TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

Betriebssysteme und Sicherheit, WS 2018/19

8. Aufgabenblatt – Dateisysteme

Geplante Bearbeitungszeit: eine Woche

Aufgabe 8.1 In klassischen Unix-Dateisystemen gibt es „harte“ und „symbolische“ Verknüpfungen (hard links, symbolic links). Worin besteht der Unterschied zwischen beiden? Welche Rolle spielen Inodes (auch I-Nodes oder I-Knoten genannt) in diesem Zusammenhang?

Aufgabe 8.2 In einem Unix-Dateisystem beträgt die Blockgröße 4 KiB. Eine Allokations-Bitmap dient der Verwaltung belegter Blöcke. In diesem Dateisystem befindet sich eine Datei `/tmp/A` mit einer Länge von 3000 Byte, welche von einem Programm zum Schreiben geöffnet wird. Das Programm hängt an die bereits vorhandenen Inhalte der Datei weitere 16 KiB Daten an und schließt die Datei danach wieder.

(a) Welche Blöcke muss das Dateisystem lesen, wenn die Datei wie oben beschrieben geöffnet wird?

(b) Welche Änderungen müssen an den Metadaten in den Dateisystemstrukturen vorgenommen werden, um die neu geschriebenen Inhalte im Dateisystem abzulegen?

(c) Was wäre eine sichere Reihenfolge in der alle modifizierten und neuen Blöcke geschrieben werden können, wenn man die in der Vorlesung diskutierte Methode des „Synchronen Schreibens“ verwendet? Welche Inkonsistenzen können bei Abstürzen nach wie vor auftreten und wie können diese korrigiert werden.

(d) Welche Vor- und Nachteile hat der Einsatz des Journaling-Verfahrens gegenüber„Synchronem Schreiben“? Erläutern Sie das Verfahren am Beispiel der Dateioperationen aus Teilaufgabe (c).

Aufgabe 8.3 Eine Anwendung muss umfangreiche Änderungen an einem bereits vorhandenen großen Dokument speichern. Der Entwickler der Anwendung möchte dabei sicherstellen, dass die Änderungen atomar vorgenommen werden. Das heißt, wenn das System während der Speicheroperation abstürzt, soll die Dokumentendatei nach dem anschließenden Neustart entweder den alten oder den neuen Inhalt haben, nicht aber eine Mischung aus beiden. Dazu legt das Programm zunächst eine Kopie der neuen Datei unter einem temporären Namen an und ersetzt anschließend die alte Version durch die neue Datei.

(a) Welche Dateisystemoperationen muss das Programm in welcher Reihenfolge durchführen um die geforderte Atomizität zu erreichen? Welche Randbedingungen müssen außerdem gelten und gegebenenfalls vom Programm durch geeignete Systemaufrufe durchgesetzt werden?

(b) Welche Annahme trifft der Entwickler über die Semantik des `rename()`-Systemaufrufs? Gibt es im Hinblick auf `rename()` prinzipielle Unterschiede zwischen den in der Vorlesung diskutierten Verfahren „Synchrones Schreiben“, „Soft Updates“, „Journaling“ und dem log-strukturierten Ansatz für Dateisystemkonsistenz?

Klausuraufgabe I

In einem Unix-artigen Dateisystem werden alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium organisiert. Die Größe eines Blocks beträgt 4 KiB.

a) Wie groß muss die Bitmap mit den belegt/frei-Bits für alle Blöcke (inklusive der für die Allokations-Bitmap selbst verwendeten Blöcke) sein, wenn das Dateisystem auf einem 4 GiB großen Speichermedium erzeugt wurde?

b) Unix-Dateisysteme verwalten pro Datei jeweils ein sogenanntes „Inode". Nennen Sie zwei verschiedene Arten von Metadaten, die in jedem Inode abgelegt sind. Erläutern Sie, welchen Vorteil Unix-Dateisysteme daraus ziehen, dass der Name einer Datei *nicht* im Inode gespeichert wird.

Ein Anwendungsprogramm hat eine neue, 42 KiB große Datei erzeugt. Im Zuge dieser Operation wurden die nachfolgend aufgelisteten Metadatenblöcke im Buffer Cache des Betriebssystems modifiziert bzw. neu allokiert und zunächst gepuffert: 1 Superblock (SB), 1 Allokations-Block für Inodes (IA), 2 Allokationsblöcke für Blöcke (BA), 2 Inode-Blöcke (I_1 und I_2), 1 Verzeichnisblock (V), 1 Indirektionsblock mit Zeigern auf Datenblöcke (Z), 11 Datenblöcke (D)

HINWEIS: I_1 enthält das neue Inode der angelegten Datei, I_2 das geänderte Inode des Elternverzeichnisses.

c) Geben Sie eine nach der Methode des „Synchronen Schreibens" sichere Reihenfolge an, in der alle genannten Blöcke auf das Speichermedium geschrieben werden können, so dass das Dateisystem nach einem Absturz keine kritischen Inkonsistenzen aufweist. Markieren Sie in Ihrer Lösung die Blockschreiboperation(en) innerhalb der Sequenz, nach denen eine „Write Barrier" für die Konsistenz nach einem Absturz *notwendig* ist.

Klausuraufgabe II

Ein Unix-artiges Dateisystem organisiert alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium. Blöcke enthalten entweder Daten (D), Inodes (I), Verzeichniseinträge (V), eine Allokations-Bitmap für Inodes bzw. Blöcke (IA bzw. BA) oder den Superblock (SB). Die Größe eines Blocks beträgt 4 KiByte. Der initiale Zustand des Dateisystems ist wie folgt:

Block	0	1	2	3	4	5	6	7	8	9	10
Typ	SB	IA	BA	I	V	D	D				
Inhalt	/: 0 frei: 4	0 – 1	0 – 6	0: 4 1: 5,6	A: 1	Nutz-daten	Nutz-daten				

Inode-Zeiger für / und Anzahl freier Blöcke

Inodes 0 und 1 belegt

Block 0 bis 6 belegt

Blockzeiger in Inodes 0 und 1

Abbildung des Dateinamens *A* auf zugehöriges Inode 1

a) Welche Blöcke muss das Betriebssystem lesen, wenn der Nutzer mit dem Kommando ln /A /B einen weiteren Hardlink für die Datei *A* im Wurzelverzeichnis anlegt?

b) Nach dem Anlegen des Hardlinks erzeugt der Nutzer mit Hilfe eines Anwendungsprogramms im Wurzelverzeichnis eine neue Datei *C* mit einer Größe von 5144 Byte. Tragen Sie in das unten stehende Schema den Zustand des Dateisystems nach Abschluss dieser Operation ein. Kennzeichnen Sie alle Blöcke, deren Inhalt sich gegenüber dem oben abgebildeten Schema geändert hat.
HINWEIS: In Inode- und Verzeichnisblöcken sei ausreichend Speicherplatz vorhanden.

Block	0	1	2	3	4	5	6	7	8	9	10
Typ											
Inhalt											
geändert											

Zur Leistungssteigerung puffert das Betriebssystem neue oder modifizierte Blöcke zunächst im Hauptspeicher und schreibt diese später im Hintergrund auf das Speichermedium. Dabei werden Blöcke bestimmter, aber nicht aller, Typen zunächst in ein Log (oder Journal) geschrieben. Für das Log sind die Blöcke 11–16 exklusiv reserviert.

c) Tragen Sie in unten stehendes Schema eine mögliche Journal-Transaktion ein, bei der sichergestellt ist, dass Datei *C* nach einem Absturz korrekt und vollständig wiederherstellbar ist. Erläutern Sie, wann die jeweiligen „in-place"-Kopien (Blocknummern 0–10) geschrieben werden und warum nicht alle Blocktypen (welche?) ins Log geschrieben werden müssen.

Block	11	12	13	14	15	16
Typ						