Praktikum angewandte Systemsoftwaretechnik (PASST)

Dateisysteme / Aufgabe 6

13. Januar 2020

Tobias Langer, Michael Eischer
und Florian Schmaus

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Rückblick

Rückblick

- Linux Kernelmodule
  - Dynamisch lad-/entladbar werden
- **U**niversal **S**erial **B**us
  - Kein wirklicher Bus
  - Verschiedene Transferarten
  - Kommunikation über Endpoints und Pipes unter Linux
- `sysfs`
  - FileIO-basierte Schicht zwischen Kernel-/Userspace
  - Stellt Zugriff auf Kernelobjekte (`kobjects`) bereit

Motivation

Leaving the beaten track

- Dateisystem ist zentraler Teil eines Betriebssystems
  - Dateiein- und ausgabe
  - Schnittstelle zum Linuxkernel
- Entwicklung betrifft dabei viele Komponenten
  - Verwaltung von Dateisysteminstanzen, Inodes, ...
  - Organisation von (Meta-)Daten
  - Adressraummapping
- Allerdings...
  - Kaum/wenige verlässliche Dokumentation
  - Viel muss anhand vorhandenem Code abgeleitet werden

Lernziele

Lernziele

Im Anschluss an diese Aufgabe solltet Ihr...

- die Kernkomponenten eines Dateisystems aufzählen
- die Funktionsweise des *VFS* erklären
- eigene Dateisystemtreiber programmieren

...können.

Agenda

Agenda

Dateisysteme

Virtual Filesystem

Zusammenfassung

Aufgabe 6

Dateisysteme

Von Dateisystemen

- Dateisystem
  - Organisation von Daten auf Datenträger
  - Zuordnung von (Meta-)Daten zu Ablageort
  - Beispiele: ext{2,3,4}, ntfs, ...
- Dateisystemtreiber
  - Setzt Zugriff auf Dateisystem um
  - Implementiert `open()`, `read()`, `write()`, ...
- Zusätzlich: Pseudo-Dateisysteme: procfs, sysfs
  - Unix-Philosophie: „Everything is a file“
  - Exportieren Kernelstrukturen als Dateischnittstelle

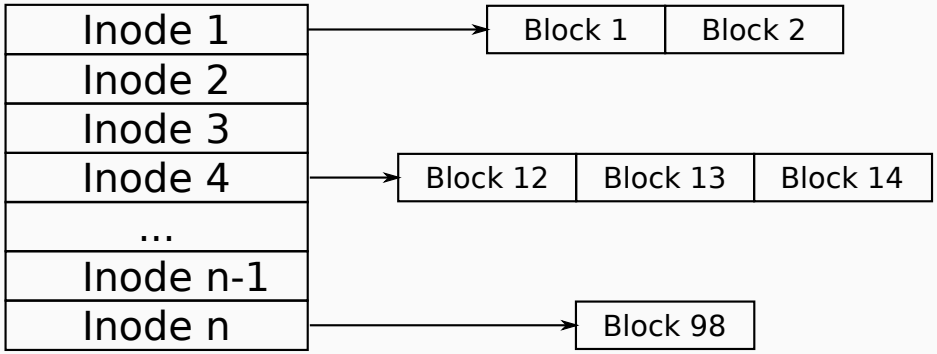
Der Dateibaum unter Linux

- Dateien sind als Baum organisiert
  - $\rightarrow$ „hierarchisches” Dateisystem
- Wurzelverzeichnis: /
- Manche Einträge sind „Mountpoints”
  - Dahinter verbirgt sich eine weitere Dateisysteminstanz
  - Operationen können dort eine vollständig andere Semantik haben

Bestandteile eines einfachen Dateisystems

Prinzipieller Aufbau eines Unix-Dateisystems:

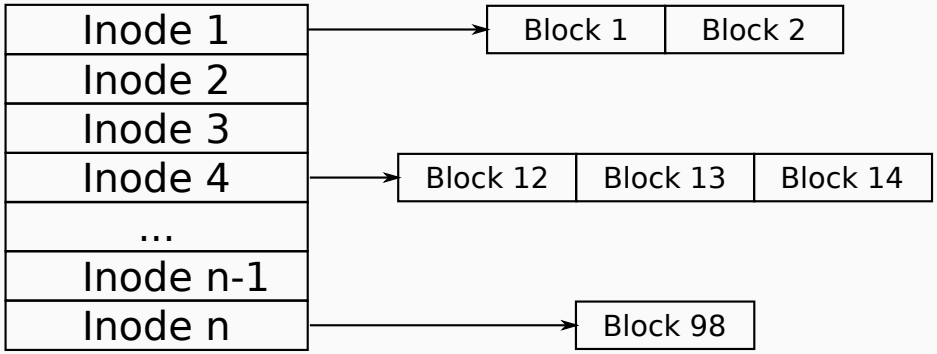
- Inodes
  - Metadaten der Dateien
  - Verweise auf Dateiblöcke
- Blöcke
  - Nutzdaten
  - Verzeichnisinhalte



Bestandteile eines einfachen Dateisystems

Prinzipieller Aufbau eines Unix-Dateisystems:

- Inodes
  - Metadaten der Dateien
  - Verweise auf Dateiblöcke
- Blöcke
  - Nutzdaten
  - Verzeichnisinhalte



Blockgröße ≠ Blockgröße

- Blockgröße wird durch Dateisystem vorgegeben
- Entspricht nicht phys. Blockgröße des Datenträgers

Inodes - Daten über Daten ...

Inode

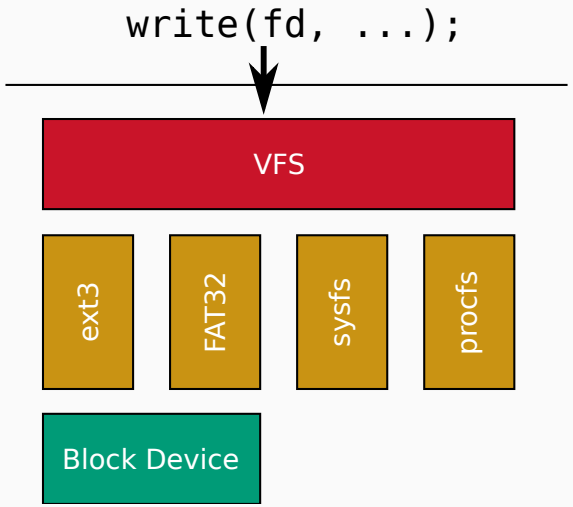
Nummer	Type	Größe	Besitzer	Verweiszähler	...	Verweis auf Datenblöcke

- Eine Index node beschreibt Objekte im Dateisystem
  - Eindeutige ID
  - Typ & Größe der Datei
  - Besitzer & Zugriffsrechte
  - Verweis auf die Nutzdaten (wenn vorhanden)
  - Verweiszähler
  - ...
- Verzeichnisse ordnen Inodes Namen zu
  - Mehrere Namen für eine Inode möglich

Virtual Filesystem

Das Linux Virtual Filesystem (VFS) (1/2)

- Abstraktion für Dateisysteme
  - Generische Schnittstelle
  - Implementiert `write()`, `open()`, ...



  - Untergliedert entsprechend Dateisystemaufbau
    - Dateisystemtyp
    - Superblock
    - Inode
    - Datei

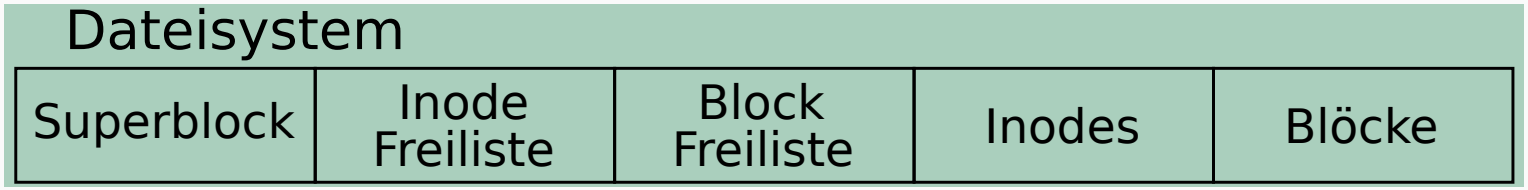
Das Linux Virtual Filesystem (VFS) (2/2)

- „Objektorientierung” im Kernel
  - Struktur für Objektinstanz
  - Zusätzliche Struktur für „Methoden” ($=$ Callbacks)
- Spezialisierung durch Treiber
  - Implementieren von Initialisierern ($\approx$ Konstruktor)
  - Implementieren & Setzen von Callbacks ($\approx$ Vererbung)
- VFS stellt generische Implementierung bereit

Das Linux Virtual Filesystem (VFS) (2/2)

- „Objektorientierung” im Kernel
  - Struktur für Objektinstanz
  - Zusätzliche Struktur für „Methoden” (= Callbacks)
- Spezialisierung durch Treiber
  - Implementieren von Initialisierern (≈ Konstruktor)
  - Implementieren & Setzen von Callbacks (≈ Vererbung)
- VFS stellt generische Implementierung bereit
  - … in der Regel
  - Andernfalls kann es zu Abstürzen kommen

Der Dateisystemtyp im VFS



Dateisystemtyp

- Repräsentiert durch `struct file_system_type`
- Erzeugt Dateisysteminstanzen

Dafür benötigt der Treiber …

- Attribute des Dateisystems (Namen, Statusflags, …)
- `fill_super`-Callback

Der Superblock im VFS

Superblock - Instanz des Dateisystems

- Repräsentiert durch `struct super_block`
- Generische Kernelschnittstelle

Dafür benötigt der Treiber ...

- `struct super_operations`
- Implementierung der unterstützten Callbacks

Ergebnis im Userspace

- Mounten eines Dateisystems

Inodes im VFS

Inode

- Repräsentiert durch `struct inode`
- Generische Kernelschnittstelle

Dafür benötigt der Treiber …

- `struct inode_operations`
- Implementierung der unterstützten Callbacks

Ergebnis im Userspace

- `open`, `link`, `rename`, …Syscalls funktionieren

Dateien im VFS

Geöffnete Dateien

- Repräsentiert durch `struct file`

Dafür benötigt der Treiber …

- `struct file_operation`
- Implementierung der unterstützten Callbacks

Ergebnis im Userspace

- Dateien können gelesen & geschrieben werden
- Verzeichnisinhalte können gelesen werden

Weitere Komponenten

- Dentries - Direcory entries
  - Repräsentiert einen Verzeichniseintrag
  - Verweis auf Inode und Eltern-Dentry
  - Werden für schnelleren Zugriff gecached
  - Werden nicht auf der Festplatte gespeichert
- Adressraum-Operationen
  - Dateioperationen interagieren indirekt mit Blockdevice
  - Stattdessen: Adressraum-Operationen
  - Implementierung des `get_block` Callbacks nötig
    - Mapping Dateisystemblöcke auf Blockdeviceblöcke

Und jetzt...?

- Schrittweises Vorgehen:
  1. Kernelmodul
  2. Mount
  3. Anzeigen von Ordnerinhalten
  4. Lesen von Dateien
  5. Schreiben von Dateien

- Dateisystemformat ist bei Weitem nicht ausgereift
  - Fragmentierung bei Verkleinerung / Löschen von Dateien
  - Kopieraufwand bei Vergrößerung von Dateien
  - ...

Zusammenfassung

Zusammenfassung

- Dateisysteme
  - Speicherung und Struktur von Daten auf Datenträger
  - Ablage als Metadaten, Inodes & Datenblöcke
  - Außerdem: Pseudodateisysteme
- Virtual File System
  - Einheitliche Dateisystemschnittstelle des Linuxkerns
  - Stellt generische Basisoperationen bereit
  - Erweitert durch konkrete Dateisystemtreiber

Aufgabe 6

Aufgabe

- Einarbeiten in die benötigten APIs im Linux-Kern
  - Dokumentation, Codebeispiele
  - Empfohlenes Dateisystem zum Verständnis: bfs, minix
- Lesen & Verstehen der Spezifikation
- Programmieren des Dateisystems
- Testen des bereitgestellten Images

Abgabe: Bis 04. Februar 2020 durch Vorführung in einer Rechnerübung

Ein einfaches Dateisystem (1/2)

Dateisystem

- Feste Zahl an Inodes & Blöcken
- Block- & Inodefreilisten als Bitmaps
- Superblock enthält statische Informationen
- Dateien werden als Blockintervall gespeichert

Ein einfaches Dateisystem (2/2)

Superblock

Magic Number	Blockgröße	Anzahl Inodes	Anzahl Blöcke	Reserviert

- Magic Number: `0x53462d5453534150`
- Blockgröße (idR. 512 Byte)
- Absolute Offsets der Dateisystemsegmente (in Blöcken)

Frage

- Welche Vor-/Nachteile hat das Layout?

Fragen?