5 alternative Übungsaufgabe #6b: Entwicklung eines Dateisystems

Diese Aufgabe kann statt der „Aufgabe 6: Entwicklung eines PCI-Treibers“ bearbeitet werden.
Im Rahmen von Aufgabe 6b soll ein Dateisystem entwickelt werden.

5.1 Grundlagen

Diese Aufgabe umfasst die Recherche notwendiger Techniken und Verfahren zur Implementierung eines Dateisystems. Einen ersten Einstieg können hierbei die Folien zu Systemprogrammierung II, letztes Kapitel(`https://www4.cs.fau.de/Lehre/WS16/V_SP2/Vorlesung/folien.shtml`) geben. Jedoch ist das hinzuziehen weiterer Resourcen, beispielsweise andere im Linux-Kernel vorhandene Dateisysteme, deren Dokumentation, Inhalte anderer Lehrveranstaltungen beispielsweise aus dem Datenbankumfeld sowie weiterer vorhandener Literatur auf dem Gebiet ratsam.

5.2 Funktionsumfang

Es soll ein Dateisystem im Kernel implementiert werden, das auf gegebenen Block-Devices das permanente Ablegen von Dateien in einer Üblichen Verzeichnisstruktur ermöglicht. Zum Erstellen eines entsprechenden (leeren) Dateisystems soll, falls nötig, eine entsprechende Userspace-Anwendung bereitgestellt werden. Falls notwendig kann ausserdem das Systemprogramm `mount` angepasst werden.
Die interne Struktur des Dateisystems, insbesondere auch das Format der Daten auf der Platte sind nicht fest vorgegeben und dürfen geeignet frei gewählt werden.

Mindestanforderungen ans Dateisystem: Das Dateisystem muss in der Lage sein

- übliche Dateinamen bis zu einer Länge von mindestens 63 Zeichen aus einem beliebigen Zeichensatz (mit den Unix-üblichen Ausnahmen Nullbyte und Schrägstrich) zu verwenden.
- „normale“ Dateien bis zu einer Grösse von mindestens 100 MByte zu speichern.
- Dateirechte (traditionelle 9-bit Unix-Rechte) sowie UID und GID zu verwalten und zu beachten.
- wie gewohnt gemountet zu werden, auch mehrmals, d.h. mehrere Dateisysteme desselben Typs müssen mountbar sein.
- Daten über ein „unmount“ und anschliessendes „mount“ oder einen Reboot hinweg (auf demselben System) zu behalten. Portabilität ist explizit nicht gefordert.
- Verzeichnishierarchien bis zu einer Tiefe von mindestens 10 Verzeichnissen zu verwalten. Pro Verzeichnis sollen mindestens 100 Objekte (Dateien, Verzeichnisse, spezielle Dateien) ablegbar sein. Die Gesamtzahl möglicher Objekte im Dateisystem soll nicht unter 1000 betragen.

Zusatzanforderungen an das Dateisystem: Zusätzlich zu den oben genannten Anforderungen soll das Dateisystem eine ausgewählte der folgenden Zusatzanforderungen erfüllen:

- Integritätsprüfungen der Datenstrukturen und/oder Daten auf der Platte, beispielsweise durch Prüfsummen. Testen mittels absichtlich eingebrachter Fehler.
- Userspace-Anwendung zur Konsistenzprüfung beschädigter Dateisysteme („fsck“) und deren Rettung/Reparatur. Testen mittels absichtlich eingebrachter Fehler.
- Transparente Kompression oder blockweise Deduplizierung auf der Platte befindlicher Daten.
- Unterstützung von „extended attributes“ (`xattr(7)`) und POSIX-(Draft-)ACLs (`acl(5)`) sowie den Unix-üblichen „speziellen Dateien“ (Sockets, Devices, Pipes).
- Verbesserte Fehlertoleranz durch Einsatz von Journaling oder Copy-On-Write („Log-Structured“).

5.3 Weitere Hinweise

Die Kodierrichtlinien des Linux-Kernels müssen eingehalten werden. Synchronisierung und Speicherverwaltung müssen den üblichen Anforderungen wie der Abwesenheit von Wettlaufbedingungen (race conditions) und der Vermeidung (oder ggf. Minimierung) von Speicherlecks genügen. Oft sind im Kernel bereits Datenstrukturen oder APIs vorhanden, die den Programmierer beim Erfüllen dieser Anforderungen unterstützen. Sucht und benutzt diese bitte auch! Alles was Teil des Standard-Kernels ist ist auf jeden Fall erlaubt.

Im Kernel gibt es bereits diverse implementierte Dateisysteme. Diese dürfen als Vorlage und Inspiration zum eigenen Dateisystem gesehen werden. Jedoch ist der Funktionsumfang oft weit grösser und berücksichtigt deutlich mehr Randfälle und spezielle Situationen als für diese Aufgabe gefordert ist. Im Zweifelsfall ist eine einfache und übersichtliche Implementierung zu bevorzugen.

Aufgaben:

→ Lesen der Dokumentation, um die für das Dateisystem notwendigen Teile des Linux-Kerns zu verstehen und benutzen zu können.

→ Wahl eines geeigneten Datenformats „auf der Platte“ für gegebene Anforderungen.

→ Implementierung des Dateisystems und des `mkfs`-Userspace-Tools.

→ Auswahl und Implementierung einer Zusatzanforderung.

→ Testen des Dateisystems nach den gegebenden Anforderungen und ggf. mittels geeigneter Werkzeuge, z.B. `bonnie`, `dd` o.Ä.

5.4 Abgabe: am 2018-02-09