Systemsoftware

Embedded System

Prof. Dr. Michael Mächtel

Informatik, HTWG Konstanz

Version vom 06.03.17

Übersicht

Übersicht

Definition

- Integrierte, mikroelektronische Steuerung
  - Meist für eine spezifische Aufgabe
  - Oft kein ausgeprägtes User-Interface
- Beispiele
  - Navi-System
  - Router
  - Handy
  - Waschmaschine
  - ....

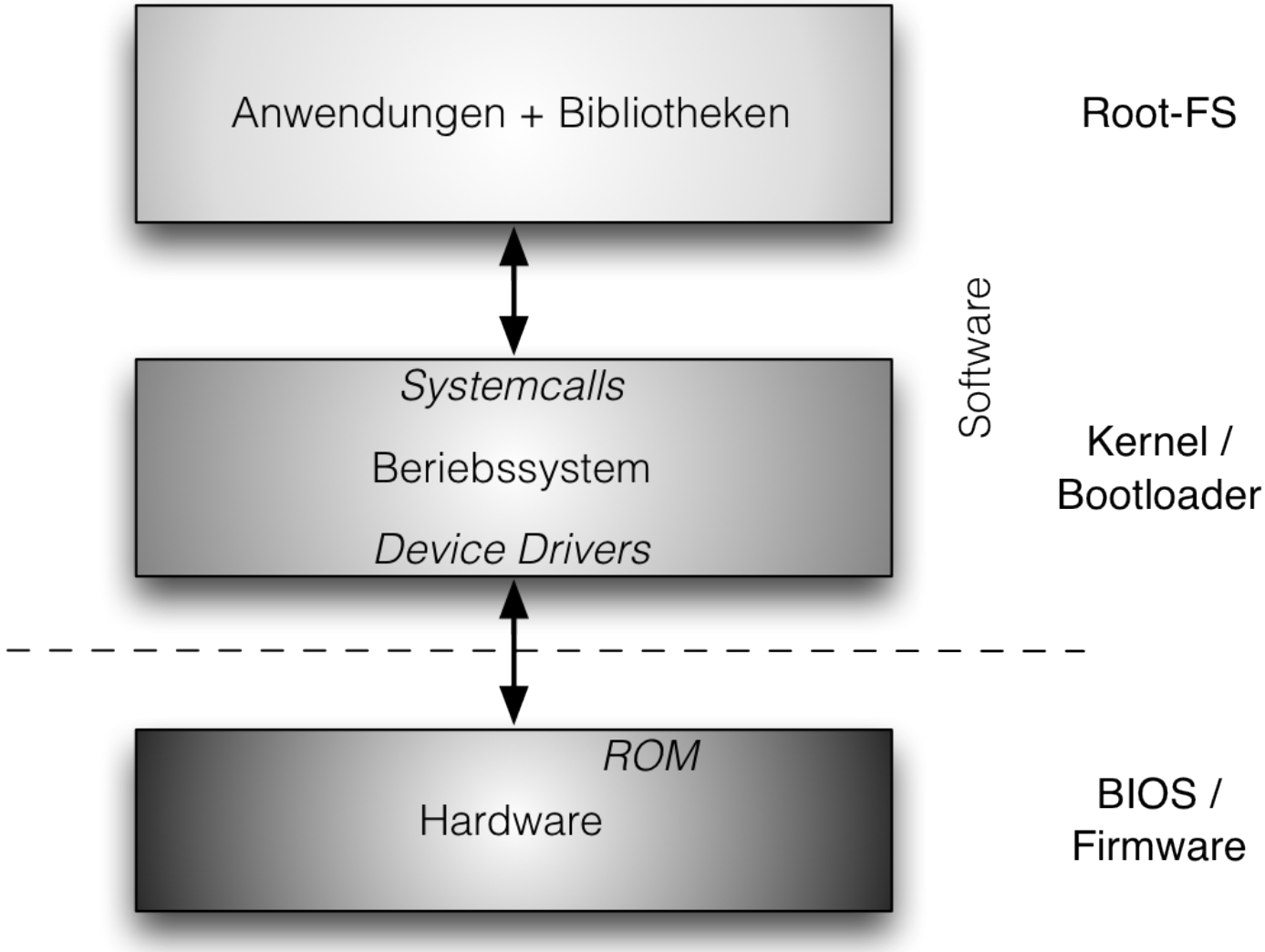
Standard- versus Embedded-System

- Ressourceneinschränkungen:
  - Eingeschränkte Platzverhältnisse, kompakt und klein.
  - Kein oder eingeschränktes GUI.
  - Oftmals stehen weder Bildschirm noch Tastatur zur Verfügung (Headless-Betrieb)
  - Oft keine sich bewegende Teile (Diskless).
- Systembesonderheiten:
  - Es werden möglichst kurze Bootzeiten gefordert.
    - Beispiel: Cockpit im Auto.
  - Systeme werden ohne Vorwarnung stromlos geschaltet.
    - Kein ordnungsgemäßes Herunterfahren.
  - Die Systeme laufen „non-stop.“
    - 24h/Tag, 7Tage/Woche, 52Wochen/Jahr.
  - Lange Lebenszeiten (Jahrzehnte)
    - Wartung muss für die Lebensdauer garantiert werden.

Folgerung

- Exakte Anpassung der Systemsoftware an die Hardware und an die gewünschte Funktionalität.
  - Es werden nur die Funktionen realisiert, die auch benötigt werden (kein überflüssiger Overhead).
  - Verwendung speziell auf Basis der Ressourcen-Knappheit angepasster Komponenten (z.B. Busybox).
- Es werden auf die Gerätefunktion angepasste Hardwareplattformen verwendet (CPU, Architektur)
  - Cross-Compile-Toolchain
  - Systemgeneratoren
  - Simulator/Emulator

Architektur Embedded System



Software Komponenten

- Firmware (Bootloader)
- Main-Operating System
  - Kernel
  - Userland
- Betriebssysteme für Zusatzkomponenten
  - Mobilfunkankopplung
- Applikationen

Transfer Software -> Image

- Der Transfer des Programms/Images auf das Target über:
  - die seriellen Schnittstelle,
  - dem Background Debug Mode Interface (BDM),
  - einem EPROM oder
  - ein Flash

Übersicht

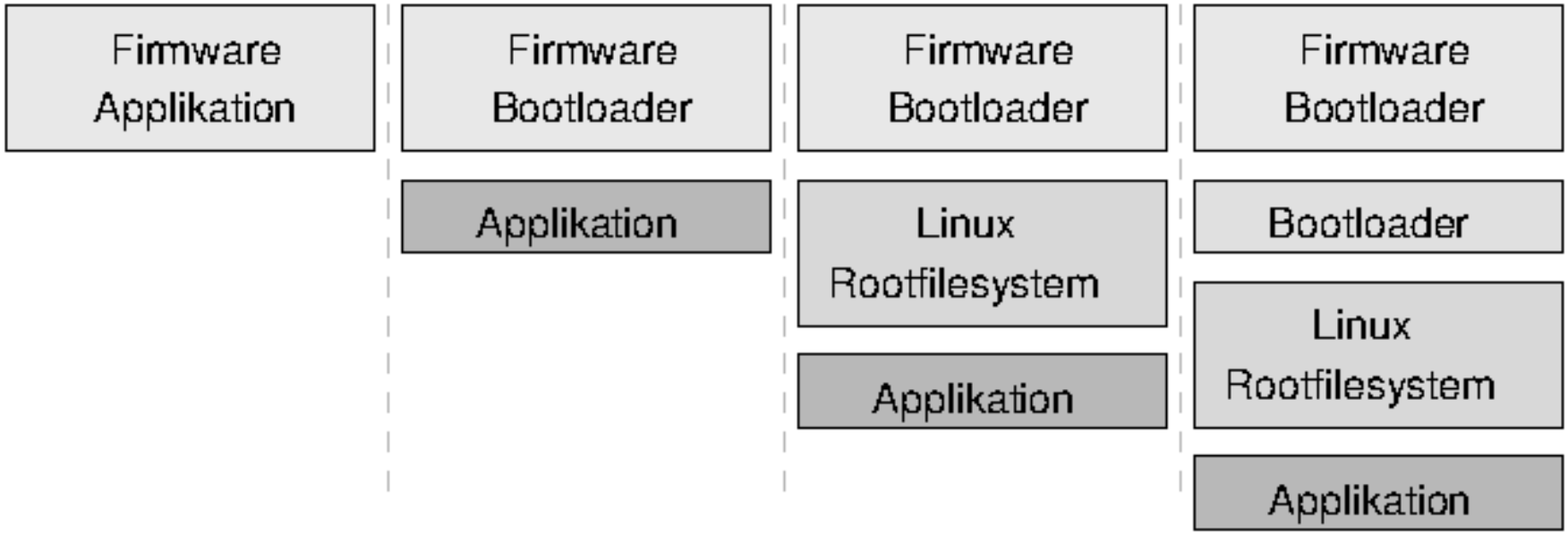
Begriffe

- Der Begriff 'Linux' ist umgangssprachlich ein austauschbarer Begriff für:
  - den Kernel des Betriebssystems,
  - das komplette System incl. dem Kernel
  - oder auch einer Distribution.
- Softwarekomponenten eines Embedded Linux Systems:
  - Firmware (Bios)
  - Bootloader
  - (Linux-) Kernel
  - Rootfilesystem

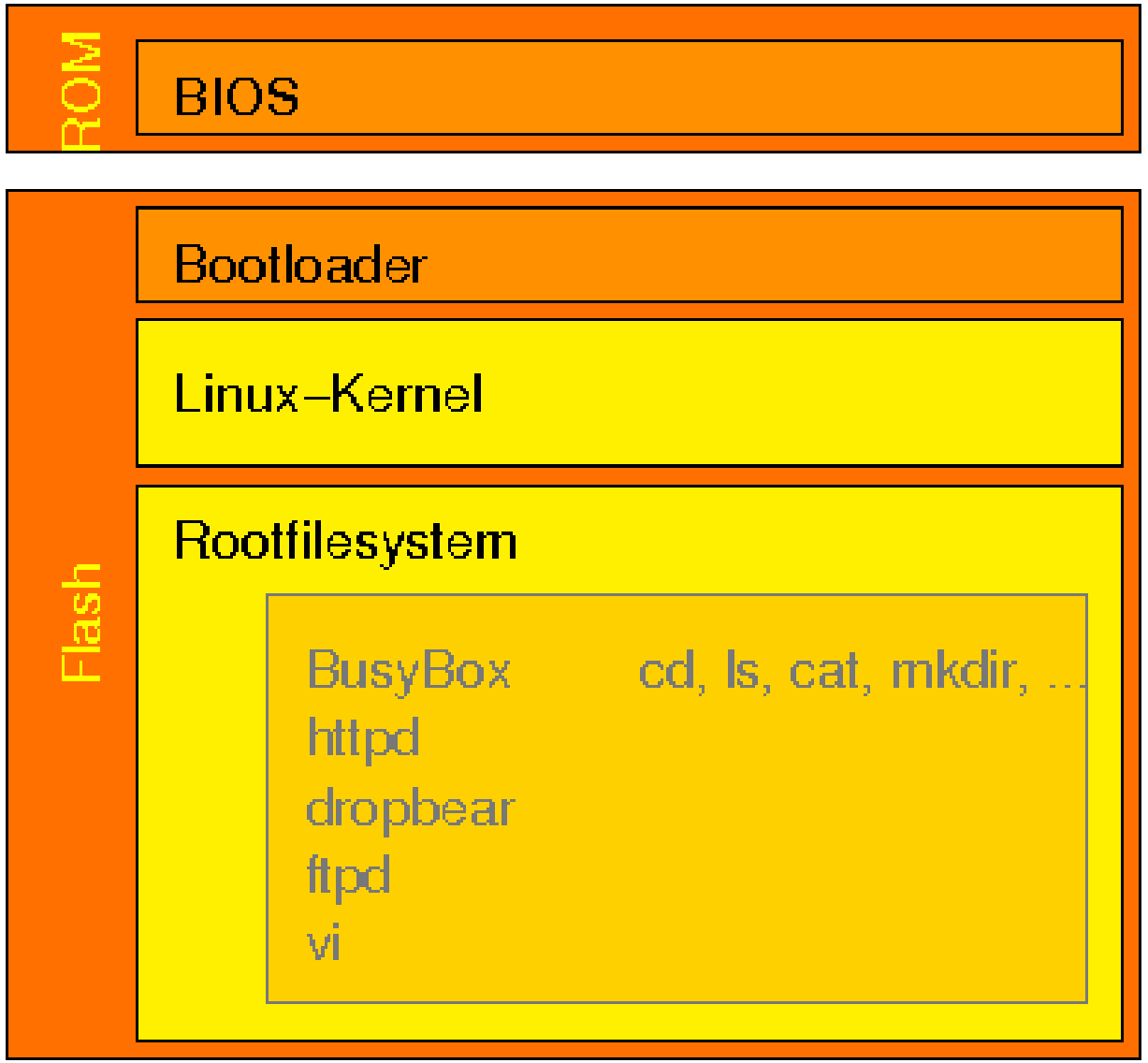
‘Embedded Linux’

- Umgangssprachlicher Begriff für komplettes System (Kernel, Lib + Anwendung)
- Embedded Linux Kernel = Linux Kernel, speziell für die entspr. Hardware konfiguriert
- Unterschiedlicher Entwicklungsablauf zwischen Anwendungen für Standard-Linux und Embedded Linux:
  - Entwicklungsumgebung auf Host System.
  - Embedded Linux läuft auf dem Target System.

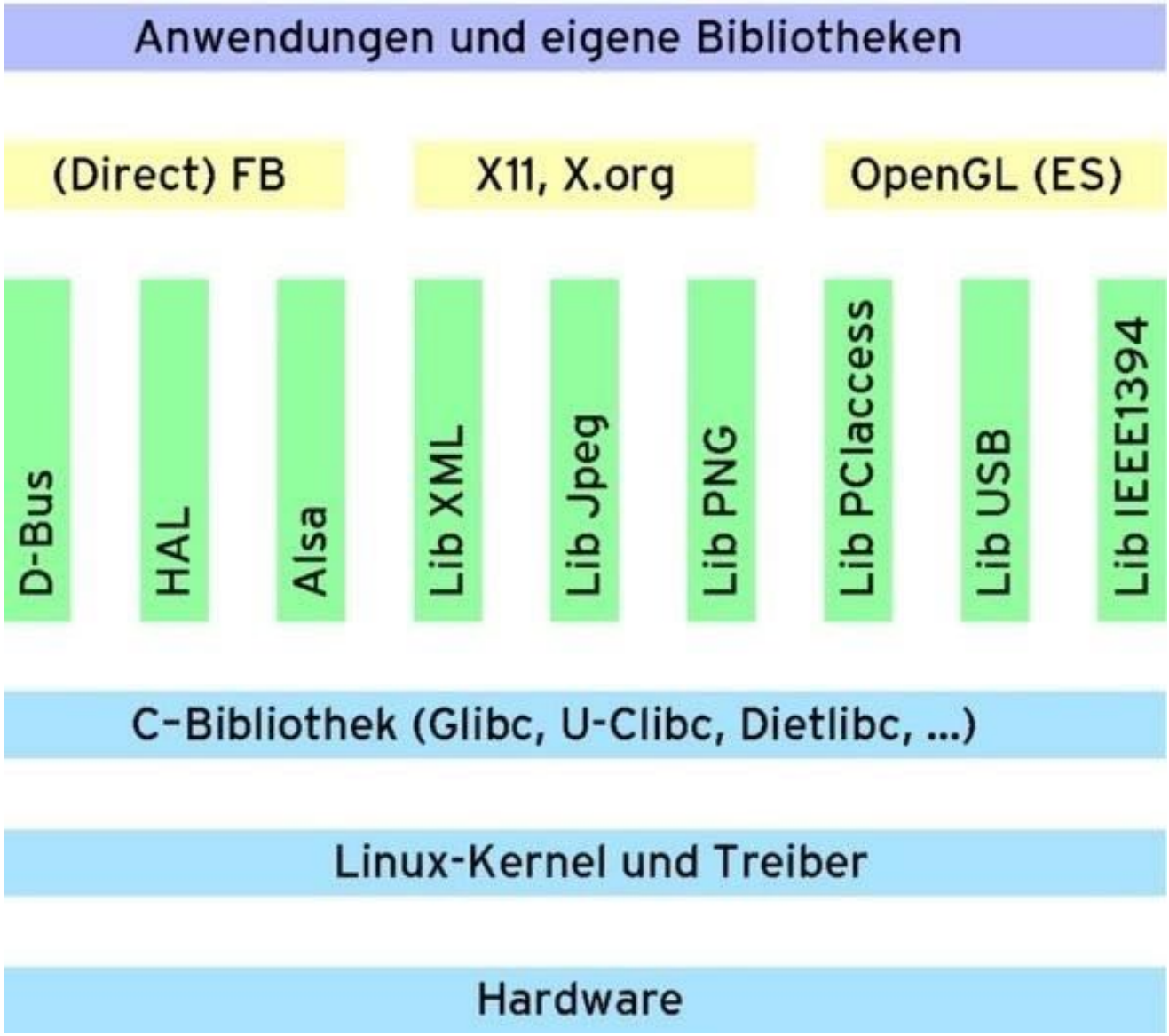
Hierachie von Embedded Architekturen



Softwareverteilung ROM/Flash



Linux Bibliotheken



Programmiersprachen und Tools

- Der Entwickler hat unter Embedded Linux die freie Wahl der Programmiersprache:
  - C, C++, Objective C, D, Ada, Pascal, Java und vielen andere ...
- Auch die aus dem Unix-Umfeld gewohnten Skriptsprachen, etwa Perl, Python, Ruby, PHP, Lua usw. stehen zur Verfügung
- Entwickelt wird wie auf dem Host System mit Standard Tools wie z.B. make, gcc, gdb usw.

ToolChain

- Kommerzielle Embedded Linux Distributionen
  - liefern eigene fertige Cross-Tool-Chains
  - Beispiele: MontaVista, TimeSys, kaeilos, ...
  - Vorteil: Support!
- freie Toolchain Builder:
  - crossTool
  - buildroot
  - openEmbedded
  - Nachteil: Kein Support!
- Welches nehmen: Frage nach Aufbau von KnowHow (inHouse oder extern)

Rechtliche Aspekte

Verwendete Software besitzt Copyrights. Verwendete Lizenzen:

- GPL
  - Modifikationen müssen öffentlich gemacht werden, modifizierter Quellcode muss (auf Nachfrage) abgegeben werden.
  - Ohne Modifikationen ist kaum ein System aufzubauen.
- BSD
  - Autoren müssen genannt werden.
  - Modifikationen sind ohne weitere Einschränkungen möglich.
- Im Bereich Kernel sind im Umfeld eingebetteter Systeme in den meisten Fällen der Code für Gerätetreiber offen zu legen.

Unterschied Standard Linux

- Typischerweise erheblich weniger Ressourcen (Speicherplatz, Rechenleistung, Stromverbrauch)
- Cross-Entwicklung mit Cross-Compiler
  - typischerweise keine grafische Oberfläche auf Gerät
- Hardwarespezifische Bootloader
- möglichst kurze Bootzeit
- Systeme ohne Vorwarnung stromlos (kein “Herunterfahren”)
- höhere Lebenszyklus des Systems
  - Support!

Vorteile Linux

- Skalierbarkeit von Linux erlaubt moderate Hardware-Ressourcen
- Unterstützung aller gängigen Embedded Prozessoren
- Aktive Entwicklergemeinde, dadurch neueste Technologien im Kernel als auch im Anwendungsbereich.
- Firmen bieten kommerzielle Tools und Entwicklungsumgebungen an incl. Support
- Gleiche Tools auf Entwicklungs- und Zielrechner möglich:
  - Linux mit Cross Umgebung für Target
- Grosse Softwareauswahl
- Standard POSIX API's

Nachteile Linux

- Mindestanforderungen an die Hardware: 32Bit, >4MB RAM
- Somit nicht für Deeply Embedded Systems geeignet
- Einschränkungen (noch) im Safty-Critical und Realtime Bereich.

Weiteres Vorgehen

- Im Praktikum benutzen wir einen Emulator
  - Emulator enthält ‘BIOS’ und ‘Bootloader’
- Für das Embedded Linux System fehlt noch:
  - Linux Kernel
  - Linux Rootfilesystem