Systemnahe Programmierung in C (SPiC)

17 Betriebssysteme

Jürgen Kleinöder, Daniel Lohmann, Volkmar Sieh

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2019

http://www4.cs.fau.de/Lehre/SS19/V_SPIC

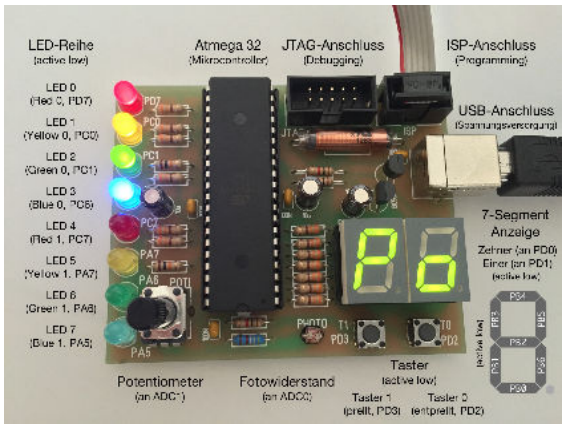
Definition „Betriebssystem”

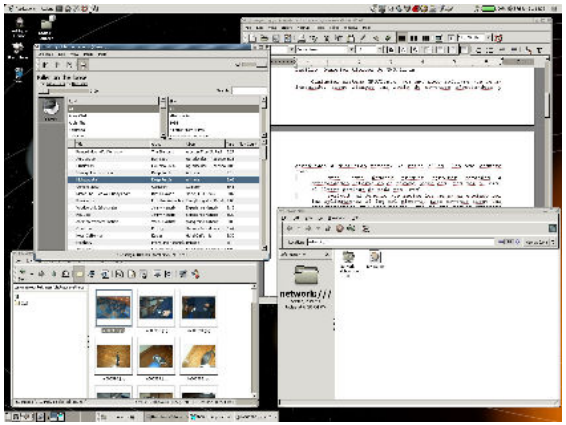
- DIN 44300
  - „... die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und die insbesondere die Abwicklung von Programmen steuern und überwachen.”
- Andy Tannenbaum
  - „... eine Software-Schicht ..., die alle Teile eines Systems verwaltet und dem Benutzer eine Schnittstelle oder eine virtuelle Maschine anbietet, die einfacher zu verstehen und zu programmieren ist [als die nackte Hardware].”
- Zusammenfassung:
  - Software zur Verwaltung und Virtualisierung der Hardware-Komponenten (Betriebsmittel)
  - Programm zur Steuerung und Überwachung anderer Programme

Überblick

Bisher:

- **Ein** Programm, dass
- **alleine**,
- **beim Booten** gestartet
- mit **Hardware-Zugriffen**
- seine Umgebung steuert.





Quelle: www.wikipedia.org

Jetzt:

- **Mehrere** Programme, die
- **nebenläufig**,
- **dynamisch** gestartet/beendet
- über **definierte E-/A-Funktionen**
- ihre Umgebung steuern.

Multitasking

Existiert mehr als eine Anwendung auf einem System („Multitasking”),

- müssen sich die Anwendungen abstimmen,
  - wer wann die/eine CPU bekommt,
  - wer welche Speicherbereiche verwenden darf,
  - wer welche Plattenblöcke verwenden darf,
  - wer welchen Teil des Bildschirms beschreiben darf,
  - ...

  Da keine Anwendung für sich allein z.B. entscheiden kann, welche Speicherbereiche noch ungenutzt sind, muss es **gemeinsam benutzte Methoden und Zustandsvariablen** geben.
- muss sichergestellt sein, dass
  - sich alle Anwendungen an die Abmachungen halten (auch die versehentlich/absichtlich fehlerhaft programmierten!)

  Hardware-Erweiterungen müssen Zugriffe auf unerlaubte Speicherbereiche bzw. E-/A-Geräte verhindern.

Betriebssystem-Kern

- „**gemeinsam benutzte Methoden und Zustandsvariablen**”
  - **Betriebssystem-Kern** („Kern”, „System-Kern”, „Kernel”)
- „**Hardware-Erweiterungen**”
  - **Priviligierungsstufen** („Privilege Level”, „Ringe”)
  - **Speicherschutz** („Memory Management Unit” („MMU”))

Priviligierungsstufen

- unpriviligierte Ebene
  („Anwendungsebene”, „User-Ebene”, „User-Ring”)
  - darf „normale” CPU-Instruktionen ausführen,
  - darf auf den ihr zugewiesenen Speicher zugreifen,
  - darf Betriebssystem-Methoden aufrufen
- priviligierte Ebene
  („System-Ebene”, „Kern-Ebene”, „Ring 0”)
  - darf alle CPU-Instruktionen ausführen,
  - darf auf jeden Speicherbereich zugreifen,
  - darf Speicherschutz umkonfigurieren,
  - darf auf E-/A-Geräte zugreifen

Wechsel in die priviligierte Ebene durch

- System-Aufrufe („System Calls”, „Traps”)
- Unterbrechnungen („Interrupts”)
- Ausnahmen („Exceptions”)

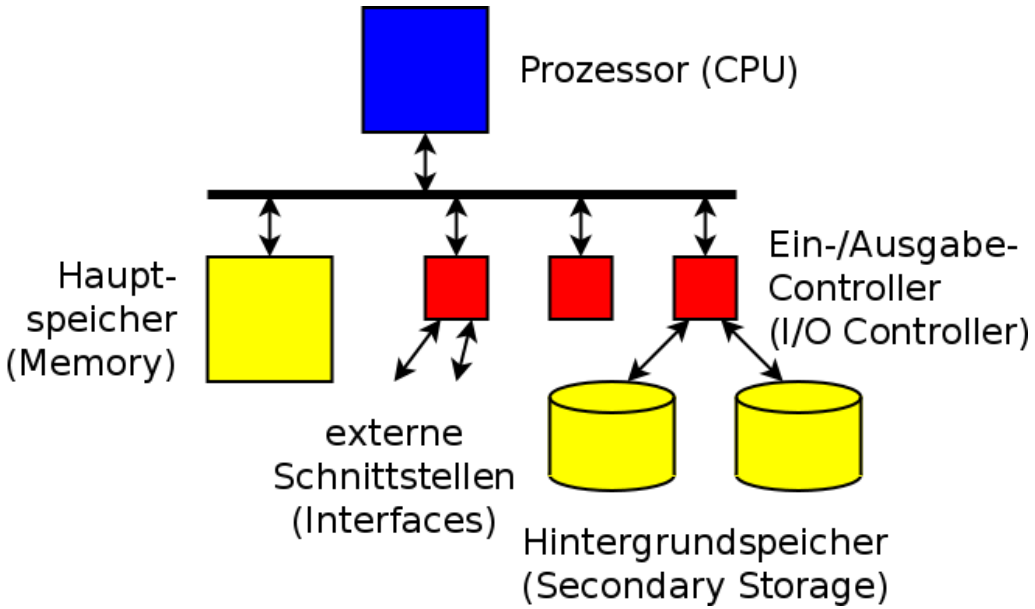
System-Aufrufe

Beispiel: Anwendung braucht mehr Speicher
Schritte:

- Anwendung berechnet, wieviel Speicher sie benötigt,
- legt Parameter in CPU-Registern ab,
- wechselt mit spezieller CPU-Instruktion in den Kern,
(=> ab jetzt priviligiert!)
- liest Parameter aus CPU-Registern,
- reserviert Speicher für sich,
- programmiert MMU um,
- legt Ergebnis in CPU-Registern ab,
- wechselt mit spezieller CPU-Instruktion zurück in Anwender-Ebene,
(=> ab jetzt wieder unpriviligiert!)
- und holt Ergebnis aus CPU-Register.

Verwaltung von Betriebsmitteln

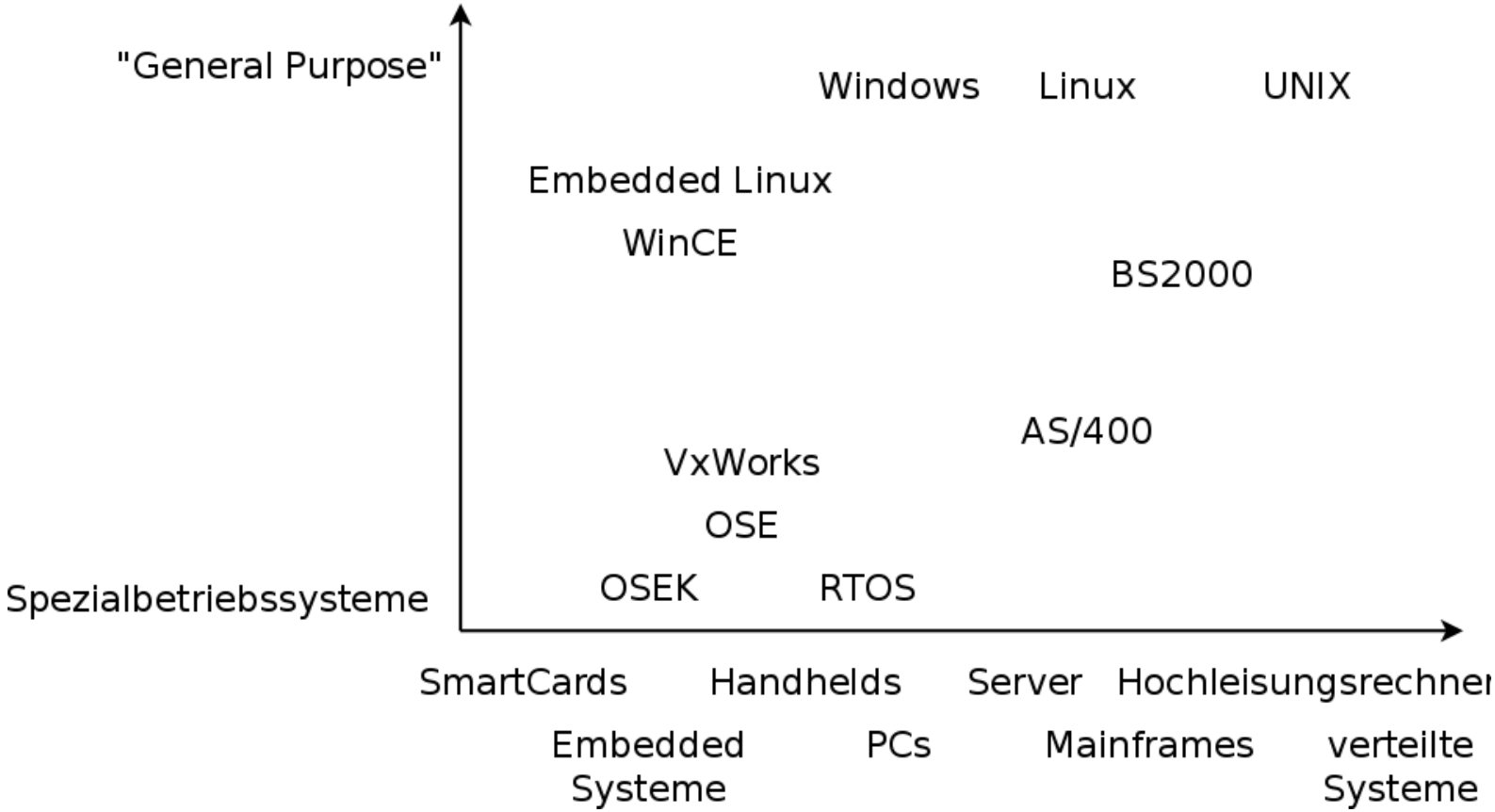
- Aufgaben des Betriebssystem-Kerns
  - Multiplexen von Betriebsmitteln für mehrere Benutzer bzw. Anwendungen
  - Schaffung von Schutzumgebungen
  - Bereitstellen von Abstraktionen zur besseren Handhabbarkeit der Betriebsmittel



- Ermöglichen einer koordinierten gemeinsamen Nutzung von Betriebsmitteln, klassifizierbar in
  - aktive, zeitlich aufteilbare (Prozessor)
  - passive, nur exklusiv nutzbare (periphere Geräte, z.B. Drucker u.Ä.)
  - passive, räumlich aufteilbare (Speicher, Plattenspeicher u.Ä.)
- Unterstützung bei der Fehlererholung

Klassifikation von Betriebssystemen

- Unterschiedliche Klassifikationskriterien
  - Zielplattform
  - Einsatzzweck
  - Funktionalität



Klassifikation von Betriebssystemen (2)

- Wenigen „General Purpose”-, Mainframe- und Höchstleistungsrechner-Betriebssystemen steht eine Vielzahl kleiner und kleinster Spezialbetriebssysteme gegenüber:

  C51, C166, C251, CMX RTOS, C-Smart/Raven, eCos, eRTOS, Embos, Ercos, Euros Plus, Hi Ross, Hynet-OS, LynxOS, MicroX/OS-II, Nucleus, OS-9, OSE, OSEK Flex, OSEK Turbo, OSEK Plus, OSEKtime, Pricise/MQX, Pricise/RTCS, proOSEK, SOS, PXROS, QNX, Realos, RTMOSxx, Real Time Architect, ThreadX, RTA, RTX51, RTX251, RTX166, RTXC, Softune, SSXS RTOS, VRTX, VxWorks, ...

  Einsatzbereich: Eingebettete Systeme, häufig Echtzeit-Betriebssysteme, über 50% proprietäte (in-house) Lösungen

- Alternative Klassifikation: nach Architektur

Betriebssystem-Architekturen

- Umfang: zehntausende bis mehrere Millionen Befehlszeilen
  => Strukturierung hilfreich
- Verschiedene Strukturkonzepte
  - Laufzeitbibliotheken (minimal, vor allem im Embedded-Bereich)
  - monolithische Systeme
  - geschichtete Systeme
  - Minimalkerne
- Unterschiedliche Schutzkonzepte
  - kein Schutz
  - Schutz des Betriebssystems
  - Schutz des Betriebssystems und Anwendungen untereinander
  - feingranularer Schutz auch innerhalb von Anwendungen

Betriebssystem-Komponenten

- Speicherverwaltung
  - Wer darf Informationen wohin im Speicher ablegen?
- Prozessverwaltung
  - Wann wird welche Aufgabe bearbeitet?
- Dateisystem
  - Speicherung und Schutz von Langzeitdaten
- Interprozesskommunikation
  - Kommunikation zwischen verschiedenen Anwendungen bzw. parallel ablaufenden Anwendungsteilen
- Ein-/Ausgabe
  - Kommunikation mit der „Außenwelt” (Benutzer/Rechner)