Betriebssysteme 2
Übungsblatt 3

Felix J. Oppermann

18. November 2007

Aufgabe 3.4 *Indexsequentielle Dateiorganisation*

Gegeben sind drei Dateien, die jeweils in mehreren Blöcken auf der Festplatte gespeichert sind:

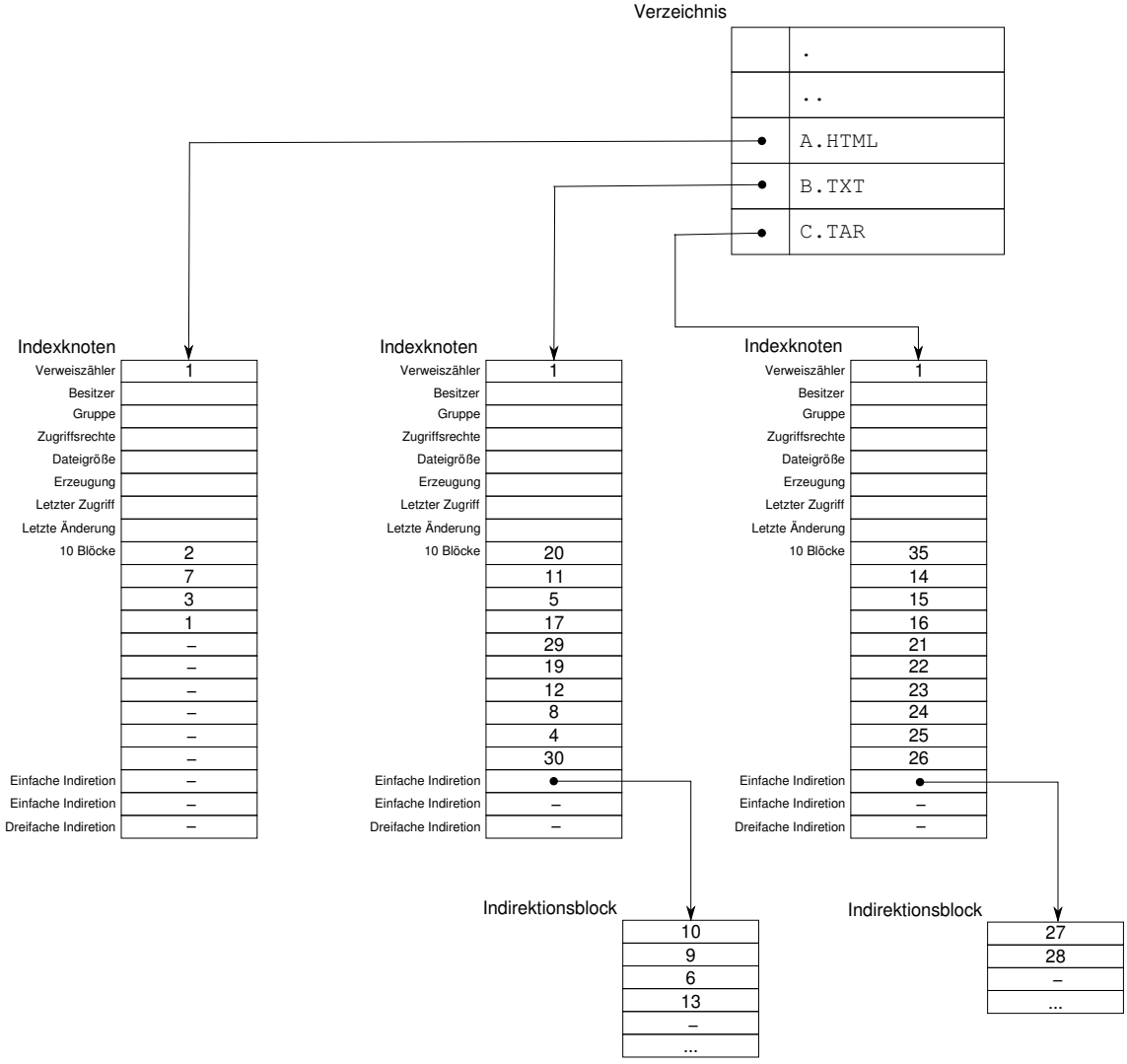
A.HTML 2,7,3,1

B.TXT 20,11,5,17,29,19,12,8,4,30,10,9,6,13

C.TAR 35,14,15,16,21,22,23,24,25,26,27,28

Die Reihenfolge spiegelt die logische Reihenfolge der Blocknummern wider.

a) *Gehen Sie von einem UNIX-artigen Dateisystem aus, wie es in der Vorlesung vorgestellt wurde, bei dem die ersten 10 Plattenblöcke im Indexknoten selbst abgespeichert werden. Skizzieren Sie hierfür das interne Aussehen eines Verzeichnisses, das diese drei Dateien enthält sowie den Aufbau der drei zugehörigen I-Nodes sowie Indirektionsblöcke.*

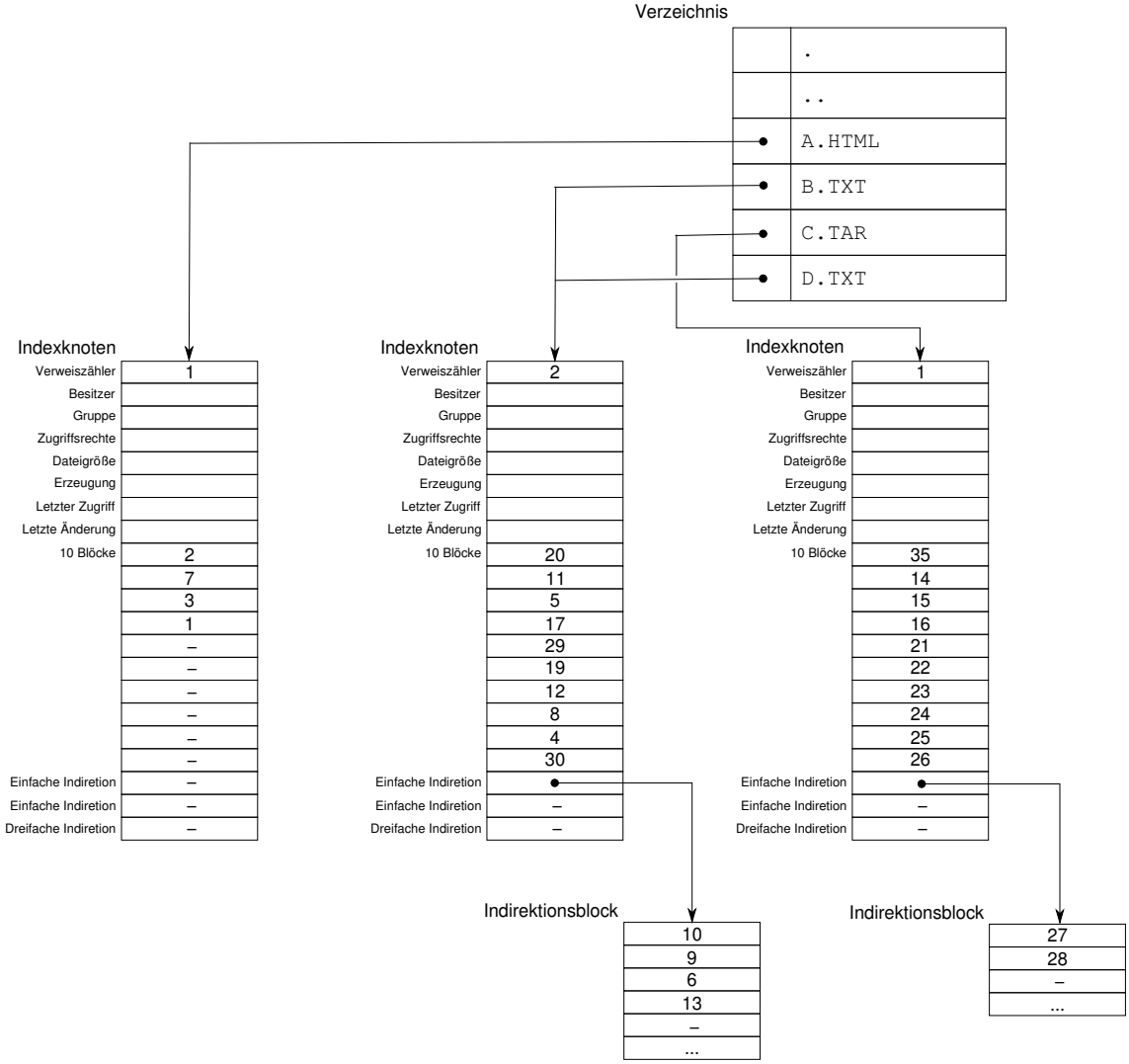


b) *Die folgenden Paare beschreiben Zugriffe auf Dateinamen und Bytenummern: (`A.HTML`,600),(`A.HTML`,2048), (`B.TXT`,6000),(`B.TXT`,20000), (`C.TAR`,2047),(`C.TAR`,7100) Die Bytenummern sind relativ zum Dateianfang 0 angegeben und die physikalische Blockgröße beträgt 2 KiByte (2^{11} Byte). Geben sie für jeden der sechs Zugriffe an, auf welche physikalischen Blocknummern zugegriffen wird.*

- **(`A.HTML`,600)** Block 2

- (`A.HTML`,**2048**) Block 7
- (`B.TXT`,**6000**) Block 11
- (`B.TXT`,**20000**) Block 4
- (`C.TAR`,**2047**) Block 35
- (`C.TAR`,**7100**) Block 15

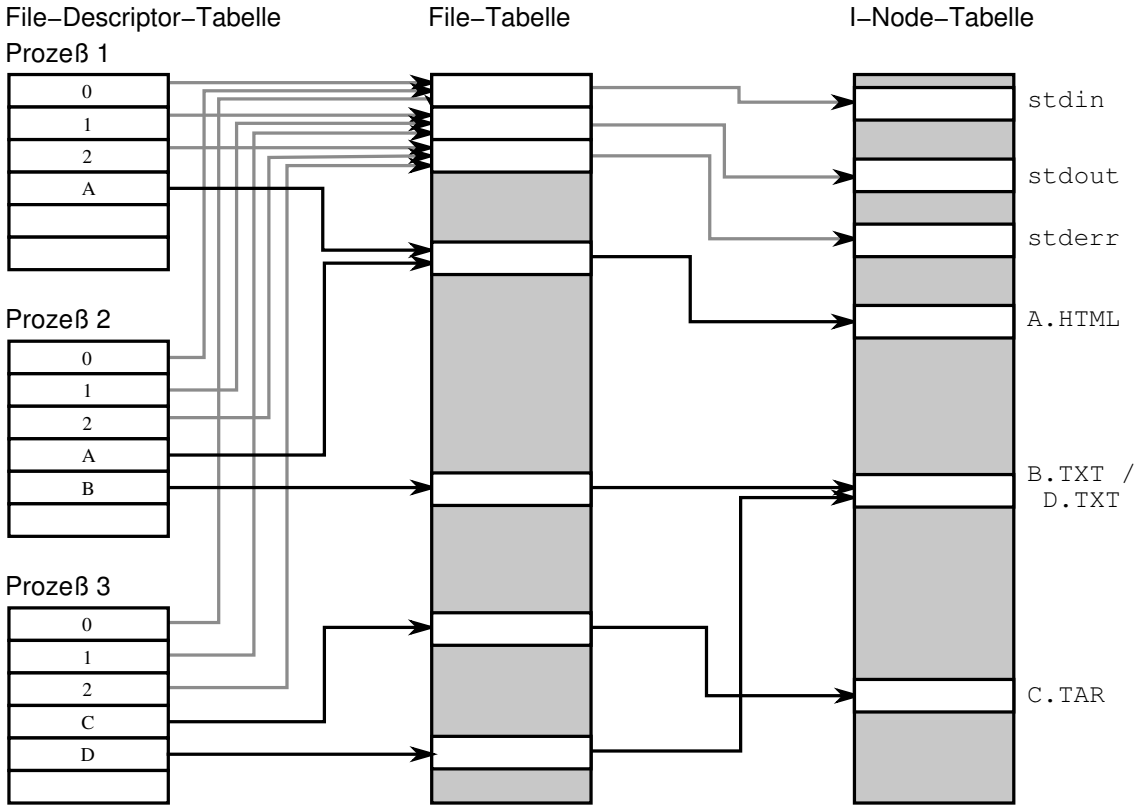
c) *Es wird ein neuer* Hard-Link *`D.TXT` erzeugt, der die gleiche Datei wie `B.TXT` bezeichnen soll. Skizzieren Sie die notwendigen Änderungen an den Strukturen.*



d) *Gegeben ist folgende Konstellation:*

- *Prozeß 1 hat die Datein `A.HTML` geöffnet und danach einen Kindprozeß 2 erzeugt, der zusätzlich die Datei `B.TXT` geöffnet hat.*
- *Prozeß 3 hat die Dateien `C.TAR` und D.TXT geöffnet.*

Skizzieren Sie ausschnittsweise die Datenstrukturen, die UNIX in dieser Situation zur Verwaltung von geöffneten Dateien benutzt.



e) *Nach einem Rechnerabsturz kann sich das Dateisystem in einem inkonsistenten Zustand befinden. Geben die detailiert Bedingungen an, denen ein konsistentes Dateisystem genügen muß.*

Um ein Dateisystem als konsistent bezeichnen zu können, müssen die Metadaten des Dateisystems widerspruchsfrei sein. Fehler in den Daten können in der Regel dennoch bestehen. Zwei Bereiche in den Widersprüche auftreten können sind zu unterscheiden: Zum Einen die Zuordnung von Dateisystemblöcken und zum Anderen die Verzeichnisstruktur.

In einem konsistenten Dateisystem muss jeder Block genau einer Datei (oder genauer einer I-node) zugeordnet sein oder er muss als frei verzeichnet sein. Bei einem Inkonsistenten Dateisystem sind drei Fehler denkbar:

- Ein Block ist meheren Dateien zugeordnet,
- ein Block ist einer Datei zugeordnet aber auch als frei markiert,
- ein Block ist keiner Datei zugeordnet aber nicht als frei markiert.

Um die Verzeichnisstruktur als konsisten betrachten zu können dürfen alle alle Verzeichnisse nur Verweise auf gültige I-Nodes enthalten. Zudem muss jede I-Node durch mindestens einen Verzeichniseintrag referenziert werden und Link-Counter der I-Nodes muss der Zahl der Verweise entsprechen. Mögliche Fehler sind:

- I-Nodes, die in keinen Verzeichnis enthalten sind und deren Daten folglich nicht mehr zugreifbar sind,
- I-Nodes mit zu geringen Verweiszähler, welche freigegeben werden können obwohl noch Verweise bestehen,
- I-Nodes mit zu hohem Verweiszähler, welche auch beim löschen des letzten Verweis nicht freigegeben würden.

Neben den genannten Anforderungen müssen alle Attribute der verwendeten Strukturen gültige Werte enthalten. Beispielsweise dürfen die Dateinahmen keine ungültigen Zeichen enthalten.

f) *Welchen Vorteil bietet die Organisation der Freispeicherliste als Bitmap gegenüber der Organisationsform, wie sie bei FAT realisiert ist? Beachten Sie dabei folgende Kriterien:*

- *Speicherplatzverbrauch*
- *Belegung eines neuen Blocke*
- *Belegung mehrerer neuer Blöcken*
- *Verhalten im Fehlerfall, Rekonstruktion*

Als Bitmap realisierte Freispeicherlisten werden in der Regel bei I-Node-basierten Dateisystemen eingesetzt. Diese verwenden mit der dateiorientierten Blockverwaltung einen grundlegenden anderen Ansatz als das FAT-Dateisystem mit einer speicherorientierten Organisation.

Beim Dateisystem FAT werden freie Blöcke durch eine spezielle Markierung in der zur FAT markiert, so dass auf eine explizite Freispeicherverwaltung verzichtet werden kann. Bei I-Node-basierten Dateisystemen sind die Informationen zur Blockbelegung nicht direkt verfügbar, so dass zusätzliche Verwaltungsinformationen benötigt werden.

Betrachtet man ausschließlich den zur Verwaltung der Freispeicherliste benötigten Speicher, so müsste man den von durch das FAT-Dateisystem verwendten Ansatz als günstiger betrachten, da hier zur Verwaltung freier Blöcke kein zusätzlicher Speicher benötigt wird. Insgesammmt ist der Specherbedarf der Verwaltungsstrukturen bei FAT jedoch in der Regel höher als bei auf I-Nodes basierenden Dateisystemen. Zudem muss beim FAT-Dateisystem die FAT in der Regel vollständig im Speicher gehalten werden um einen effizienten Zugriff zu ermöglichen. Bei I-Node-basierten Dateisystemen genügt es die I-Nodes geöffnete Dateien und ggf. die zur Verwaltung freier Blöcke benötigten Strukturen im Speicher zu halten. Der Speicherbedarf letzter ist erheblich geringer als die Speicherplatzanforderungen der FAT.

Eine Suche nach freien Blöcken erfordert bei FAT eine Suche nach als frei markierten Einträgen in der FAT. Bei dieser Suche müssen auch alle zu bereits belegten Blöcken gehörenden Einträge untersucht werden. Da die FAT vollständig im Arbeitsspeicher gehalten wird erfordert die Suche nach freien Blöcken keine zusätzlichen Zugriffe auf den Hintergrundspeicher. Die Datenstruktur einer Bitmap ermöglicht eine effizientere Suche, so dass der zur Blockbelegung nötige Aufwand geringer ausfällt. Statt vollständigen Strukturen müssen nur einzelne Bits in der Bitmap betrachtet werden. Aufgrund des verhältnismäßig geringen Speicherbedarfs kann die Bitmap ebenfalls im Hauptspeicher gehalten werden.

Um bei der Belegung mehrer zusammengehöriger Blöcke einen möglichst geringen Abstand zu erreichen um später beim Lesen unnötige Armbewegungen zu erreichen ist eine Datenstruktur, die vollständig im Hauptspeicher gehalten werden kann. Durch die relativ geringe Größe kann dies für die Freispeicherbitmap in der Regel erfüllt werden. Da jedoch auch die FAT in der Regel vollständig im Hauptspeicher gehalten wird, stellt dieses keinen Vorteil bei der Verwendung einer Bitmap dar. Für sehr Große Dateisystem ist dies jedoch bei Verwendung einer FAT ggf. nicht mehr praktikabel (siehe Windows 98), so dass die günstige Belegung von neuen Blöcken erschwert wird.

Die in der als Bitmap realisieren Freispeicherliste enthaltenen Informationen lassen sich vollständig rekonstruieren, sofern alle anderen Verwaltungsinformationen fehlerfrei geblieben sind. Die freien Blöcke können auch anhand der Zuordnung von Blöcken zu I-Nodes ermittelt werden. Die Bitmap dient nur einem schnelleren Zugriff auf diese Information. Bei FAT wird eine gemeinsamme Datenstrukturen zur Verwaltung freier Blöcke und der Zuordnung von belegten Blöcken zu Dateien verwendet. Die Information über freie Blöcke liegt nicht redundant vor und es besteht keine Rekonstruktionsmöglichkeit.