

Treiberentwicklung, Echtzeit- und Betriebssysteme

Prof. Dr. rer. nat. Peter Gerwinski

12. Mai 2025

Treiberentwicklung, Echtzeit- und Betriebssysteme

1 Einführung

2 Unix

3 Treiberentwicklung

3.0 Massenspeicher und Gerätedateien

3.1 Mikrocontroller

3.2 Betriebssysteme ohne Speicherschutz

3.3 Linux-Kernel-Module

3.4 Die Systembibliothek (libc)

3.5 Der Betriebssystemkern (Kernel)

...

3.3 Linux-Kernel-Module

- **#include** <linux/module.h>
#include <linux/kernel.h>
- Zwei „Hauptprogramme“: `init_module()`, `cleanup_module()`
- `printk()` statt `printf()`
- Compilieren mit speziellem **Makefile**
- Angabe der Lizenz ist wichtig: **MODULE_LICENSE()**
- Kommunikation mit Anwendungen:
Geräte-Dateien (*major/minor number*)
- Kommunikation mit dem Betriebssystemkern:
Callback-Funktionen (virtuelle Methoden)
- Automatisches Anlegen von Gerätedateien:
Geräteklassen

3.4 Die Systembibliothek (libc)

Verbindung zwischen Anwendung und Betriebssystemkern (Kernel)

- Verbindung zur Anwendung
- Verbindung zum Betriebssystemkern

3.4 Die Systembibliothek (libc)

Verbindung zwischen Anwendung und Betriebssystemkern (Kernel)

- Verbindung zur Anwendung: Funktionsaufruf
- Verbindung zum Betriebssystemkern

3.4 Die Systembibliothek (libc)

Verbindung zwischen Anwendung und Betriebssystemkern (Kernel)

- Verbindung zur Anwendung: Funktionsaufruf
Indirekter Funktionsaufruf über eine Tabelle von Zeigern,
abhängig von der Datei
—→ Aufruf einer virtuellen Methode (objektorientiert)
- Verbindung zum Betriebssystemkern

3.4 Die Systembibliothek (libc)

Verbindung zwischen Anwendung und Betriebssystemkern (Kernel)

- Verbindung zur Anwendung: Funktionsaufruf
Indirekter Funktionsaufruf über eine Tabelle von Zeigern,
abhängig von der Datei
—→ Aufruf einer virtuellen Methode (objektorientiert)
- Verbindung zum Betriebssystemkern: System Call

3.4 Die Systembibliothek (libc)

Verbindung zwischen Anwendung und Betriebssystemkern (Kernel)

- Verbindung zur Anwendung: Funktionsaufruf
Indirekter Funktionsaufruf über eine Tabelle von Zeigern,
abhängig von der Datei
—→ Aufruf einer virtuellen Methode (objektorientiert)
- Verbindung zum Betriebssystemkern: System Call
Das Betriebssystem hat mehr Rechte als die Anwendung,
daher darf die Anwendung Funktionen des Betriebssystems
nur auf sehr kontrollierte Weise aufrufen.
—→ Hardware-Unterstützung
Beispiel: x64-Befehlsarchitektur: `syscall`

3.5 Der Betriebssystemkern (Kernel)

- „Eingang“
Wo nimmt der Kernel den `syscall`-Befehl entgegen?
- „Ausgang“
Wo ruft der Kernel die Callback-Funktionen der Treibermodule auf?
- „Weg durch den Kernel“
Was passiert dazwischen?