

Betriebssysteme

Tutorium 11

Philipp Kirchhofer

`philipp.kirchhofer@student.kit.edu`

`http://www.stud.uni-karlsruhe.de/~uxbtt/`

**Lehrstuhl Systemarchitektur
Universität Karlsruhe (TH)**

27. Januar 2010

Was machen wir heute?

1 Tutorien Übungsblatt

- Mounting Filesystems
- Disk Space Allocation
- File System Implementation
- File System Recovery
- File System Cache

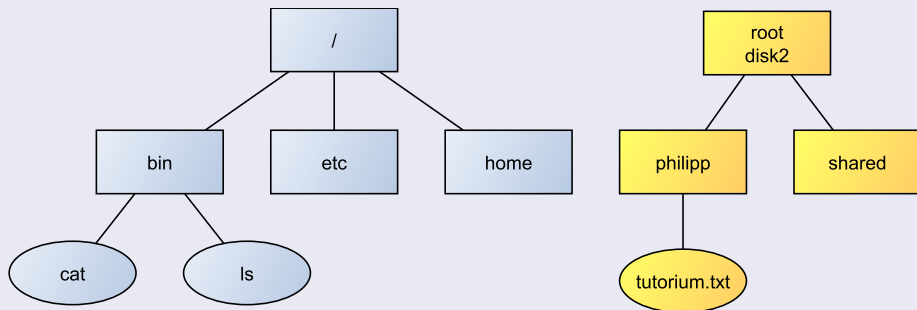
Frage 11.1.a

Was passiert beim „Mounten“ eines Dateisystems?

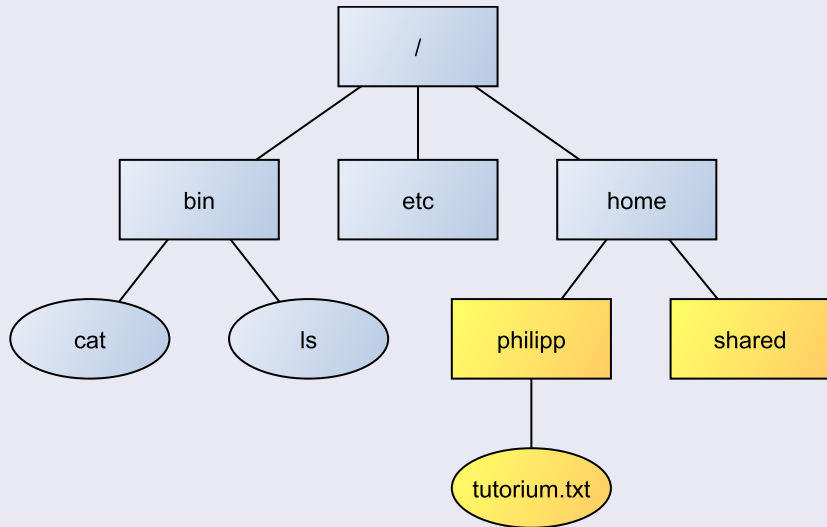
Antwort

Beim mounten wird ein vorher nicht zugreifbares Dateisystem in ein anderes Dateisystem, auf das zugegriffen werden kann eingehängt. Nach dem mounten zeigt ein Verzeichniszeiger auf das oberste Verzeichnis des neu eingehängten Dateisystems.

Mounten: Vorher



Mounten: Nachher



Frage 11.1.b

Ein XFS Dateisystem mit der Datei */foo/bar* ist im ext3 Dateisystem unter dem Pfad */mnt/extras* eingebunden. Wie verläuft der Zugriff auf die Datei */mnt/extras/foo/bar/baz*?

Frage 11.2.a

Wie funktioniert zusammenhängende (contiguous) Allokation?

Antwort

Für jede Datei wird ein zusammenhängender Bereich von Blöcken auf der Festplatte reserviert.

- Dateisystem speichert Startblock und Größe der Datei
- ⊕ Sequentieller und wahlfreier Zugriff möglich
- ⊖ Datei kann nicht größer als der reservierte Bereich werden
- ⊖ Externe Fragmentierung

Frage 11.2.b

Wie funktioniert verlinkte Allokation?

Antwort

Bei verlinkter Allokation können Dateien verteilt auf der Festplatte gespeichert werden, eine Datei muss nicht mehr am Stück vorliegen.

- Dateisystem speichert Startblock
- ⊕ Keine externe Fragmentierung
- ⊖ Nur sequentieller Zugriff auf Dateien effizient möglich

Frage 11.2.c

Was ist die grundlegende Idee einer „File Allocation Table“ (FAT)?

Antwort

Bei einem FAT Dateisystem wird verlinkte Allokation eingesetzt, allerdings werden die Verlinkungsinformationen in einer eigenen Datenstruktur (FAT) gespeichert. Die FAT kann im Hauptspeicher gecached werden.

- ⊕ Zugriff auf Dateiblöcke schneller als bei verlinkter Allokation
- ⊖ Größe der FAT ist abhängig von der Größe des Dateisystems
- ⊖ Wahlfreier Zugriff weiterhin langsam

Frage 11.2.d

Wie funktioniert indizierte (indexed) Allokation?

Antwort

Die Zeiger auf die Datenblöcke werden in einer eigene Datenstruktur, dem Indexblock, zusammengefasst. Für jede Datei gibt es einen Indexblock.

- ⊕ Schneller Zugriff auf Datenblöcke
- ⊖ Größe des Indexblocks begrenzt Dateigröße

Frage 11.2.e

Wie kann die maximale Dateigröße einer Datei bei indizierter Allokation vergrößert werden ohne die Größe des Indexblocks zu erhöhen?

Antwort

- Verlinkte Liste von Indexblöcken
- Mehrstufige Liste von Indexblöcken
ähnlich wie mehrstufige Seitentabellen

Beide Verfahren können auch miteinander kombiniert werden:

Bei einer zweistufigen Indexliste können Blöcke z.B. direkt, einfach indirekt oder zweifach indirekt abgespeichert werden.

Frage 11.2.f

Es sei ein Dateisystem auf Inode-Basis gegeben:

Ein Block auf der Festplatte sei 8 KB und ein Zeiger 4 Byte groß. Ein Inode enthält 12 Zeiger auf direkt referenzierte Blöcke und je einen Zeiger auf einen ein-, zwei- bzw. dreifach indirekt referenzierten Block.

Wie groß kann eine Datei maximal werden?

Antwort

In einem Block können $\frac{8\text{KB}}{4\text{B}} = 2048$ Zeiger gespeichert werden.

Über den einfach indirekt referenzierten Block können 2048 Blöcke adressiert werden. Über den zwei- bzw. dreifach indirekt referenzierten Block können $2048 * 2048$ bzw. $2048 * 2048 * 2048$ Blöcke adressiert werden.

Maximale Dateigröße: $(12 + 2048 + 2048^2 + 2048^3) * 8\text{KB} = 64\text{TB}$

Frage 11.3.a

Welche der folgenden Informationen werden üblicherweise in einem Inode gespeichert:

- Dateiname
- Name des Verzeichnis
- Dateigröße
- Dateityp
- Anzahl symbolischer Links auf Datei
- Name/Ort der symbolischen Links
- Anzahl Hardlinks auf Datei
- Name/Ort der Hardlinks
- Zugriffsrechte
- Zeitstempel (Letzter Zugriff, Letzte Änderung)
- Dateiinhalt
- Sortierte Liste aller Blöcke, die von der Datei belegt werden

Dateiname

Wird im Verzeichnis gespeichert (Eine Datei kann mehrere Dateinamen haben)

Name des Verzeichnis

Nicht im Inode gespeichert, eine Datei kann gleichzeitig in verschiedenen Verzeichnissen liegen (Hardlinks)

Dateigröße

Im Inode gespeichert, wird benötigt für Dateien, deren Größe nicht einem Vielfachen der Blockgröße entspricht.

Dateityp

Im Inode gespeichert, dass Dateisystem muss zwischen Verzeichnissen, Dateien und Symbolischen Links unterscheiden können.

Anzahl symbolischer Links auf Datei Name & Ort der symbolischen Links

Informationen werden nicht im Dateisystem gespeichert

Anzahl Hardlinks auf Datei Name & Ort der Hardlinks

Die Anzahl an Hardlinks wird im Inode gespeichert, verwendet um die belegten Blöcke beim Löschen der letzten Referenz freigeben zu können.
Name & Ort der Hardlinks werden nicht gespeichert.

Zugriffsrechte

Zugriffsrechte werden im Inode gespeichert, da sie den Zugriff auf Daten regeln, optional

Zeitstempel (Letzter Zugriff, Letzte Änderung)

Im Inode gespeichert, optional

Dateiinhalt

Inode enthält nur Metadaten, Dateiinhalt in Datenblöcken gespeichert, die über den Inode referenziert werden.

Sortierte Liste aller Blöcke, die von der Datei belegt werden

Im Inode gespeichert, ermöglicht Zugriff auf Dateidaten:

$$\text{Dateiblock} = \frac{\text{Offset}}{\text{Blockgröße}}$$

$$\text{Offset in Block} = \text{Offset} \bmod \text{Blockgröße}$$

Frage 11.3.b

Wie können Verzeichnisse implementiert werden?
Welche Informationen werden gespeichert?

Antwort

Verzeichnisse können als Dateien vom Typ „Verzeichnis“ implementiert sein. Die innere Struktur dieser Datei ist dem Dateisystem bekannt und besteht aus einer variablen Liste von Einträgen, in denen mindestens der Dateiname und die zugehörige Inode-Referenz gespeichert sind.

Frage 11.3.c

Was sind Hardlinks?

Antwort

Hardlinks verknüpfen Namen mit einem Inode. Jeder Verzeichniseintrag ist also ein Hardlink.

Frage 11.3.d

Was sind symbolische Links?

Antwort

Symbolische Links sind Dateien vom Typ „Symbolischer Link“. Ihr Inhalt besteht aus einem relativen oder absoluten Pfad, der vom Dateisystem interpretiert wird.

Frage 11.3.e

Wie können Links unter Linux erzeugt werden?

Antwort

ln f h erzeugt einen Hardlink von h auf f.

ln -s f s erzeugt einen symbolischen Link von s auf f.

Frage 11.3.f

Was passiert wenn die Datei *f* umbenannt wird?

Kann auf die Datei weiterhin über den Hardlink oder den symbolischen Link zugegriffen werden?

Antwort

Das Umbenennen von *f* ändert nur den Verzeichniseintrag, aber nicht den Inode selber. Dadurch kann weiterhin auf die Datei über den Hardlink zugegriffen werden. Der symbolische Link ist dagegen nicht mehr gültig, da er auf den alten, nicht mehr existierenden Namen verweist.

Frage 11.3.g

Was passiert wenn f vorher nach g kopiert und anschließend f gelöscht worden wäre?

Antwort

Der symbolische Link wäre weiterhin ungültig.
Der Hardlink h und die Datei f würden auf verschiedene Dateien verweisen.

Frage 11.3.h

Was passiert wenn anschließend eine neue Datei f erzeugt wird?

Antwort

g und h bleiben unverändert, s verweist auf die neue Datei.

Frage 11.3.i

Was ist der Vorteil von symbolischen Links?

Antwort

Mit symbolischen Links können zwei Nachteile von Hardlinks behoben werden:

- 1 Hardlinks können nur auf Dateien im gleichen Dateisystem wie der Hardlink selber verweisen, da Hardlinks direkt Inodes referenzieren.
- 2 Es können keine Hardlinks auf Dateien erstellt werden.

Frage 11.4

Die Liste der freien Blöcke in einem Dateisystem sei verloren gegangen. Wie kann die Liste neu aufgebaut werden?

Antwort

Jeder Block wird als frei markiert. Anschließend werden die Dateisystem Metadaten als belegt markiert. Zum Schluß wird rekursiv vom Hauptverzeichnis ausgehend für jede Datei die von der Datei belegten Blöcke in der Liste als belegt markiert.

Frage 11.5.a

Was ist die grundlegende Idee eines Dateisystem Caches im Hauptspeicher?

Antwort

Dateiblöcke, die in der Vergangenheit gelesen oder geschrieben wurden werden auch sehr wahrscheinlich in der Zukunft weiter verändert. Durch den Einsatz eines Caches können mehrfache Lese-/Schreibzugriffe auf den gleichen Block vermieden werden.

Frage 11.5.b

Was sind die Vor- und Nachteile einer festen Begrenzung des Dateisystem Caches?

Antwort

- ⊕ Einfach zu implementieren
- ⊖ Kann nicht dynamisch auf verschiedene Programmtypen reagieren

Frage 11.5.c

Was sind die Vor- und Nachteile von „Read ahead“ Systemen?

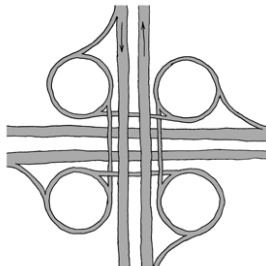
Antwort

- ⊕ Gute Performance bei sequentiellen Zugriffsmustern
- ⊕ Verbesserter Festplattendurchsatz bei nebeneinanderliegenden Dateiblöcken
- ⊖ Verschwendete E/A-Bandbreite, falls die vorgeladenen Daten nicht genutzt werden

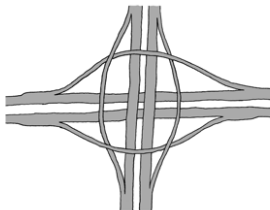
Fragen & Kommentare?

HIGHWAY ENGINEER PRANKS:

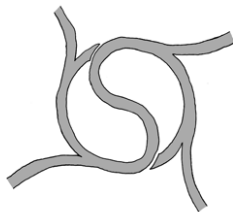
THE INESCAPABLE CLOVERLEAF:



THE ZERO-CHOICE INTERCHANGE:



THE ROTARY SUPERCOLLIDER:



xkcd: Highway Engineer Pranks