Betriebssysteme I

Claudius Schnörr

Hochschule München
FK 7: Informatik und Mathematik

Unterlagen zur Vorlesung

Anmerkungen zur Vorlesung

- Die Vorlesung wird folienbasiert gehalten
- Ankündigungen sowie Praktikums- und Abgabetermine unter
  http://schnoerr.userweb.mwn.de/Vorlesungen/...
  von allen Teilnehmern bitte die emails!
- Praktika:
  - Praktika sind Zulassungsvoraussetzung zur Prüfung
  - werden in Python gehalten
  - Abgaben im Praktikum an angegebenen Terminen
  - 2 Gruppen, Gruppeneinteilung hier
- Besprechungstermine:
  - in der Vorlesung / im Praktikum
  - ansonsten: direkt nach der Vorlesung
- Prüfung:
  - schriftlich, 90 Minuten
  - keine Hilfsmittel (außer Taschenrechner)

Voraussetzungen

- Rechnerarchitektur + Grundlagen d. Informatik:
  - grober Aufbau eines Computers:
    - CPU
    - Hauptspeicher
    - Peripherie
    - Systembus
- Umgang mit einem Linux-Rechner:
  - bash
  - Start / Stop von Programmen
  - cp, mkdir, cd, etc...
- Skriptsprache Python:
  - eigenständige Einarbeitung hier vorgesehen
  - Auffinden der Dokumentationen
  - Start/Stop eines Programms
  - einfache Programmstrukturen: Schleife, Funktion, if, usw.

- Warum Python ?
  - sehr gute objektorientierte Sprache:
    - vielseitig
    - weit verbreitet
    - gut dokumentiert
    - einfach zu erlernen
    - <-> perl (kryptisch)
  - kein Compiler notwendig
  - kann Konstrukte unter
    - ksh
    - bash
    - perl

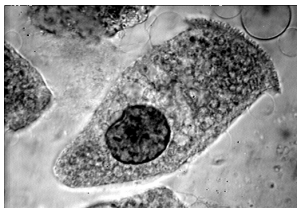
    ersetzen !

Zielsetzungen

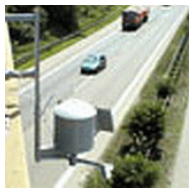
- Nicht angestrebt:
  - VHS-Kurs zur Bedienung eines Rechners
    „nun auf den Button „XYZ“ klicken, dann ...“
  - interne Spezialitäten (Kernel-Entwicklung, Device-Treiber, etc.)
  - Beispiele meist in Linux
- Sondern die Grundlagen eines Betriebssystems:
  - Verständnis der Konzepte und Mechanismen
    - wie wirken diese ?
    - welche Probleme und Lösungsmöglichkeiten gibt es ?
    - welche Konsequenzen ergeben sich für den Anwendungsentwickler ?
  - neben kleinen praktischen Tips
- Grundlage für andere Vorlesungen

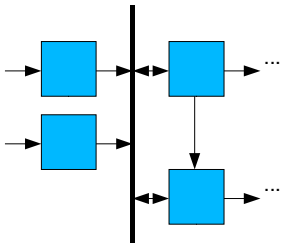
zu meiner Person (1)

- Studium der Elektrotechnik / Nachrichtentechnik an der TH Karlsruhe
- Promotion an der Univ. Stuttgart in der Bildverarbeitung und Mustererkennung
- Beispiel für ein industrielles Umfeld:
  Systementwicklung in der Verkehrsanalyse
  - Signalverarbeitung:
    - Analyse von Verkehrsdaten und
    - Synthese zu Verkehrsmeldungen und Reisezeitschätzungen
  - C++ - Entwicklung von Client-Serveranwendungen:
    - unter Solaris
    - mit Hochverfügbarkeitsanforderungen
    - mit SmartSockets als Middleware



Flächendeckende Verkehrinformationen





Detektorsysteme:
- IR-Detektoren, asynchron, ~4000
- Induktionsschleifen, synchron, ~7000
- Floating-Car Daten (FCD), ~1000

Meßgrößen:
- Geschwindigkeiten
- Flußmessungen

Telegrammaufkommen:
- ~40 Mio/Tag ~ 500/Sekunde

Entwicklungs- und Produktionsumgebung:
- Solaris / Sun
- Hochverfügbarkeitssystem
- C++, Java, Multithreaded Applications

Middleware:
- SmartSockets / Talarian
- Sockets
- RPC
- (CORBA)

zu meiner Person (2)

- seit 2004 an der HM
- Aufbau des Masters Informatik
- Lehrangebot:
  - Master Informatik:
    - Bildverarbeitung,
    - Mustererkennung,
    - 3D-Rekonstruktion,
    - Bildfolgenauswertung und Bewegungssehen
    - Hauptseminar
  - Bachelor Informatik:
    - Seminar Bildverarbeitung
    - Bildverarbeitung u. Computergrafik (nur BV)
    - Betriebssysteme
- Forschungsinteressen:
  - Gründung der Forschungsschwerpunktes CORSNAV
    Computer Vision - Remote Sensing - Navigation
  - http://schnoerr.userweb.mwn.de/F+E-Projekte
  - www.hm.edu/corsnav
- Studentische Mitwirkungsmöglichkeiten:
  - anspruchsvolle Bachelor- / Masterarbeiten in CORSNAV
  - SHK-Anstellungen
  - Teilzeit-Anstellungen parallel zum Master

Motivation

Einleitende Fragestellungen:

- Prozesse und Threads:
  - was ist der Unterschied ?
  - wie kann man Prozesse und Threads nutzen, was ist dabei zu beachten ?
- Welche Eingriffsmöglichkeiten hat man bein Scheduling ?
- Welche Formen der Interprozeßkommunikation gibt es ?
  - Einordnung und Unterschiede ?
  - Eignung für welche Aufgaben ?
- Welche Standard-Synchronisationsmechanismen gibt es ?
- Wie kann man in eigenen Anwendungen die Gefahr von Deadlocks mindern ?

Gliederung

1. **Einführung und Übersicht**
2. Prozesse und Threads
3. Interrupts
4. Scheduling
5. Synchronisation
6. Interprozesskommunikation
7. Speicherverwaltung

Übersicht: Prozesse und Threads

Prozess:

- Programm, das in den Speicher geladen wurde und ausgeführt wird / werden soll:
  - eigene Daten
  - Programmkode
  - Stack
  - Programmzähler
  - ...

Thread:

- ähnlich wie Prozess, aber
  - mehrere Threads greifen auf gleichen Speicher zu
  - Thread-Verwaltung nicht unbedingt im Kernel
  - User-Level / Kernel-Level
  - müssen sinnvoll synchronisiert werden

Übersicht: Interrupts | Scheduling

Verschiedene Interrupt-Typen:

- Hardware-Interrupts
- Software-Interrupts (Traps)
- Exceptions, z.B.
  - Division 1/0
  - Zugriff auf falsche Adresse
- Interrupt-Handler

Scheduler:

- Rechenzeit an Prozese verteilen
- Prinzipien:
  - präemptiv
  - kooperativ
- Verfahren:
  - Round Robin
  - Priority
  - Shortest Job First
  - ...
- Was passiert beim Prozesswechsel ?

Übersicht: Synchronisation | Interprozesskommunikation

Synchronisation

- Parallelverarbeitung koordinieren
  - parallele Threads/Prozesse
  - Zugriff auf gemeinsame Daten
  - Race-Conditions, kritische Abschnitte, gegenseitiger Ausschluß
- Synchronisationsmethoden:
  - Mutex
  - Semaphor
  - Events
  - Signale
  - Locking
  - Monitor

Interprozess-Kommunikation:

- Mehrere Prozesse arbeiten gemeinsam an einer Aufgabe
  - Austausch von Informationen
- Prinzipien:
  - synchron / asynchron
  - verbindungorientiert / multicast / broadcast
- IPC-Techniken:
  - Pipes (Pipelines)
  - Shared-Memory
  - Nachrichtenversand
  - Sockets

Übersicht: Speicherverwaltung

Speicherverwaltung

- Effiziente Nutzung und Verwaltung verfügbarer Speicherressourcen:
  - Zuteilung zwischen mehreren Prozessen
  - virtueller Adressraum
    - wie setzt sich eine Speicheradresse zusammen
  - Auslagerung auf externe Datenträger
    - Paging
    - Swapping
  - ...

Einführung

und

Übersicht

Einführung und Übersicht

Übersicht:

- Beschreibung und Einordnung
- Aufgaben eines Betriebssystems
- Arten von Betriebssystemen
- (historische Entwicklung)
- Zentrale Konzepte

Beschreibung

Was ist ein Betriebssystem (BS) ?

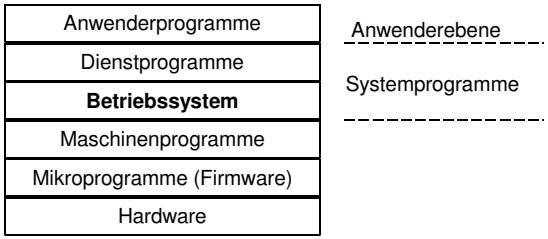
- **Aus Systemsicht:**
  - Bezeichnung für alle Programme, die
    - die Ausführung der Benutzerprogramme,
    - die Verteilung der Betriebsmittel (z.B. Speicher, Prozessor, Dateien, Netzwerk, Drucker)
    - die Aufrechterhaltung der Betriebsart (z.B. Batch, Timesharing, Echtzeit)

      steuern und überwachen. (Duden Informatik)
- **Aus Anwendersicht:**
  - Das BS stellt dem Anwender die Sicht einer virtuellen Maschine zur Verfügung, die einfacher zu benutzen ist als die reale Hardware, z.B.
    - steht der Rechner einem Anwender scheinbar alleine zur Verfügung
    - können Programme einfach auf Geräte zugreifen,
      - z.B. unterschiedlicher Hersteller oder Kategorien

Einordnung

BS als Mittler zwischen Anwendungen und Hardware

- Schichtenmodell eines Computers:



  - Hardware: Register, Speicher, System- und Adressbus, Peripherie
  - Mikroprogramme: Interpretation der Maschinenbefehle durch Signalfolgen (meist in einem ROM)
  - Systemprogramme:
    - Dienstprogramme: Editor, Compiler, Linker, Shell
    - Betriebssystem(kern)

Aufgaben eines Betriebssystems (1)

Was soll ein Betriebssystem leisten ?

Rechner sind trotz ähnlicher Architektur im Detail sehr unterschiedlich, z.B. Einteilung des Adressraums (Speicher, E/A-Controller)

- **Abstraktionsschicht zwischen Hardware und Programmen:**
  - BS realisiert eine virtuelle Maschine mit virtuellem Adressraum
  - BS realisiert einheitliche Sicht für Anwendungen auf Geräteklassen, z.B.
    - Datenträger (Platte, DVD, USB-Stick, Netzlaufwerk, ...)
    - Drucker (PostScript-Laser, Tintenstrahler, „File“-Drucker)
    - BS findet freie Sektoren auf einer Platte, verwaltet diese, usw.
- **Verwaltung von Betriebsmitteln (Ressourcen):**
  - alles, was Anwendungen brauchen, z.B.
    - CPU-Rechenzeit,
    - Speicher,
    - Gerätezugriffe (Speicher, Platte, Netzwerk),

Aufgaben eines Betriebssystems (2)

Was soll ein Betriebssystem leisten ?

- **Schutz der Hardware vor direkten Zugriffen**
  - z.B. vor defekter Software, anderen Prozessen oder Anwendern
- **Zulassen und Abgrenzung mehrerer Anwender** (Multi-User-Betrieb)
- **Parallelbetrieb mehrere Anwendungen** (Multitasking): Faire Aufteilung der Ressourcen
- **Virtualisierung des Speichers**
  - Anwendungen müssen nicht wissen, wo sie im Speicher liegen
  - Speicher über phys. RAM hinaus verfügbar

Arten von Betriebssystemen

- **Mainframe-BS**
  - schnelle E/A, viele Prozesse, Transaktionen
- **Server-BS**
  - Viele Anwender gleichzeitig, Netzwerkanbindung
- **Multiprozessor-BS**
  - für Parallelrechner
- **Echtzeit-BS**
  - Reaktion auf äußere Signale innerhalb fest vorgegebener Zeit
- uvm

==> die Grenzen sind fließend
==> die grundlegenden Konzepte und Kenntnisse sind vielfach nützlich

Historische Entwicklung

Unterhaltsam, aber nicht wichtig
=> Kurz die persönliche Erfahrung des Dozenten:

- Apple-OS, Atari (1980-1990)
  - single-User single Task
  - eigentlich kein BS
- VMS (1988-1993)
  - ohne Grafik, reine Terminals
  - Multi-User
  - preemtives Multi-Tasking
- MacOS (altes) (1990-1997)
  - single-User
  - kooperatives Multitasking
  - gute Abstraktion (GUI, Schnittstellen zur Hardware, etc.)
- Unix / Irix und Solaris (1997-2004)
- Linux und etwas Windows

Zentrale Konzepte (1)

- **Ein BS verwaltet Systemressourcen:**
  - Hauptspeicher (Memory Management)
  - Anfragen externer Geräte (I/O-System)
  - organisiert Daten in Dateien auf externen Speichergeräten (file-System)
- **... führt Anwenderprogramme aus:**
  - es verwaltet Prozesse und Threads (die Ausführungsumgebung eines Programms),
  - bestimmt die Reihenfolge und Dauer, die Programme laufen (Scheduling)
  - bietet Programmen Services über Systemaufrufe (System Calls) an, z.B. Synchronisationsmechanismen, etc.
- **... behandelt auftretende Interrupts ausgelöst durch**
  - Hardware-Komponenten (z.B. Systemuhr, I/O) (Hardware-Interrupts)
  - Programme (Software Interrupts, Exceptions)
- Nicht zum eigentlichen Betriebssystem (Betriebssystemkern) gehören z.B.:
  - Shell, GUI, Editor, Compiler, Linker, etc.

Zentrale Konzepte (2)

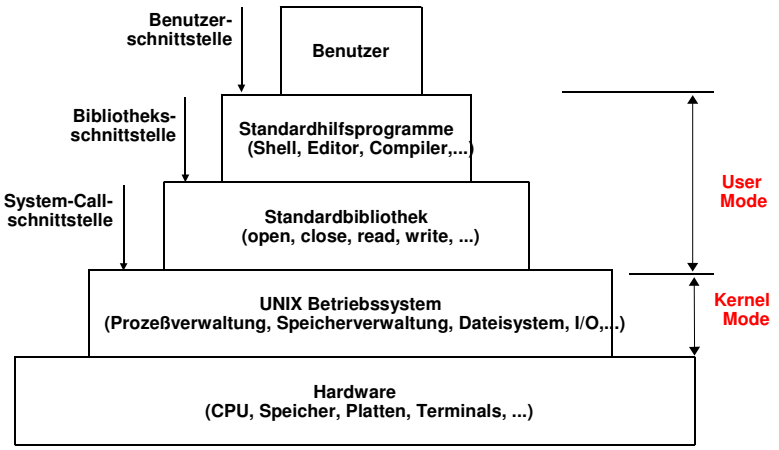
User Mode und Kernel Mode

- **Prozessoren unterscheiden zwischen der Ausführung von Kode im**
  - **User Mode:** Anwenderprogramme und Teile des BS
  - **Kernel Mode:** die meisten Teile des BS

  Ein Bit in der CPU entscheidet, in welchem Modus der Prozessor arbeitet
- **Zwei Unterschiede bei der Kode-Ausführung:**
  - Ausführung einiger Maschinenbefehle nur im Kernel-Mode möglich, z.B.
    - aktivieren/deaktivieren von Interrupts
    - Schreiben des CPU-Statusregisters
  - Für jede Speicherseite (Page) bestimmt ein Protection-Kode, welcher Zugriff erlaubt ist
- **Ein Wechsel in den Kernel-Mode ist nur möglich über**
  - Aufruf eines System Calls
  - Auslösen eines Interrupts

Zentrale Konzepte (3)

Illustration am Beispiel von Unix:



Literatur

[1] W. Stallings, Operating Systems Internals and Design Principles, Prentice Hall

[2] Silberschatz, Galvin, Gagne, Operating System Concepts, John Wiley
(enthält Beispiele für Unix, Windows und andere)

[3] A.S. Tanenbaum, Modern Operating Systems, Prentice Hall

[4] Linux Kernel 2.4 Internals, Kap.2, http://www.faqs.org/docs/kernel_2_4/lki-2.html

[5] J.Quade, E.-K. Kunst: „Linux-Treiber entwickeln“, http://ezs.kr.hsnr.de/Treiberbuch/html/

Ansonsten:

- Paralelle Vorlesung von Prof. C. Vogt:
  - Wahlmöglichkeiten bitte selbst abklären
  - http://www.cs.fhm.edu/~vogt/os/os_index.htm
  - dort auch beispielhafte Prüfungsfragen
- Folien und Material von Herrn Eßer: http://hm.hgesser.de/bs-ss2011/