

Praktikum angewandte Systemsoftwaretechnik (PASST)

Dateisysteme / Aufgabe 6

04. Juli 2019

Tobias Langer, Stefan Reif, 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

- Linux Kernelmodule
 - Dynamisch lad-/entladbar werden
- **Universal Serial Bus**
 - 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

- Dateisysteme sind Hauptkomponenten
 - Dateiein- und ausgabe
 - Kommunikation mit dem 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

Im Anschluss an diese Aufgabe solltet Ihr...

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

...können.

Agenda

Dateisysteme

Virtual Filesystem

Zusammenfassung

Aufgabe 6

Dateisysteme

Was ist ein Dateisystem?

- Beispiele: ext{2,3,4}, ntfs, ...
 - Verwaltet persistenten Speicher
 - Einheitliche Schnittstelle
 - Implementierung von `open()`, `read()`, `write()`, ...
- Pseudo-Dateisysteme: procfs, sysfs
 - Exportieren Kernel-Datenstrukturen
 - Zugriff über Dateischnittstelle

Was ist ein Dateisystem?

- Beispiele: ext{2,3,4}, ntfs, ...
 - Verwaltet persistenten Speicher
 - Einheitliche Schnittstelle
 - Implementierung von `open()`, `read()`, `write()`, ...
- Pseudo-Dateisysteme: procfs, sysfs
 - Exportieren Kernel-Datenstrukturen
 - Zugriff über Dateischnittstelle

Aufgabe 6

Implementierung eines Dateisystems im Linux-Kernel

- Dateien sind als Baum organisiert
 - „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

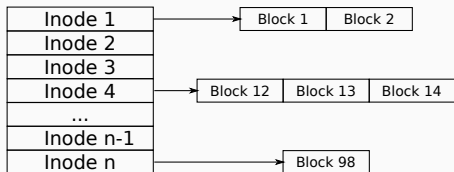
Prinzipieller Aufbau eines Unix-Dateisystems:

■ Inodes

- Metadaten der Dateien
- Verweise auf Dateiblocke

■ Blöcke

- Nutzdaten
- Verzeichnisinhalte



Bestandteile eines einfachen Dateisystems

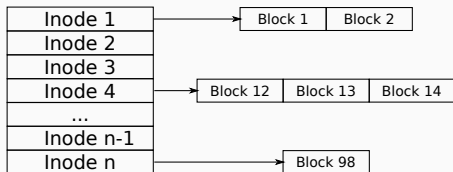
Prinzipieller Aufbau eines Unix-Dateisystems:

■ Inodes

- Metadaten der Dateien
- Verweise auf Dateiblöcke

■ Blöcke

- Nutzdaten
- Verzeichnisinhalte



Blockgröße \neq Blockgröße

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

Inode

Nummer	Type	Größe	Besitzer	Verweiszähler	...	Verweis auf Datenblöcke
--------	------	-------	----------	---------------	-----	-------------------------

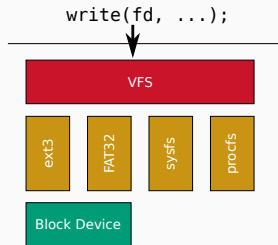
- Eine Index node beschreibt Objekte im Dateisystem
 - Eindeutige ID
 - Typ & Größe der Datei
 - Besitzer & Zugriffsrechte
 - Verweis auf die Nutzdaten (wenn vorhanden)
 - Verweiszähler
- Verzeichnisse weisen Inodes Namen zu
 - Mehrere Namen für eine Inode möglich

Virtual Filesystem

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

- Abstraktion für Dateisysteme

- Generische Schnittstelle
- Implementiert `write()`, `open()`, ...



- Untergliedert entsprechend Dateisystemaufbau

- Dateisystemtyp
- Superblock
- Inode
- Datei

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

Dateisystem

Superblock	Inode Freiliste	Block Freiliste	Inodes	Blöcke
------------	--------------------	--------------------	--------	--------

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

Dateisystem

Superblock	Inode Freiliste	Block Freiliste	Inodes	Blöcke
------------	-----------------	-----------------	--------	--------

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

Dateisystem

Superblock	Inode Freiliste	Block Freiliste	Inodes	Blöcke
------------	-----------------	-----------------	--------	--------

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

Dateisystem

Superblock	Inode Freiliste	Block Freiliste	Inodes	Blöcke
------------	-----------------	-----------------	--------	--------

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

- 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

- 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

- 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

- 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 25. Juli 2019 durch Vorführung in einer Rechnerübung

Ein einfaches Dateisystem (1/2)

Dateisystem

Superblock	Inode Freiliste	Block Freiliste	Inodes	Blöcke
------------	--------------------	--------------------	--------	--------

- 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?