

### Blatt 3

## RTFS – Ein neues echtzeit-fähiges Dateisystem für das Linux VFS

Im Rahmen des vorliegenden Aufgabenblattes soll ein einfaches Dateisystem – genannt *Real-Time File System RTFS* – entwickelt werden, welches für Echtzeitanwendungen geeignet ist. Diese Eigenschaft wird dadurch gewährleistet, dass der Inhalt einer Datei im RTFS immer physikalisch zusammenhängend auf dem zugrunde liegenden Device abgelegt wird. Dadurch erfolgt das sequenzielle Lesen der Datei in konstanter Zeit, wenn der zugreifende Prozess das Device für sich exklusiv nutzen kann: Es erfolgen keine zufällig langen Plattenkopfbewegungen, wie es bei Ext2-ähnlichen Dateisystemen nötig werden kann, sobald die Datenblöcke einer Datei weit über die Partition verstreut liegen. Die Nutzung eines RTFS-Dateisystems erfolgt auf folgende Weise:

1. Das Dateisystem wird per Mount verfügbar gemacht. Für die Übung nehmen wir als Device immer ein Loop-Device, so dass der Treiber bereits verfügbar ist und zum Ausprobieren der Lösung keine Plattenpartitionierung erforderlich ist.
2. Nach dem ersten `open()` auf eine neue Datei sind Lese- und Schreiboperationen zunächst noch verboten. Es muss zuvor noch ein `ioctl()` Befehl erfolgen, der als Parameter die maximale Größe der Datei angibt. Dabei wird die Datei physikalisch auf dem Device mit Maximalgröße angelegt (wenn der erforderliche Platz dort zur Verfügung steht) und ist ab sofort beschreibbar.
3. Dateien werden physikalisch durch
  - innerhalb des Dateisystems eindeutige “Inode”-Nummer,
  - Eigenschaften der Datei (z.B. aktuelle Größe),
  - Datenblöcke

repräsentiert – alle diese Informationen liegen auf dem Device physikalisch direkt hintereinander.

4. Wenn die Datei mit Flag `O_APPEND` geöffnet wird, erfolgt der nächste Schreibbefehl am Ende der Datei hinter dem letzten beschriebenen Byte.

5. Mittels `lseek()` lässt sich die Datei an jedem Byte innerhalb der vorgegebenen Größe beschreiben. Lesebefehle, die unbeschriebene Sektionen berdecken, liefern dort 0-Bytes zurück (das ist die Standard-Semantik von `lseek()`). Anders als bei den “normalen” Dateisystemen führt ein `lseek()` auf dem RTFS-Dateisystem zu einer Fehlermeldung, wenn versucht wird, über die definierte Maximalgröße hinaus zu positionieren.
6. Directories werden – wie es auch bei Ext2/Ext3 üblich ist – durch spezielle Dateien repräsentiert. Dabei weichen wir für RTFS in folgenden Punkten von Ext2/Ext3 ab:
  - Die maximale Länge von Datei- oder Verzeichnisnamen ist 64 Bytes; für jede Datei/Unterverzeichnis im Verzeichnis wird dieser konstante Platz für den Namen reserviert.
  - Es dürfen maximal 512 Dateien in einem Verzeichnis liegen. Damit kann jedes Verzeichnis als RTFS-Datei fixer Maximalgröße repräsentiert werden.
  - Mit dem Verzeichnisnamen wird nicht nur die “Inode”-Nummer der Datei/des Unterverzeichnisses abgelegt, sondern auch die Blocknummer, unter welcher die Datei/das Unterverzeichnis auf dem Device zu finden ist.

### **Aufgabe 1: Entwurf des RTFS**

**40 %**

Entwickeln Sie ein Entwurfsdokument mit Arbeitsplan für die Realisierung des oben skizzierten RTFS. Das Dokument soll folgende Punkte detaillieren:

1. Struktur des RTFS auf einem Device – Superblock – Deskriptoren – Block-Bitliste – Directories – Nutzdaten-Dateien
2. Formatierung des Filesystems auf einem Loop-Device.
3. Mounten des Filesystems: Modul mit RTFS-Operationen und Datenstrukturen – Operation `fill_super()`
4. Erforderliche Superblock-Operationen
5. Erforderliche Inode-Operationen
6. Erforderliche Dentry-Operationen
7. Erforderliche File-Operationen
8. Abbildung der Filesystem-internen Datenstrukturen (siehe Punkt 1) auf die Superblock-/Inode-/Dentry-/File-Strukturen des Linux Kernels.

Verschiedene Operationen müssen nicht implementiert werden, z. B. Unterstützung für symbolic und hard links, sowie Zugriffsrechteverwaltung.

## **Aufgabe 2: Implementierung und Test des RTFS**

**60 %**

Implementieren und testen Sie die im obigen Arbeitsplan ausgearbeiteten Strukturen und Konzepte, so dass das Filesystem als Modul dazugeladen werden kann.

Zur Implementierung gehören mindestens

1. Ein Kernel Module, welches die Operationen des RTFS implementiert.
2. Ein Formatierungsprogramm, welches eine RTFS-Partition auf einem (Loop-)Device initialisiert.
3. Ein Testprogramm, welches die Dateioperationen auf RTFS-Dateien erprobt und automatisch auf Korrektheit prüft.

Die Abgabe soll den gut dokumentierten Quellcode sowie die Ergebnisse der Testfälle enthalten. Die Implementierung kann gruppenbergreifend durchgeführt werden.

**Abgabe: Bis Dienstag, 30.06.2009, in der Vorlesung.**

**Sowohl bei der schriftlichen Lösung als auch im Source-Code die Namen aller Gruppenmitglieder nicht vergessen!**