Übungen zur Vorlesung
Betriebssysteme II
Sommersemester 2006

AG BS
Jan Peleska

Blatt 4

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 Loop-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,
   - Bitliste der noch nicht beschriebenen/beschriebenen Blöcke,
   - 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 auf ersten freien Block hinter dem letzten beschriebenen.

5. Unabhängig von den Open-Flags lässt sich die Datei mittels `lseek()` an jedem Byte innerhalb der vorgegebenen Größe beschreiben. Lesebefehle, die unbeschriebene Sektionen überdecken, liefern dort 0-Bytes zurück (das ist die Standard-Sematik von `lseek()`). Anders als bei den "normalen" Dateisystemen führt ein `lseek()` auf dem RTFS-Dateisystem zu einer Fehlermeldung, wenn versucht wird, über das definierte Dateiende 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 auf 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 Loop-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 read_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. Mehr Details dazu in der Vorlesung und Übung am 07.06./12.06.2006.

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 Imlementierung 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 Implementierung kann gruppenübergreifend durchgeführt werden.

Abgabe: Bis Mittwoch, 12.07.2006, in der Übung.

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