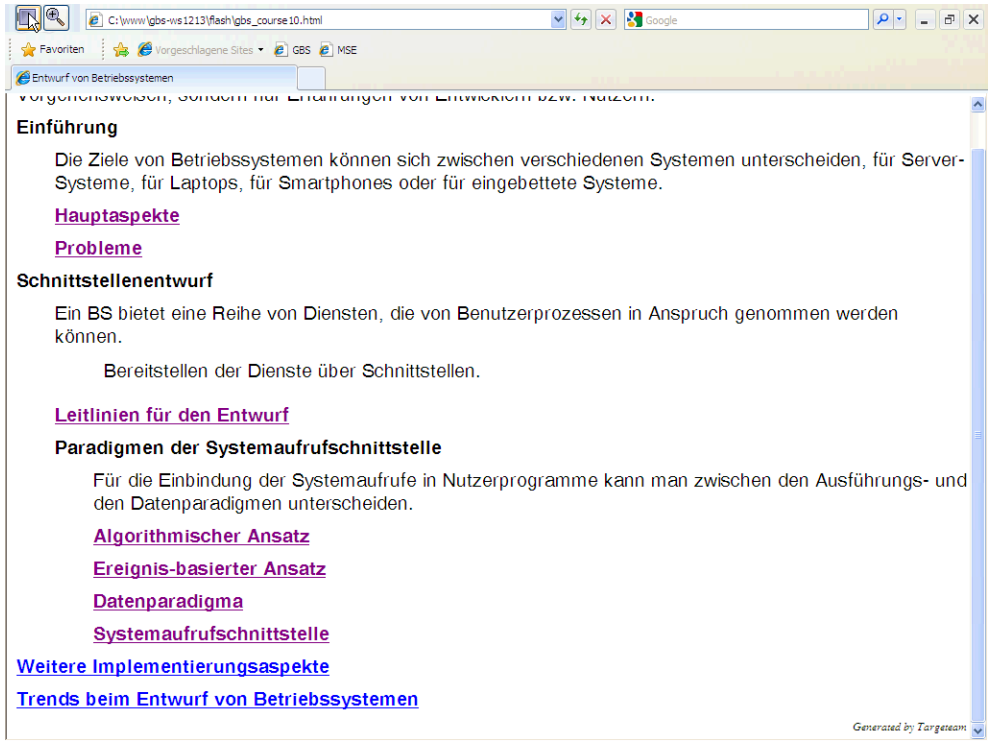
Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (06.02.2013)

Date: Wed Feb 06 13:16:00 CET 2013

Duration: 45:17 min

Pages: 13

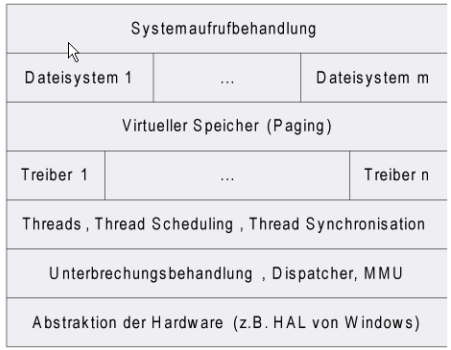


Neben den Diensten und deren Bereitstellung über die Systemaufrufschnittstelle stellen die folgenden Implementierungsaspekte eine wichtige Rolle beim Entwurf von Betriebssystemen.

Architektur
Mechanismen vs. Policies
Namensräume
Statische - Dynamische Datenstrukturen
Verbergen der Hardware
Speicherplatz vs. Laufzeit

Generated by Targeteam

Ein etablierter und erprobter Architekturansatz sind geschichtete Systeme.



Generated by Targeteam

Trennung von Mechanismen und Policies.

Beispiel: BS unterstützt Prioritätenbasierten Scheduler

Mechanismus: Feld gemäß absteigender Priorität sortiert; Scheduler durchsucht Feld beginnend von der höchsten Priorität

Policy: Festlegen der Prioritäten

Beispiel: BS unterstützt Paging für Arbeitsspeicher

Mechanismus: Verwaltung der Seiten-Kacheltabelle, MMU-Management, Funktionen zum Transport der Seiten zwischen Festplatte und Arbeitsspeicher

Policy: welche Seiten werden ausgelagert bei Seitenfehler

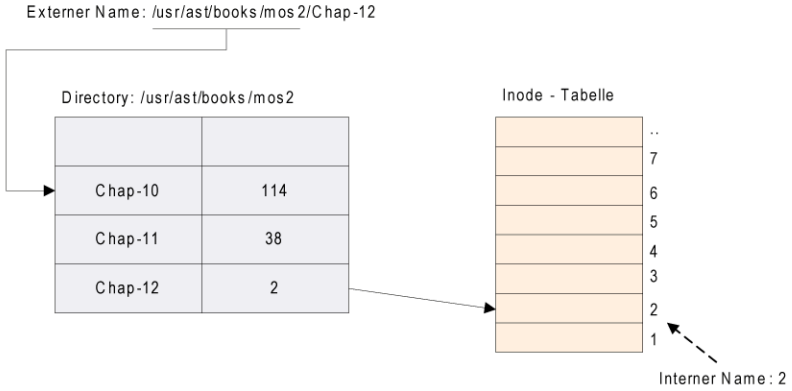
Namensräume

Die meisten langlebigen Datenstrukturen eines BS haben einen Namen oder einen Bezeichner zugeordnet, z.B. Login-Namen, Dateinamen, Gerätenamen, Prozess-ID etc. Namen werden auf 2 Ebenen vergeben

extern: Namen, welche Menschen verwenden (z.B. Dateiname)

intern: Namen, welche das System verwendet (z.B. inode Nummer)

Directories bilden externe Namen auf interne Namen ab.



Statische - Dynamische Datenstrukturen

Statische Strukturen sind einfacher und schneller, jedoch weniger flexibel als dynamische Strukturen

Beispiel Prozesstabelle:

Statisch: feste Prozesstabelle mit 256 Einträgen.

Dynamisch: Prozesstabelle als verkettete Liste.

Schnelle Suche in statischer Prozesstabelle.

Statische Tabellen sind vernünftig

Wenn die Zahl der Einträge sehr gut vorausgesagt werden kann.

Die unterste Schicht, die Hardware Abstraction Schicht (z.B. HAL von Windows) versucht Hardware-spezifische Eigenschaften zu verbergen

Abfrage der Arbeitsspeichergröße zum Bootzeitpunkt

Speicherung in einer Variablen

CPU-abhängige bedingte Übersetzung.

```
#include "config.h"
init()
{
    #if (CPU == PENTIUM) /* Pentium initialization here */ #endif
    #if (CPU == ULTRASPARC) /* ULTRASPARC initialization here */ #endif
... }
```

Hardware-Architekturen unterstützen unterschiedliche WortlängenCPU-abhängige bedingte Übersetzung.

```
#include "config.h"
{
    #if (WORD_LENGTH == 32) typedef int Register; #endif
    #if (WORD_LENGTH == 64) typedef long Register; #endif
Register R0, R1, ...;
```

Bei der Realisierung von Algorithmen besteht ein Abwägen zwischen Speicherplatz und Laufzeit

Nutzung von Prozeduren ⇒ Aufwand für Prozeduraufruf

Nutzung von Makros ⇒ direkte Einbindung; kein Prozeduraufruf

Beispiel: ***Prozedur*** , um die Bits mit Wert 1 in einem Byte zu zählen.

```
#define BYTE_SIZE 8 /* a byte contains 8 bits*/
int bit_count (int byte) {
    int i, count=0;
    for (i=0; i<BYTE_SIZE; i++) if((byte >> i) &1) count++;
    return(count);
}
```

Beispiel: ***Macro*** , um die Bits mit Wert 1 in einem Byte zu zählen.

Shiften des Arguments, Ausmaskieren aller Bits außer dem niederwertigen Bit, Aufaddieren der 8 Terme

```
#define bit_count(b) (b&1) + ((b>>1)&1) + ((b>>2)&1) + ((b>>3)&1) +
((b>>4)&1) + ((b>>5)&1) + ((b>>6)&1) + ((b>>7)&1)
```

Beispiel: ***Macro*** , um die Bits mit Wert 1 in einem Byte zu zählen.

Wert des Bytes ist Index in eine Tabelle mit 256 Einträge

Eintrag gibt die Anzahl der Bits von Wert 1 an

Speicherplatz vs. Laufzeit

Nutzung von Makros ⇒ direkte Einbindung; kein Prozeduraufruf

Beispiel: ***Prozedur*** , um die Bits mit Wert 1 in einem Byte zu zählen.

```
#define BYTE_SIZE 8 /* a byte contains 8 bits*/
int bit_count (int byte) {
    int i, count=0;
    for (i=0; i<BYTE_SIZE; i++) if((byte >> i) &1) count++;
    return(count);
}
```

Beispiel: ***Macro*** , um die Bits mit Wert 1 in einem Byte zu zählen.

Shiften des Arguments, Ausmaskieren aller Bits außer dem niederwertigen Bit, Aufaddieren der 8 Terme

```
#define bit_count(b) (b&1) + ((b>>1)&1) + ((b>>2)&1) + ((b>>3)&1) +
((b>>4)&1) + ((b>>5)&1) + ((b>>6)&1) + ((b>>7)&1)
```

Beispiel: ***Macro*** , um die Bits mit Wert 1 in einem Byte zu zählen.

Wert des Bytes ist Index in eine Tabelle mit 256 Einträge

Eintrag gibt die Anzahl der Bits von Wert 1 an

```
char bits[256] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4,1, ...};
#define bit_count(b) (int) bits[b];
```


Trends beim Entwurf von Betriebssystemen

Bedingt durch den vielfältigen Einsatz der Rechner sind die jeweiligen Trends von unterschiedlicher Bedeutung

BS mit großem Adressraum

Übergang von 32 Bit zu 64 Bit-Adressräumen

Adressraum von $2 * 10^{19}$ Bytes

Netze werden die Grundlage für BS

Zugriff auf Web ist vollständig und nahtlos in BS integriert

Bessere Unterstützung von parallelen und verteilten Systemen.

BS für batteriebetriebene Computer (PDAs, Smart Phone, etc.)

Energiemanagement; Adaption von BS und Anwendungen je nach Energieversorgung

Trend hin zu "Green Computing".

BS für eingebettete Systeme

Die Bezeichnung "***Eingebettetes System*** " bezieht sich auf die Nutzung von Computer Hardware und Software in einem Produkt. Unterscheidung nach Anwendungsdomäne

Automobil, z.B. Zündung, Motorkontrolle, Bremssystem

Consumer Elektronik, z.B. Set-top Box, Küchengeräte, Kamera, Smartphone

Medizinbereich, z.B. Dialysegerät, Infusionspumpe

Bürobereich, z.B. Faxgerät, Drucker

industrielle Nutzung, z.B. Roboter und Kontrollsysteme für Produktion

Eingebettete Systeme sind eng gekoppelt mit dem Produkt, wobei die Anforderungen sehr variieren können

Größe des Produkts, in das es eingebettet ist.

Lebensdauer.

Nutzungsumgebung.

Realzeitverhalten.

Es wurden eine Vielzahl von speziellen Betriebssystemen für eingebettete Systeme realisiert

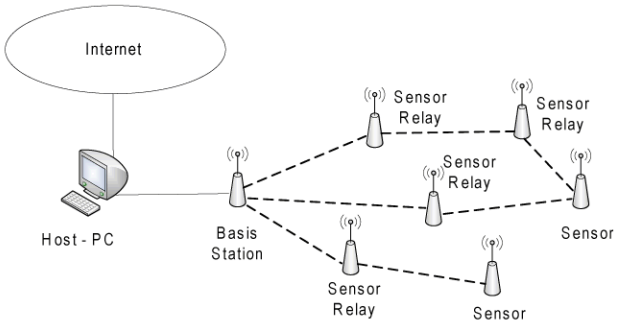
für mobile Endgeräte: Symbian OS, Windows CE oder Android von Google.

Android Systemarchitektur basiert auf Linux Kernel (Treiber, Speicher-/Prozessmanagement); darauf existieren Bibliotheken (z.B. für Graphik), eine Laufzeitumgebung und ein Anwendungsframework.

TinyOS

für drahtlose Sensornetze: TinyOS , entwickelt von der UC Berkeley

zentraler Teil des BS ist sehr klein, ca. 400 Bytes Code und Datenspeicher



TinyOS hat keinen BS-Kern und keinen Speicherschutzmechanismus.

komponenten-basierter Ansatz und ereignis-basiertes Ausführungsmodell.

geschichtete Anordnung der Komponenten; zwischen Komponenten bestehen Aufrufbeziehungen; elementare Komponenten kapseln Hardware Komponenten (z.B. Timer).

Anwendungen werden durch Verknüpfung von Komponenten realisiert.

2 Arten von Komponenten: a) Module = implementieren Funktionen, und b) Konfigurationen = beschreiben die Verknüpfung von Komponenten.

eine Komponente implementiert eine oder mehrere "tasks".

Zusammenfassung

Diese Vorlesung beschäftigte sich mit den technischen Aspekten von Rechensystemen. Es gab eine Einführung in Betriebssysteme und systemnahe Programmierung. Insbesondere wurden folgende Aspekte behandelt:

Modellierung von Prozessen, z.B. Petrinetze, sowie die Synchronisation von Prozessen beim Zugriff auf gemeinsame Ressourcen.

Verwaltung von Prozessen und deren Zuteilung an die CPU, um sie auszuführen.

Verwaltung des Arbeitsspeichers aus der Sicht des Betriebssystems (virtuelle Speicherverwaltung, Seitenadressierung).

persistente Speicherung von Information in Dateien.

Prozesskommunikation in lokalen und verteilten Systemen.

Verwaltung der Geräte.

Sicherheit in Rechensystemen.

Der Entwurf und die Implementierung von Betriebssystemen ist komplex und aufwändig. Wichtige Aspekte sind

definierte Abstraktionen, z.B. Prozesse, Threads, Adressraum

Definition von Schnittstellen zur Bereitstellung von Operationen, z.B. Operation zur Synchronisation, persistente Speicherung von Information

Schnittstelle sollte einfach, vollständig und effizient sein.

Sicherstellen der Abgrenzung, z.B. Isolieren von Fehlern.

Verwalten der Hardware, z.B. Geräteverwaltung.

Festlegung der Paradigmen, z.B. Ausführungsparadigma, Systemaufrufschnittstelle, Speicherplatz vs. Laufzeit