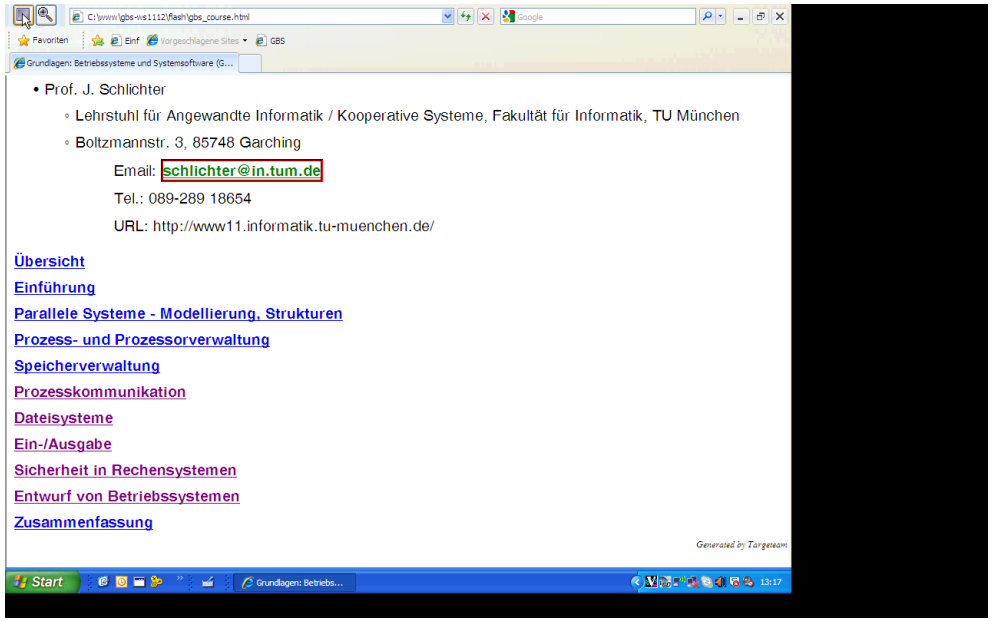
Script **generated by TTT**

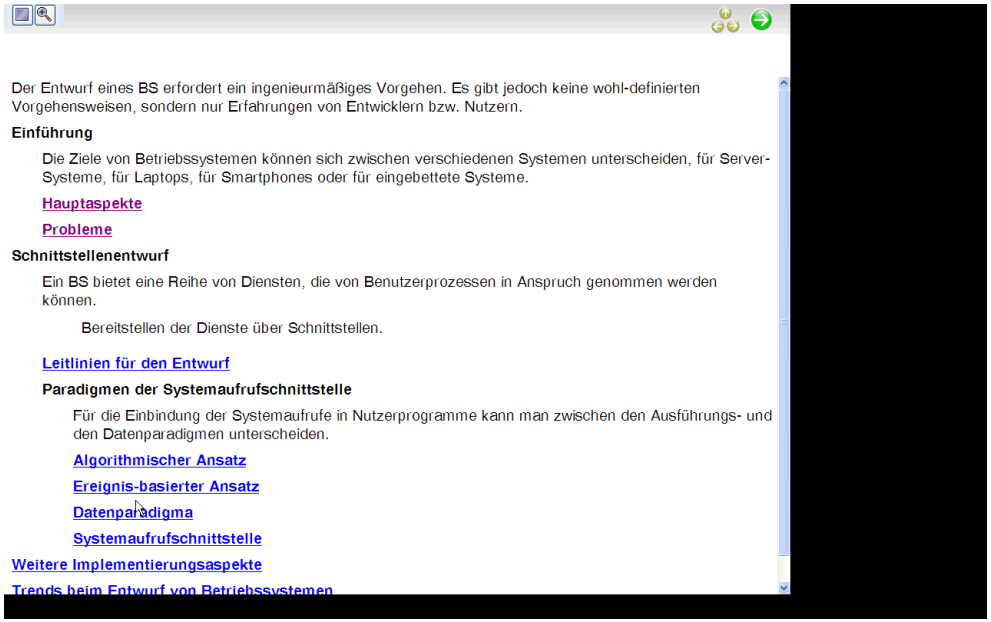
Title: Grundlagen_Betriebssysteme (08.02.2012)

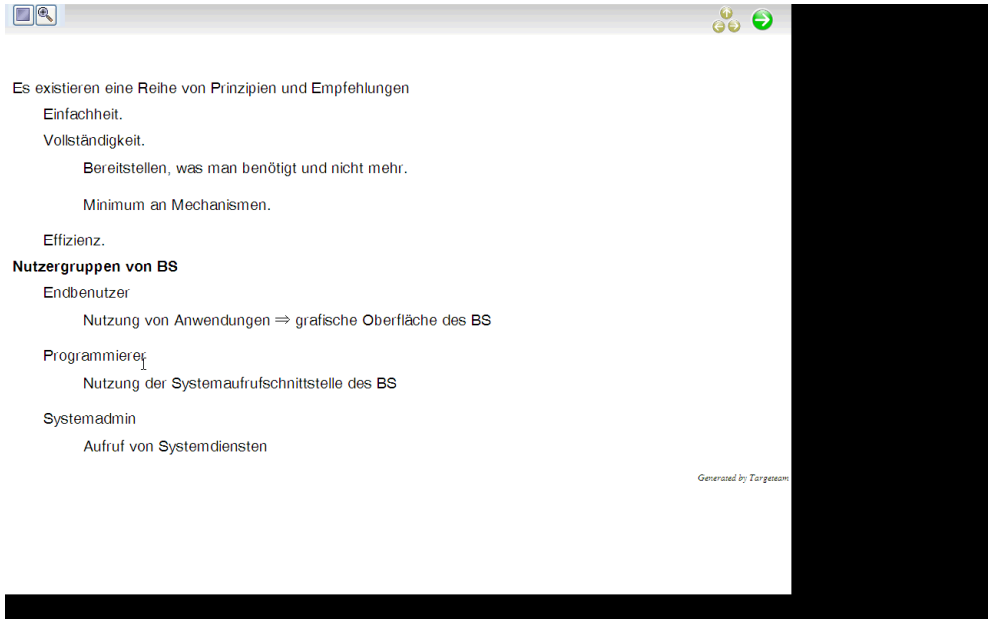
Date: Wed Feb 08 13:17:06 CET 2012

Duration: 44:42 min

Pages: 19







Vorgehensweisen, sondern nur Erfahrungen von Entwicklern bzw. Nutzern.

Einführung

Die Ziele von Betriebssystemen können sich zwischen verschiedenen Systemen unterscheiden, für Server-Systeme, für Laptops, für Smartphones oder für eingebettete Systeme.

Hauptaspekte

Probleme

Schnittstellenentwurf

Ein BS bietet eine Reihe von Diensten, die von Benutzerprozessen in Anspruch genommen werden können.

Bereitstellen der Dienste über Schnittstellen.

Leitlinien für den Entwurf

Paradigmen der Systemaufrufschnittstelle

Für die Einbindung der Systemaufrufe in Nutzerprogramme kann man zwischen den Ausführungs- und den Datenparadigmen unterscheiden.

Algorithmischer Ansatz

Ereignis-basierter Ansatz

Datenparadigma

Systemaufrufschnittstelle

Weitere Implementierungsaspekte

Trends beim Entwurf von Betriebssystemen

basiert auf der Idee, dass ein Programm eine bestimmte Funktion erfüllen soll, die es im Voraus oder durch Parameter kennt.

Basislogik ist im Code festgelegt

Gelegentliche Systemaufrufe, um

Benutzereingaben zu erhalten

BS-Dienste in Anspruch zu nehmen

Beispiel

```
main () {
    int ...;
    init();
    do_something();
    read (...); /* Systemaufruf */
    do_something_else();
    write (...); /* Systemaufruf */
    continue();
    exit(0);
}
```


Wie werden Systemstrukturen und Geräte im Betriebssystem gegenüber dem Programmierer präsentiert?

Beispiel Unix

"Alles" ist eine Datei

Vereinheitlichung von Dateien, E/A-Geräte und Pipes

Zugriff über Dateioperationen

fd1 = open("file1", O_RDWR)

fd2 = open("/dev/tty", O_RDWR)

Beispiel Windows 2000

"Alles" ist ein Objekt.

Entwurf von Betriebssystemen

Der Entwurf eines BS erfordert ein ingenieurmäßiges Vorgehen. Es gibt jedoch keine wohl-definierten Vorgehensweisen, sondern nur Erfahrungen von Entwicklern bzw. Nutzern.

Einführung

Die Ziele von Betriebssystemen können sich zwischen verschiedenen Systemen unterscheiden, für Server-Systeme, für Laptops, für Smartphones oder für eingebettete Systeme.

Hauptaspekte

Probleme

Schnittstellenentwurf

Ein BS bietet eine Reihe von Diensten, die von Benutzerprozessen in Anspruch genommen werden können.

Bereitstellen der Dienste über Schnittstellen.

Leitlinien für den Entwurf

Paradigmen der Systemaufrufschnittstelle

Für die Einbindung der Systemaufrufe in Nutzerprogramme kann man zwischen den Ausführungs- und den Datenparadigmen unterscheiden.

Algorithmischer Ansatz

Ereignis-basierter Ansatz

Datenparadigma

Systemaufrufschnittstelle

Weitere Implementierungsaspekte

Trends beim Entwurf von Betriebssystemen

Systemaufrufschnittstelle

Entscheidung bzgl. der Bereitstellung unterschiedlicher Varianten von Systemaufrufen

Z.B. Lese-Systemaufruf: read_file, read_process, read_tty, ...

Besser: nur Bereitstellung des allgemeinen Falls

Z.B. Lese-Systemaufruf: read

Restliche Varianten über Bibliotheksfunktionen auf den allgemeinen Systemaufruf abbilden

Sichtbarkeit von Systemaufrufen

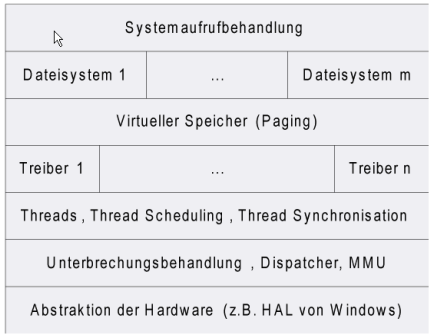
Trennung von Systemaufrufen und Bibliotheksaufrufen

⇒ Ermöglicht übersichtliche Darstellung der vorhandenen Systemaufrufe

Weitere Implementierungsaspekte

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

Ein etablierter und erprobter Architekturansatz sind geschichtete Systeme.



Mechanismen vs. Policies

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

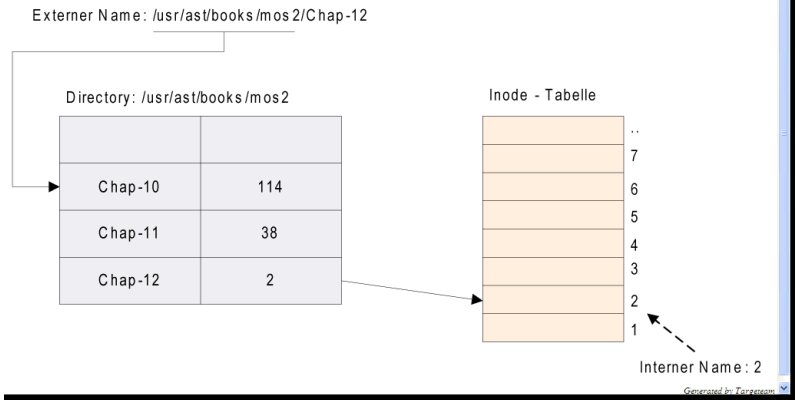
Namensräume

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.



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

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)
```

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

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.

BS für eingebettete Systeme

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.

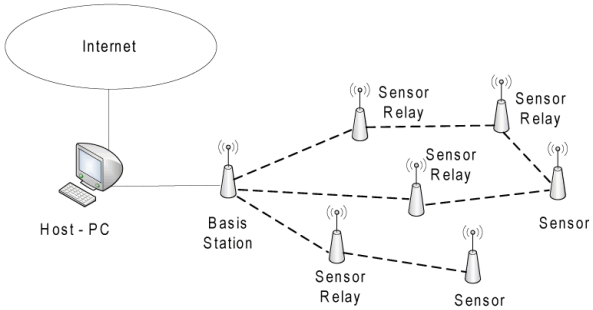
Android Systemarchitektur basiert auf Linux Kernel (Treiber, Speicher-/Prozessmanagement); darauf existieren Bibliotheken (z.B. für Graphik), eine Laufzeitumgebung und ein Anwendungsframework.

TinyOS

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).

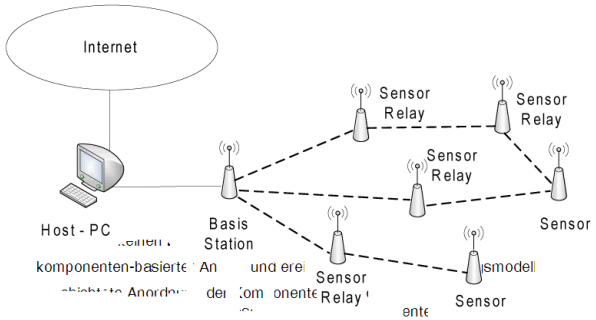
Anwendungen werden durch Verknüpfung von Komponenten realisiert.

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

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