle müssen jedoch mit dem Begriff/Produkt umgehen können

Betriebssysteme zu verstehen hilft, Phänomene zu begreifen

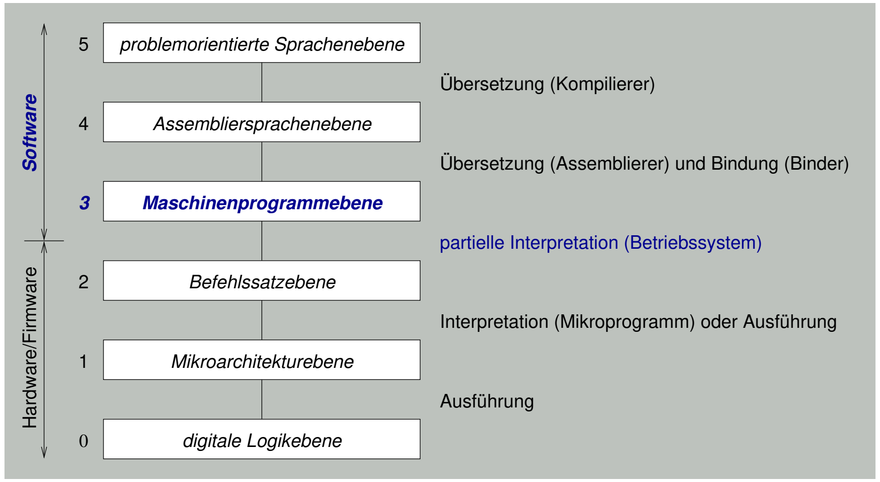
- unterschiedliches Systemverhalten erklären zu können
- Eigenschaften und Fehlern auseinanderhalten zu können

Betriebssysteme müssen immer im Anwendungskontext beurteilt werden

- kein einzelnes System ist für alle möglichen Zwecke optimal geeignet

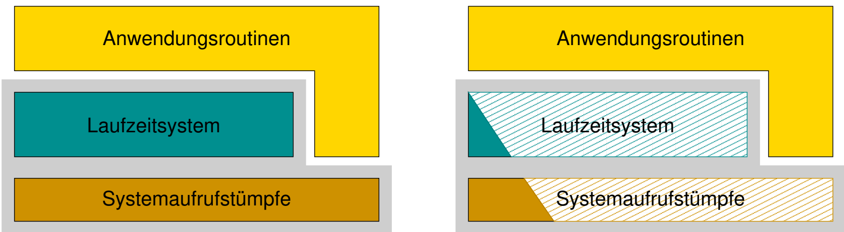
Strukturierte Organisation von Rechensystemen

Betriebssystem als abstrakter Prozessor für Programme der Ebene $_3$



Organisation von Maschinenprogrammen

Logische Struktur: statische vs. dynamische Bibliotheken



statische Bibliothek

- Laufzeit einsparen
- vollständige Programme
- Binder & Lader

dynamische Bibliothek

- Hintergrundspeicher einsparen
- unvollständige Programme
- bindender Lader

Unterbrechungen und Ausnahmesituationen

Teilinterpretation

Programmunterbrechungen zeigen Ausnahmebedingungen an und bedeuten die partielle Interpretation von Maschinenprogrammen:

Trap synchron, vorhersagbar, reproduzierbar

Interrupt asynchron, unvorhersagbar, nicht reproduzierbar

- macht determinierte Programme nicht-deterministisch
- Nebenläufigkeit, kritischer Abschnitt

Ausnahmebehandlung bringt Kontextwechsel mit sich, die abrupte Zustandswechsel das ausführenden Prozessors bewirken:

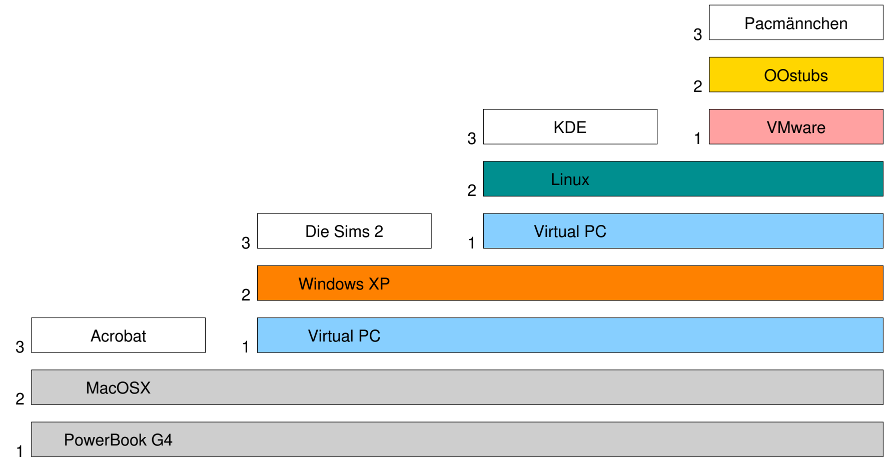
- vom unterbrochenen Programm zum behandelnden Programm ↓BS
- vom behandelnden Programm zum unterbrochenen Programm BS↑

Hardware und Software sind (funktional) äquivalent: Emulation

- die Nachahmung der Eigenschaften von Hardware durch Software

Hierarchie virtueller Maschinen

Selbstvirtualisierung durch Teilinterpretation



Stapelbetrieb
Stapelsysteme

abgesetzter Betrieb Satellitenrechner, Hauptrechner
- Entlastung durch Spezialrechner

überlappte Ein-/Ausgabe DMA, *Interrupts*
- nebenläufige Programmausführung

überlappte Auftragsverarbeitung Einplanung, Vorgriff
- Verarbeitungsstrom von Aufträgen

abgesetzte Ein-/Ausgabe *Spooling*
- Entkopplung durch Pufferbereiche

Mehrprogrammbetrieb *Multiprogramming*
- Multiplexen der CPU

☞ programmiertes dynamisches Laden von Überlagerungen (*Overlays*)

Mehrzugangsbetrieb
Interaktive Systeme

Dialogbetrieb Dialogstationen
- mehrere Benutzer gleichzeitig bedienen können

Hintergrundbetrieb Mischbetrieb
- Programme im Vordergrund starten

Teilnehmerbetrieb Zeitscheibe, *Timesharing*
- eigene Dialogprozesse absetzen können

Teilhaberbetrieb residente Dialogprozesse
- sich gemeinsame Dialogprozesse teilen können

Multiprozessorbetrieb Parallelrechner, SMP
- Parallelverarbeitung von Programmen

☞ Umlagerung (*Swapping*) kompletter Programme und virtueller Speicher

Netzbetrieb
Verteilte Systeme

- als Einzelsystem präsentierter Zusammenschluss vernetzter Rechner
- Virtualisierung der im Rechnerverbund verfügbaren Betriebsmittel

Fehlerverarbeitung erkennen, maskieren, tolerien
- Ausfälle vernetzter Rechensysteme sind partiell

Transparenz insb. Netzwerktransparenz
- Zugriffs- und Ortstransparenz: Prozedurfernaufruf

Sicherheit Vertraulichkeit, Integrität, Verfügbarkeit
- Eigenwert von Informationen sichern

☞ Diensteschicht (*Middleware*) zum netzwerkzentrischen Rechnerbetrieb

Integrationsbetrieb
Eingebettete Systeme

Jedes in einem Produkt versteckte Rechensystem, wobei das Produkt selbst jedoch kein Rechner ist:
- Haushaltsgeräte, Audio-/Videogeräte, (Mobil-) Telefone, ...
- Fahrzeuge (Schiff, Bahn, Auto) und Flugzeuge

Spezialzweckbetrieb: Betriebssystem und Anwendungsprogramm(e) sind mit- bzw. ineinander (in funktionaler Hinsicht) verwoben

$$\left\{\begin{array}{rl} \text{[verteiltes]} & \textit{grid} \\ \text{[durchdringendes]} & \textit{pervasive} \\ \text{[allgegenwärtiges]} & \textit{ubiquitous} \end{array}\right\} \textit{computing} \Longrightarrow \textit{ambient intelligence}$$

Echtzeitbetrieb

Zeitabhängige Systeme

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

weich auch „schwach“ . . . . *soft*

- Terminverletzung ist tolerierbar

fest auch „stark“ . . . . *firm*

- Terminverletzung ist tolerierbar, führt zum Arbeitsabbruch

hart auch „strikt“ . . . . *hard*

- Terminverletzung ist keinesfalls tolerierbar

☞ querschneidender Belang der gesamten Systemsoftware + Anwendung

Adressraum

Ausführungs- und Schutzdomäne von Programmen

physikalischer Adressraum (Hardware) . . . . Ebene $_2$

- ist durch die jeweils gegebene Hardwarekonfiguration definiert
- nicht alle Adressen sind gültig, zur Programmspeicherung verwendbar

logischer Adressraum (Kompilierer, Binder, Betriebssystem) . . . Ebene $_{5/4/3}$

- abstrahiert von Aufbau/Struktur des Haupt- bzw. Arbeitsspeichers
- alle Adressen sind gültig und zur Programmspeicherung verwendbar

virtueller Adressraum (Betriebssystem) . . . . Ebene $_3$

- auf Vorder- und Hintergrundspeicher abgebildeter log. Adressraum
- erlaubt die Ausführung unvollständig im RAM liegender Programme

Speicher

Zusammenspiel aneinander angepasster Funktionen zu gegenseitigem Nutzen

Laufzeitsystem (bzw. Bibliotheksebene) verwaltet den lokal vorrätigen Speicher eines logischen/virtuellen Adressraums

- Speicherblöcke können von sehr feinkörniger Struktur/Größe sein
  - einzelne Bytes bzw. Verbundobjekte
- Verfahrensweisen orientieren sich (mehr) an Programmiersprachen

Betriebssystem verwaltet den global vorrätigen Speicher (d.h. den bestückten RAM-Bereich) des physikalischen Adressraums

- Speicherblöcke sind üblicherweise von grobkörniger Struktur/Größe
  - z.B. eine Vielfaches von Seiten
- Verfahrensweisen fokussieren auf Benutzer- bzw. Systemkriterien

Datei

Abstraktion von Informationen (über-) tragenden Betriebsmitteln

Aufbewahrungsmittel für zu speichernde Informationen

- kurz-, mittel-, langfristige Speicherung
- bleibende Speicherung (persistente Daten)

Kommunikationsmittel für kooperierende Prozesse

- gemeinsamer (externer) Speicher
- Röhre zum Informationsaustausch

Abstraktionsmittel für den Betriebsmittelzugang

Hardware CPU, RAM, Peripherie, . . .
Software Adressräume, Prozessinstanzen, . . .

Abbildung symbolische Adresse $\mapsto$ numerische Adresse:

- einen Dateinamen auf auf eine Dateikopfnummer abbilden
- ein Dateiverzeichnis (auch) als Umsetzungstabelle verstehen

Namensraum
Namen Kontexte zuordnen

Namensräumen eine Struktur aufprägen und dadurch einem Namen in „benutzerfreundlicher Weise“ eine eindeutige Bedeutung geben können:

flache Struktur eines einzigen Kontextes

hierarchische Struktur mehrerer Kontexte (d.h. flacher Strukturen)
- hierarchischer Namensraum
  - Pfadnamen zur Navigation im Namensraum
  - spezielle Kontexte (UNIX-ähnlicher Systeme)
  - Bindung und Auflösung von Namen
- Hierarchie von Namensräumen
  - montiereren von Dateisystemen

Dateisysteme und Namensräume sind (logisch) zwei verschiedene Dinge:
- das eine organisiert den Hintergrundspeicher (zur Dateiablage)
- das andere dient der Identifikation von Objekten (nicht nur Dateien)

Prozess
Abstraktes Gebilde vs. Identität einer Programmausführung

Gewichtsklasse eine Frage der Isolation von Adressräumen

Federgewicht keine Isolation
- der „reine“ Kontrollfluss: Faden

Leichtgewicht vertikale Isolation
- vom Betriebssystemadressraum

Schwergewicht horizontale Isolation
- von anderen Programmadressräumen

Einplanung Reihenfolgen festlegen, Aufträge sortieren
- Ablaufplan zur Betriebsmittelzuteilung erstellen
- Ablaufzustände von Prozessen fortschreiben

☞ charakteristische Eigenschaften der Einplanung/Einlastung von UNIX

Koordinationsmittel
Sequentialisierung nicht-sequentieller Programme

Semaphor abstrakter Datentyp zur Signalisierung von Ereignissen
- unteilbare Operationen auf eine Koordinationsvariable
  - P und V manipulieren eine nicht-negative ganze Zahl
- zur blockierenden Synchronisation gleichzeitiger Prozesse

Botschaft Synchronisation kombiniert mit Datentransfer
- Primitiven (Semantiken) zum Botschaftenaustausch
  - {*no-wait*, *synchronization*, *remote-invocation*} *send*
- Rollenspiele bei der Interprozesskommunikation
  - gleich- vs. ungleichberechtigte Kommunikation
- Kommunikationsendpunktadressen und Verbindungen

Betriebsmittel vs. synchrone/asynchrone bzw. blockierende IPC:

konsumierbares Betriebsmittel Nachricht (bzw. Botschaft)

wiederverwendbares Betriebsmittel Nachrichtenpuffer

Vertiefung
Ausgewählte Kapitel der Systemprogrammierung

Prozessverwaltung KW 24 (≈ 2 Termine)
- Einplanung (klassisch, Fallstudien), Einlastung

Koordination KW 25 (≈ 2 Termine)
- ein-/mehrseitig, blockierend/nicht-blockierend

Speicherverwaltung KW 26 (≈ 2 Termine)
- Disziplinen, MMU (Pentium), virtueller Speicher

Sicherheit KW 27 (≈ 1 Termin)
- Befähigung, Zugriffskontrollliste

Dateiverwaltung KW 28 (≈ 2 Termine)
- Organisation des Hintergrundspeichers, RAID

...und falls noch Zeit übrig geblieben sein sollte:

Architektur KW 28 ($\approx \frac{1}{2}$ Termin)
- Monolith, geschichtetes System, Minimalkerne