Prof. Dr. Christoph Scholl
Dr. Tim Welschehold
Alexander Konrad
Niklas Wetzel

Freiburg, 09.12.2022

Betriebssysteme
Musterlösung zu Übungsblatt 8

Aufgabe 1 (2 + 1 + 1 Punkte)

Hardlinks und symbolische Links

a) Fassen Sie kurz zusammen, was der Unterschied zwischen harten und symbolischen Links ist und nennen Sie jeweils zwei Vor- und Nachteile.

b) In der Vorlesung wurde erwähnt, dass Hardlinks nur innerhalb eines Dateisystems angelegt werden können. Was ist ein möglicher Grund dafür?

c) Unter Linux ist es Ihnen nicht erlaubt, Hardlinks auf Verzeichnisse anzulegen. Nennen Sie einen Grund, warum Hardlinks auf Verzeichnisse problematisch sind.

Lösung:

a) Unterschiede:

- Alle Hardlinks zu einer Datei verweisen auf einen einzigen I-Node. Im Gegensatz dazu hat jeder symbolische Link einen eigenen I-Node, dessen Daten verweisen auf einen Verzeichniseintrag. [Bemerkung: In der Vorlesung wurde vorgestellt, dass der I-Node eines symbolischen Link einen Zeiger auf einen Datenblock enthält, der wiederum den Pfadnamen des Ziels enthält. Bei manchen Dateisystemen (z.B. ext) wird der Pfad des Ziels auch direkt im I-Node gespeichert.]
- Ein Hardlink ist nur ein Verzeichniseintrag, jeder symbolische Link ist ein eigenständiger I-Node.
- Wird das Original gelöscht, so zeigen symbolische Links ins Leere, während über Hardlinks der Inhalt der Datei immer noch zugänglich ist.
- Wird das Original gelöscht und eine Datei mit dem selben Namen angelegt, so zeigen die symbolischen Links auf die neue Datei, während Hardlinks weiterhin auf das I-Node mit dem alten Inhalt zeigen.
- Während symbolische Links weit verbreitet sind, existieren Hardlinks nur in Dateisystemen mit I-Nodes oder ähnlichen Strukturen.
- Hardlinks können nur innerhalb des selben Dateisystems angelegt werden, symbolische Links funktionieren auch über Dateisysteme hinweg.
- Ordner können i.d.R. nur mit symbolischen Links als Ziel verwendet werden.

[0.25 Punkte pro Unterschied, maximal 1 Punkt, auf 0.5 aufrunden. Bei falschen Aussagen entsprechend abziehen.]

	Vorteile	Nachteile
symbolische Links	können auf beliebige Objekte (Dateien, Verzeichnisse, Devices usw.) zeigen	zeigt nach Löschen oder Verschieben des Originals ins Leere
	können auf Objekte in anderen Dateisystemen zeigen	Anzahl der Links auf eine Datei nur durch Suche bestimmbar
	Linkziel sichtbar im Dateibrowser / per `ls -l`	Zugriff auf Zieldatei ist aufwendiger, da der komplette Linkpfad nachverfolgt werden muss
	Existiert für eine Vielzahl von Dateisystemen	
Hardlinks	bleibt bei Löschen oder Verschieben des Originals gültig	können nicht auf Verzeichnisse zeigen
	Anzahl der Links auf eine Datei im I-Node gespeichert	nur innerhalb eines Dateisystems möglich
	Zugriff auf Zieldatei sehr effizient, da der Hardlink direkt auf den I-Node verweist	mit `ls -l` nicht erkennbar, welche Links auf die selbe Datei zeigen
	geringerer Speicherplatzverbrauch als bei symbolischen Links, da bei der Erstellung eines Hardlinks nur ein Verzeichniseintrag hinzugefügt wird	Nur in Dateisystemen mit I-Nodes oder ähnlichen Strukturen verfügbar

[0.125 Punkte für jeden Vor- und jeden Nachteil, maximal 1 Punkt, auf 0.5 aufrunden. Bei falschen Aussagen entsprechend abziehen.]

b) Wenn dies möglich wäre, müsste man zusätzlich zum I-Node abspeichern, in welchem Dateisystem/in welcher Partition das Ziel liegt. Das wiederum macht aber keinen Sinn, da die Dateisysteme an verschiedenen Stellen, zu unterschiedlichen Zeiten und möglicherweise von unterschiedlichen Computern gemountet werden könnten und damit könnte dies zu unerwartetem Verhalten führen.

Nehmen wir an, es würde eine Datei A erstellt und es verweisen zwei Hardlinks von unterschiedlichen Dateisystemen auf diese Datei. In welchem Dateisystem befinden sich nun tatsächlich die Daten? Was muss man tun, wenn ein Dateisystem nicht mehr mit dem Rechner verbunden ist? Sind die Daten noch vorhanden? Wenn ja, kann ich sie löschen? [1]

c) Erstelle in Verzeichnis `D1` ein Verzeichnis `A`. Nun erstelle in Verzeichnis `A` mit `ln ../A B` einen Hardlink `B` auf `A`. Wechsle nun mit `cd B` in das Verzeichnis. Man befindet sich nun gleichzeitig in `D1` und `A`. Was soll nun passieren wenn man `cd ..` eingibt? Das Verzeichnis hat zwei Vaterverzeichnisse (`D1,A`). Wie soll das Dateisystem wissen, welches ausgewählt werden soll?

Es gibt auch noch andere Probleme, z.B. gehen UNIX-Befehle immer von einer azyklischen Verzeichnisstruktur aus. Ein Zyklus könnte deshalb zu Endlosschleifen führen. [1]

Hinweis zur Korrektur:
Bei den Teilaufgaben b) und c) kommt es darauf an, dass die Studenten sich Gedanken über das Thema machen und einen plausiblen Grund angeben (mit Begründung). Die genanten Gründe müssen nicht mit der Musterlösung übereinstimmen.

Aufgabe 2 (2 + 2 Punkte)

Es ist in der Praxis üblich, die Größe einer Datenmenge in KB (auch KByte, „Kilobyte“), MB (auch MByte, „Megabyte“) oder GB (auch GByte, „Gigabyte“) anzugeben.

Anders als sonst bezeichnet ein Kilo hier aber in der Regel nicht 1000, sondern $2^{10} = 1024$ Einheiten, ein Mega nicht 1000000, sondern $2^{20} = 1048576$ Einheiten usw. Ein KB entspricht daher genau 2^{10} Byte, ein MB entspricht 2^{20} Byte und ein GB entspricht 2^{30} Byte. Die Notation und Interpretation ist jedoch oft uneinheitlich.

Es gibt Normierungsbestrebungen, die versuchen, die in diesem Zusammenhang häufig auftretende Verwirrung zu beseitigen. So wurden von der IEC 1998 die Bezeichnungen KiB, MiB etc. eingeführt (vgl. `https://de.wikipedia.org/wiki/Binärpräfix`):

SI-Dezimalpräfixe		
k	Kilo	$10^3 = 1.000$
M	Mega	$10^6 = 1.000.000$
G	Giga	10^9
T	Tera	10^{12}

IEC-Binärpräfixe		
Ki	Kibi	$2^{10} = 1.024$
Mi	Mebi	$2^{20} = 1.048.576$
Gi	Gibi	2^{30}
Ti	Tebi	2^{40}

Wir verwenden bei der Angabe von Datei- und Festplattengrößen KiB, MiB und GiB im Sinne von 2^{10}, 2^{20} bzw. 2^{30} Byte.

a) Vervollständigen Sie folgende Tabelle, indem Sie die Größenangaben in Bit und Byte (=8 Bit) umrechnen und in 2er-Potenzen und in Dezimaldarstellung angeben.

	Angabe in Bits		Angabe in Bytes	
Angabe	2er-Potenz	dezimal	2er-Potenz	dezimal
2 Byte	2^4 Bit	16 Bit	2^1 Byte	2 Byte
2048 MiB				
32 Byte				
16 MiBit				
1024 KiBit				

b) Sie haben eine „3,0 TB“-Festplatte gekauft. Was glauben Sie, welche Interpretation von „TB“ der Festplattenhersteller gewählt hat? Wie groß ist der Unterschied (in GiB) zwischen den Interpretationen?

Lösung:

a) Umrechnung der Größenangaben:

	Angabe in Bits		Angabe in Bytes	
Angabe	2er-Potenz	dezimal	2er-Potenz	dezimal
2 Byte	2^4 Bit	16 Bit	2^1 Byte	2 Byte
2048 MiB	2^{34} Bit	17.179.869.184 Bit	2^{31} Byte	2.147.483.648 Byte
32 Byte	2^8 Bit	256 Bit	2^5 Byte	32 Byte
16 MiBit	2^{24} Bit	16.777.216 Bit	2^{21} Byte	2.097.152 Byte
1024 KiBit	2^{20} Bit	1.048.576 Bit	2^{17} Byte	131.072 Byte

0,5 Punkte für die korrekte Angabe aller 2er-Potenzen und 0,5 Punkte für die korrekte Angabe aller Dezimalwerte.

b) Die Einheit TB bezeichnet hier typischerweise 10^{12} Bytes, da die Festplattenkapazität in SI-Einheiten „größer“ aussieht als in Zweierpotenz-Einheiten [0.5]. Im Gegensatz dazu ergibt sich für Arbeitsspeicher wegen der parallelen Adressierung immer eine Zweierpotenz, weshalb Arbeitsspeicher fast immer mit Binärpräfix angegeben wird.

Differenz der Intepretationen:

$3{,}0 \cdot 2^{40}$ Byte $-3{,}0 \cdot 10^{12}$ Byte $= 298534883328 = 278{,}032$ GiB [0.5]

Aufgabe 3 (1 + 2 Punkte)

In der Vorlesung haben Sie das FAT32-Dateisystem kennengelernt. Es ist eine FAT-Tabelle mit einer Blockgröße von 32 KB gegeben. Zusätzlich steht eine Liste aller freien Plattenblöcke und eine Tabelle mit den Verzeichniseinträgen zur Verfügung:

FAT:

Plattenblock 0	
Plattenblock 1	
Plattenblock 2	10
Plattenblock 3	11
Plattenblock 4	7
Plattenblock 5	
Plattenblock 6	3
Plattenblock 7	2
Plattenblock 8	
Plattenblock 9	
Plattenblock 10	12
Plattenblock 11	14
Plattenblock 12	−1
Plattenblock 13	
Plattenblock 14	−1
Plattenblock 15	
⋮	⋮

Liste freier Plattenblöcke:

15	13	1	8	9	5	0	...

Verzeichniseinträge:

Dateiname	Erweiterung	Datei-Attribute	Erster Plattenblock	Dateigröße
`BRIEF`	`TXT`	(...)	`4`	`129 KB`
`EDITOR`	`EXE`	(...)	`6`	`101 KB`
⋮	⋮	⋮	⋮	⋮

a) Sie sehen, dass im FAT32-Dateisystem im Gegensatz zu inode-basierten Dateisystemen die Dateiattribute im Verzeichnis stehen. Bei FAT32 gibt es zudem keine Hardlinks. Welches Problem würde in diesem Zusammenhang entstehen, wenn man in FAT32 Hardlinks erlauben würde?

b) Der Benutzer legt eine neue Datei „`AUFGABE.DOC`“ mit einer Dateigröße von 158 KB an. Geben Sie die FAT, die Liste der freien Plattenblöcke und die Tabelle der Verzeichniseinträge nach dem Anlegen der Datei an. Die freien Blöcke werden in der Reihenfolge belegt, wie sie in der Liste der freien Plattenblöcke gegeben ist.

Lösung:

a) Ein Hardlink in einem anderen Verzeichnis hätte einen eigenen Verzeichniseintrag. Wird etwas an einer Datei und somit am Verzeichniseintrag verändert (z.B gelöscht), müsste auch der Eintrag des Hardlinks entsprechend verändert werden, dafür müssten aber alle Hardlinks voneinander wissen. [1P]

b) Es werden $\left\lceil \frac{158\,\text{KB}}{32\,\text{KB/Block}} \right\rceil = 5$ Blöcke benötigt. Situation nach Anlegen der Datei:

FAT:

Plattenblock 0	
Plattenblock 1	8
Plattenblock 2	10
Plattenblock 3	11
Plattenblock 4	7
Plattenblock 5	
Plattenblock 6	3
Plattenblock 7	2
Plattenblock 8	9
Plattenblock 9	−1
Plattenblock 10	12
Plattenblock 11	14
Plattenblock 12	−1
Plattenblock 13	1
Plattenblock 14	−1
Plattenblock 15	13
⋮	⋮

Liste freier Plattenblöcke:

5	0	...

Verzeichniseinträge:

Dateiname	Erweiterung	Datei-Attribute	Erster Plattenblock	Dateigröße
BRIEF	TXT	(...)	4	129 KB
EDITOR	EXE	(...)	6	101 KB
AUFGABE	DOC	(...)	15	158 KB
⋮	⋮	⋮	⋮	⋮

Wie die Liste der freien Plattenböcke angegeben wird, ist hier egal (Blöcke durchstreichen, durch -1 ersetzen, löschen), solange klar wird, welche Blöcke frei bzw. belegt sind. 0,5 Punkte für die richtige Anzahl benötigter Blöcke, weitere 1,5 Punkte für richtige Belegung.

Aufgabe 4 (2 + 2 Punkte)

I-Nodes: Maximale Dateigröße

In der Vorlesung wurden I-Nodes und ihre Struktur bei dem Betriebssystem „System V“ vorgestellt: Es verfügt über 10 direkte Zeiger und je einen Zeiger auf einen ein-, zwei- und dreifach indirekten Block.

a) Geben Sie eine Formel an für die maximale Anzahl von frei verfügbaren Dateiblöcken N_b pro I-Node in Abhängigkeit von der Zeigergröße z und der Blockgröße b. Gehen Sie bei Ihren Berechnungen davon aus, dass die Zeigergröße immer ausreichend groß gewählt wurde, sodass die Anzahl der eindeutig adressierbaren Blöcke keine Rolle spielt.

b) Berechnen Sie für die Blockgrößen 1 und 4 KB die jeweils maximale Größe einer Datei auf diesem System, wenn die Zeigergröße 4 Byte beträgt. Wie groß ist die maximale Zahl aller Datenblöcke des Dateisystems, die eindeutig adressiert werden können?

Lösung:

a) Bei der 1-/2-/3-fach indirekten Adressierung passen $\left\lfloor \frac{b}{z} \right\rfloor$ Zeiger in einen Block [0.5]. Die Anzahl der Datenblöcke, die ein I-Node adressieren kann, ist daher [1]:

$$N_b = 10 + \left\lfloor \frac{b}{z} \right\rfloor + \left\lfloor \frac{b}{z} \right\rfloor^2 + \left\lfloor \frac{b}{z} \right\rfloor^3 = 10 + \sum_{i=1}^{3} \left\lfloor \frac{b}{z} \right\rfloor^i \tag{1}$$

b) Maximale Dateigrößen

- Für Blockgröße 1 KiB:
  Anzahl Zeiger pro Block:
  $$\left\lfloor \frac{b}{z} \right\rfloor = \frac{1 \frac{\text{KiB}}{\text{Block}}}{4 \frac{\text{Byte}}{\text{Zeiger}}} = 256 \frac{\text{Zeiger}}{\text{Block}}$$
  Maximale Anzahl der adressierbaren Datenblöcke pro I-Node:
  $$N_b = 10 + 256 + 256^2 + 256^3 = 10 + 256 + 65536 + 16777216 = 16843018$$
  Maximale Größe einer Datei [0.5]:
  $$16843018\,\text{Blöcke} \cdot 1 \frac{\text{KiB}}{\text{Block}} = 17247250432\ \text{Byte} = 16843018\ \text{KiB} \approx 16448\ \text{MiB} \approx 16{,}06\ \text{GiB}$$
- Für Blockgröße 4 KiB:
  Anzahl Zeiger pro Block:
  $$\left\lfloor \frac{b}{z} \right\rfloor = \frac{4 \frac{\text{KiB}}{\text{Block}}}{4 \frac{\text{Byte}}{\text{Zeiger}}} = 1024 \frac{\text{Zeiger}}{\text{Block}}$$
  Maximale Anzahl der adressierbaren Datenblöcke pro I-Node:
  $$N_b = 10 + 1024 + 1024^2 + 1024^3 = 10 + 1024 + 1048576 + 1073741824 = 1074791434$$
  Maximale Größe einer Datei [0.5]:
  $$1074791434\,\text{Blöcke} \cdot 4 \frac{\text{KiB}}{\text{Block}} = 4402345713664\ \text{Byte} = 4299165736\ \text{KiB} \approx 4100\ \text{GiB} \approx 4{,}00\ \text{TiB}$$

Aufgrund der gewählten Zeigergröße von 4 Byte können maximal 2^{32} Blöcke = 4294967296 Blöcke adressiert werden [0.5].

Abgabe: als PDF im Übungsportal bis 16.12.2022 um 12:00