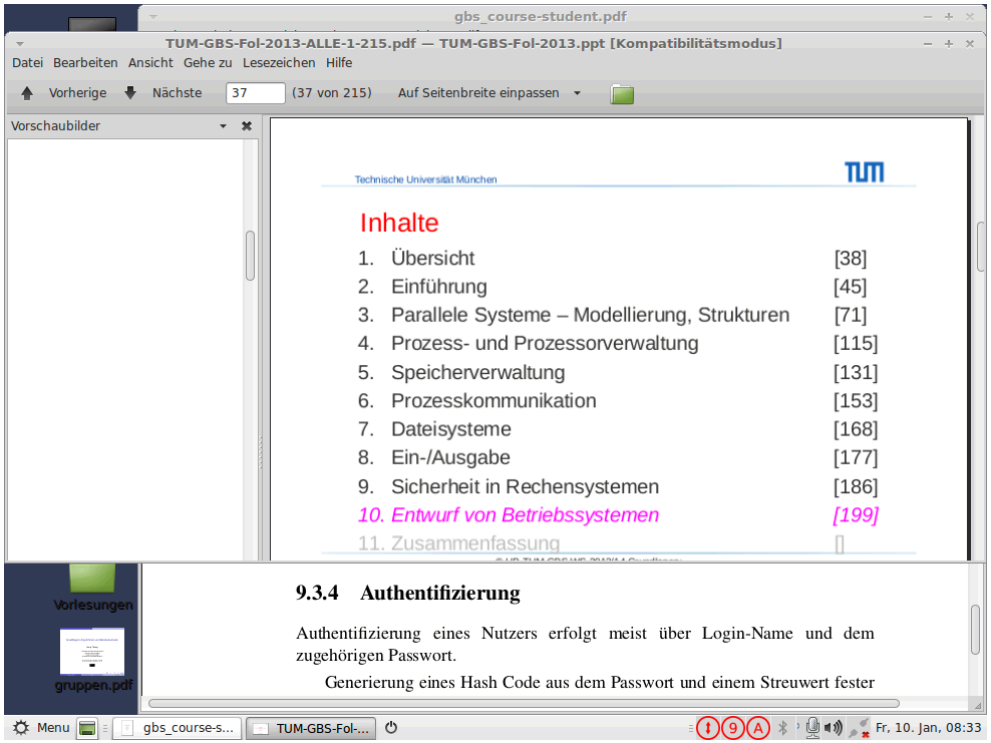
Script **generated by TTT**

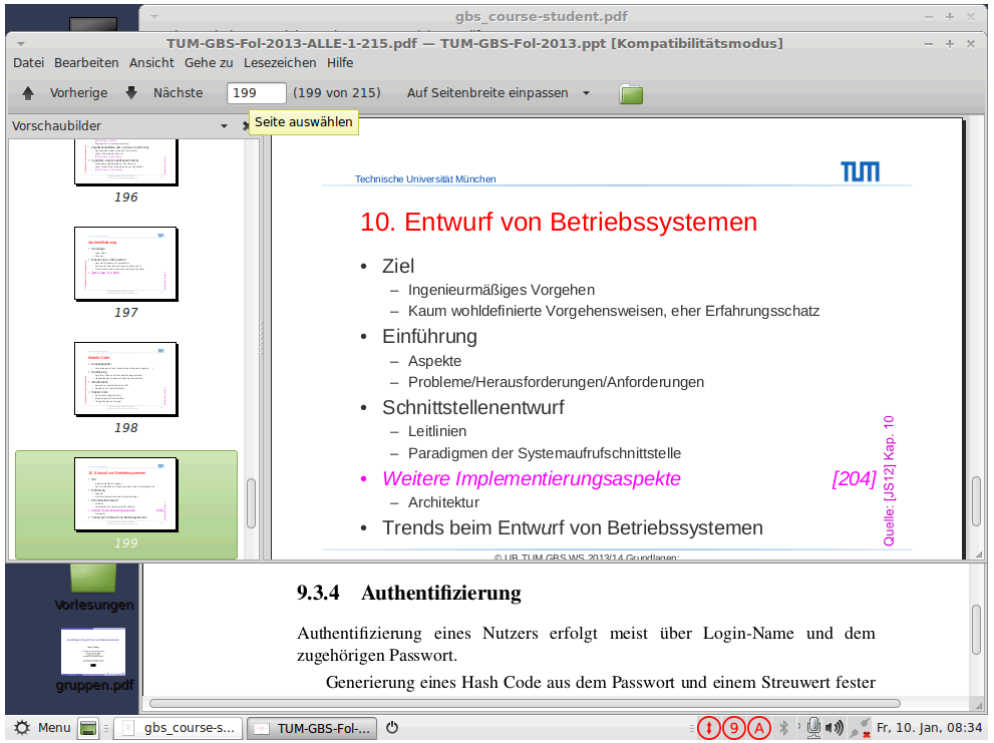
Title: Baumgarten: GBS (10.01.2014)

Date: Fri Jan 10 08:32:45 CET 2014

Duration: 56:50 min

Pages: 24





Weitere Implementierungsaspekte (2)

- Namensräume
  - Namen für langlebige „Datenstrukturen“ (Objekte)
  - Ebenen-Organisation
    - externe Namen (z.B. Dateiname)
    - interne Namen (z.B. inode Nummer)
  - Vergleiche: „Adresse“ vs. „Bezeichner“ vs. „Identifikator“
- Statische vs. Dynamische Datenstrukturen
  - Statisch
    - Vorteile: einfacher, schneller zugreifbar
  - Dynamisch
    - Vorteile: flexibler
  - Abwägung nach Einsatzzweck

Weitere Implementierungsaspekte (3)

- Verbergen der Hardware
  - Z.B.: HAL (Hardware Abstraction Layer)
  - Abfrage der AS-Größe zum Bootzeitpunkt
  - CPU-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
  - Architektur-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
- Speicherplatz vs. Laufzeit
  - Nutzung von Prozeduren (Funktionen) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 217]

Technische Universität München

Weitere Implementierungsaspekte (3)

ASM

- Verbergen der Hardware
  - Z.B.: HAL (Hardware Abstraction Layer)
  - Abfrage der AS-Größe zum Bootzeitpunkt
  - CPU-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
  - Architektur-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
- Speicherplatz vs. Laufzeit
  - Nutzung von Prozeduren (Funktionen) [JS12, Kap. 10, p. 216]

10. Entwurf von Betriebssystemen

Kap. 10

Vorlesungen

gruppen.pdf

9.3.4 Authentifizierung

Authentifizierung eines Nutzers erfolgt meist über Login-Name und dem zugehörigen Passwort.

Generierung eines Hash Code aus dem Passwort und einem Streuwert fester

Technische Universität München

Weitere Implementierungsaspekte (3)

- Verbergen der Hardware
  - Z.B.: HAL (Hardware Abstraction Layer)
  - Abfrage der AS-Größe zum Bootzeitpunkt
  - CPU-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
  - Architektur-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
- Speicherplatz vs. Laufzeit
  - Nutzung von Prozeduren (Funktionen) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 217]

10. Entwurf von Betriebssystemen

Quelle: [JS12] Kap. 10

© UB TUM GBS WS 2013/14 Grundlagen: Betriebssysteme und Systemsoftware (IN0009) 206

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-215.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 206 (206 von 215) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München

Weitere Implementierungsaspekte (3)

- Verbergen der Hardware
  - Z.B.: HAL (Hardware Abstraction Layer)
  - Abfrage der AS-Größe zum Bootzeitpunkt
  - CPU-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
  - Architektur-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
- Speicherplatz vs. Laufzeit
  - Nutzung von Prozeduren (Funktionen) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 217]

10. Entwurf von Betriebssystemen

Quelle: [JS12] Kap. 10

Register R0, R1, ...;

10.3.6 Speicherplatz vs. Laufzeit

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

Vorlesungen

gruppen.pdf

Menu gbs_course-s... TUM-GBS-Fol-... Fr, 10. Jan, 08:42

---

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 216 (223 von 228) 150%

Vorschaubilder

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, ...;
```

10.3.6 Speicherplatz vs. Laufzeit

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

Vorlesungen

gruppen.pdf

Menu gbs_course-s... TUM-GBS-Fol-... Fr, 10. Jan, 08:42

---

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 216 (223 von 228) 150%

Vorschaubilder

10.3.6 Speicherplatz vs. Laufzeit

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)
```


Vorlesungen

gruppen.pdf

Menu gbs_course-s... TUM-GBS-Fol-... Fr, 10. Jan, 08:43

---

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 217 (224 von 228) 150%

Vorschaubilder

- 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];
```

10.4 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

Vorlesungen

gruppen.pdf

Menu gbs_course-s... TUM-GBS-Fol-... Fr, 10. Jan, 08:44

Weitere Implementierungsaspekte (3)

- Verbergen der Hardware
  - Z.B.: HAL (Hardware Abstraction Layer)
  - Abfrage der AS-Größe zum Bootzeitpunkt
  - CPU-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
  - Architektur-abhängige Übersetzung
    - [JS12, Kap. 10, p. 216]
- Speicherplatz vs. Laufzeit
  - Nutzung von Prozeduren (Funktionen) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 216]
  - Nutzung von Makros (Shiften und Addieren) [JS12, Kap. 10, p. 217]

Trends beim Entwurf

- BS mit großem Adressraum
  - 32 Bit-Architekturen vs. 64 Bit-Architekturen
- Netze werden die Grundlage für BS
  - Web-Zugriff nahtlos integriert
- Bessere Unterstützung paralleler und verteilter Systeme
- Energie als knappe Ressource
  - Energiemanagement
  - Green Computing
- BS für eingebettete Systeme
  - Anwendungsbereiche
  - Beispiele

Eingebettete Systeme

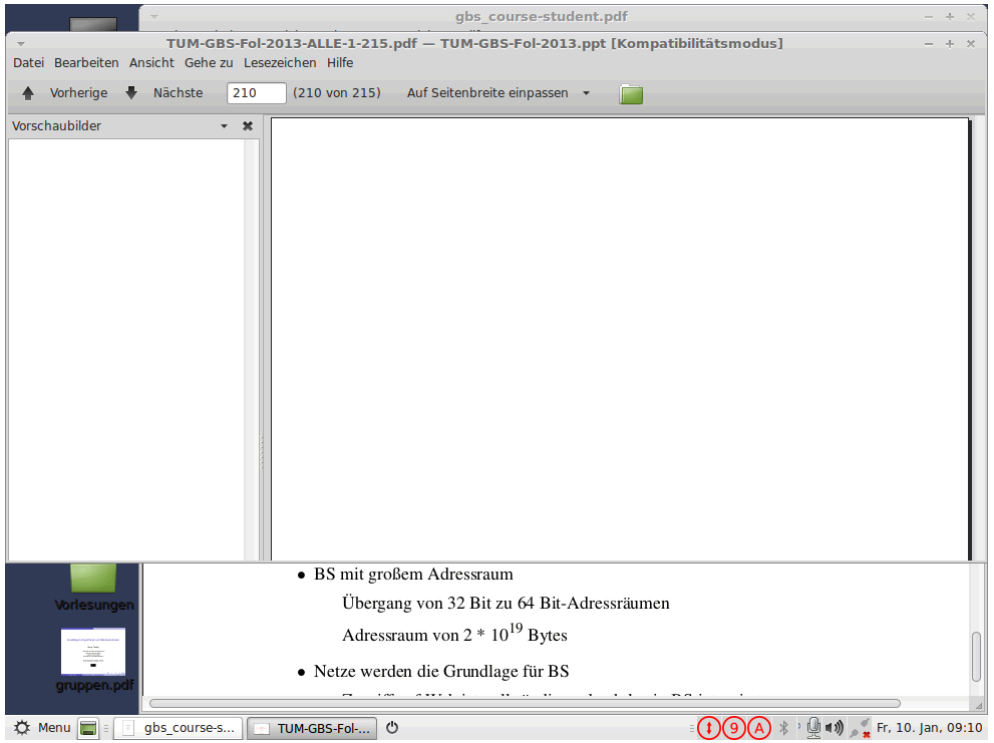
- Anwendungsdomänen
  - Automotive
    - z.B. Motorsteuerung, Bremssysteme, Assistenz
  - Luft- und Raumfahrt
    - Flugzeugsteuerung, Satelliten
  - Consumer Electronics
    - z.B. Set-top-Box, Küchengeräte, Kamera, Smartphone
  - Medizintechnik
    - z.B. Dialysegerät, Infusionspumpe
  - Bürobereich
    - z.B. Faxgerät, Scanner, Drucker
  - Industrielle Nutzung
    - Roboter und Kontrollsysteme für die Produktion

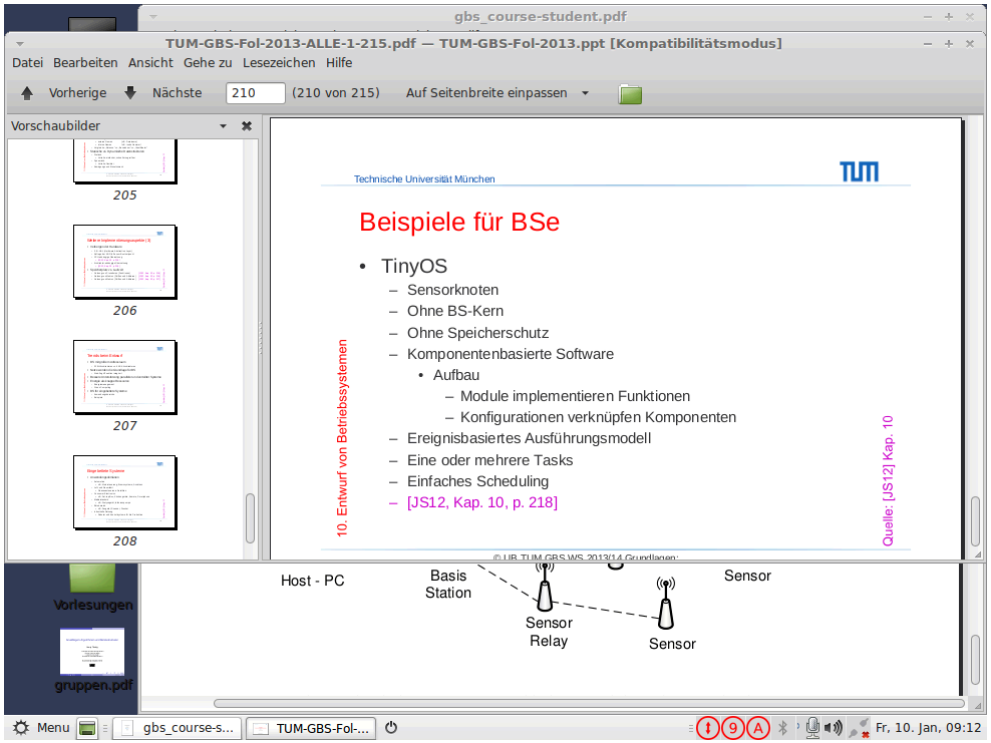
Eingebettete Systeme

- Produktkopplung
- Anforderungsvielfalt
  - Größe
  - Lebendauer
  - Nutzungsumgebung
  - Realzeitverhalten
- Betriebssysteme
  - Viele proprietäre BSe
  - Mobile Betriebssysteme
    - Symbian OS
    - Windows CE
    - Android auf Linux aufbauend

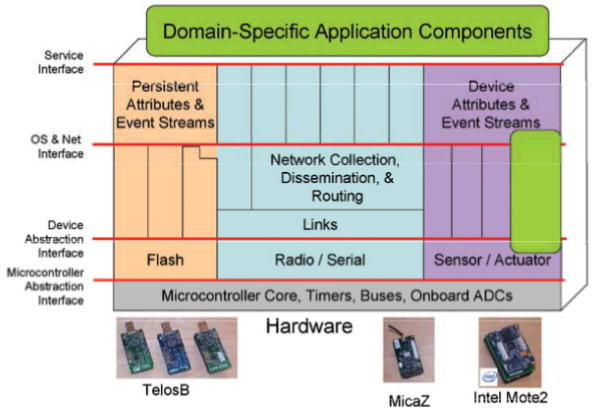
Beispiele für BSe

- TinyOS
  - Sensorknoten
  - Ohne BS-Kern
  - Ohne Speicherschutz
  - Komponentenbasierte Software
    - Aufbau
      - Module implementieren Funktionen
      - Konfigurationen verknüpfen Komponenten
  - Ereignisbasiertes Ausführungsmodell
  - Eine oder mehrere Tasks
  - Einfaches Scheduling
  - [JS12, Kap. 10, p. 218]





TinyOS



Exkurs: Strukturierungskonzepte

- Konzepte
  - Prozedur-orientiert vs. Nachrichten-orientiert
  - Kerne, Mikro-, Nano,-Pico-, Exo-Kerne
  - Virtualisierung
- BS-Strukturen
  - Monolithisch
  - Systemken
  - Schichtung
    - Problem: „Up-calls“
  - Hierarchie abstrakter Maschinen

Technische Universität München

Exkurs: Strukturierungskonzepte

- Konzepte
  - Prozedur-orientiert vs. Nachrichten-orientiert
  - Kerne, Mikro-, Nano,-Pico-, Exo-Kerne
  - Virtualisierung
- BS-Strukturen
  - Monolithisch
  - Systemken
  - Schichtung
    - Problem: „Up-calls“
  - Hierarchie abstrakter Maschinen

10. Entwurf von Betriebssystemen

© UB TUM GBS WS 2013/14 Grundlagen: Betriebssysteme und Systemsoftware (IN0009) 212

Technische Universität München

Exkurs: Aufrufformen

- Aufruf-Möglichkeiten
  - Sprünge
  - Kontexte
  - Umgebungen
  - Parameter
  - Ablaufstruktur
- Vor- und Nachteile
- Einsatz für Betriebssysteme

10. Entwurf von Betriebssystemen

Technische Universität München

Exkurs: Aufrufformen - Beispiele

a. Sprungbefehl
b. Lokaler Prozeduraufruf, statische Bindung
c. Lokaler Prozeduraufruf, dynamische Bindung
d. Systemaufruf
e. Nichtlokaler Prozeduraufruf
f. Erzeugung eines Dienstleistungsprozesses
g. Nachrichtenübermittlung
h. Lokaler nachrichtenbasierter Prozeduraufruf
i. Objektorientierter Methodenaufruf

[SB06] Kap. 9.2 (detaillierte Vor- und Nachteile)

10. Entwurf von Betriebssystemen

Exkurs: Weitere Beispiele

- DOS
- OS/2
- BS2000
- z/OS
- Apple Newton MessagePad
- RIM BB10