Proseminar Speicher- und Dateisysteme

Sommersemester 2012

Einführung in Dateisysteme

Malte Hamann

29.08.2012

Betreuer: Michael Kuhn

Inhaltsverzeichnis

1. Einführung

Dateisysteme sind in unserer heutigen technisierten Welt überall vorhanden und vor allem grundlegende Vorraussetzung für das Funktionieren vieler Geräte. Sie organisieren auf jedem technischen Speicher die dort vorhandenen Dateien. Trotzdem haben viele Nutzer wahrscheinlich noch nicht einmal davon gehört, dass es Dateisysteme gibt. In dieser Ausarbeitung werde ich nach einer kleinen Einführung die theoretischen Grundlagen erläutern, Praxisbeispiele geben und weitere Aspekte wie zum Beispiel Sicherheit bei Dateisystemen und Zukunftsaussichten diskutieren.

1.1. Geschichte

Dateisysteme sind älter als viele vermuten werden. Bereits im 18. Jahrhundert gab es sehr einfache Dateisysteme für Lochkarten und Lochstreifen, die zum Beispiel für die Steuerung von automatisierten Webstühlen verwendet wurden. Später kamen dann Magnetbänder hinzu, die aber mit ähnlich einfachen Systemen auskamen. Über viele Jahre blieb es bei diesen einfachen Dateisystemen und erst als die Speichermedien komplexer wurden, mussten neue, fortgeschrittenere Dateisysteme entwickelt werden. Dies geschah mit der Einführung von Trommelspeichern und später Festplatten. Im Gegensatz zum sequentiellen Einlesen der alten Speichermedien Lochkarte, Lochstreifen und Magnetband war nun ein direkter Zugriff auf alle Speicherbereiche möglich, was neue Anforderungen an Dateisysteme stellte.

1.2. Wo wären wir ohne Dateisysteme?

Wenn man sich überlegt, was man von einem Speichermedium ohne Dateisystem hat, ist klar, dass man nur noch das Speichermedium hat aber keine sinnvolle Verwaltung der Daten da drauf. Das einzige, was einem bleibt ist die Physikalische Speicheradresse, die man entweder klassisch als Cylinder, Head und Sector (CHS) eingeben kann, oder moderner mit der sogenannten Blocknummer, die nach einer speziellen Formel aus der CHS Adresse errechenbar ist. Allerdings gibt es dabei ein im wahrsten Sinne des Wortes großes Problem. Bei heutigen Größenordnungen von Speichermedien im Terabyte – Bereich liegt die Anzahl dieser Blöcke in der Größenordnung von einer Milliarde. Das hieße jeder Benutzer müsste sich merken oder aufschreiben, was auf jedem der 1 Mrd. Blöcke gespeichert ist und gleichzeitig noch andersrum für jede Datei wissen, in welchen Blöcken diese liegt. Angenommen man könnte die Informationen für jeweils 1000 Blöcke auf einer Din-A4 Seite speichern bräuchte man immer noch eine Million Seiten Papier. Ein unmögliches Unterfangen, was aber zum Glück von

Dateisystemen ohne, dass wir davon im Normalfall etwas mitbekommen still und leise übernommen wird.

1.3. Wofür brauchen wir Dateisysteme?

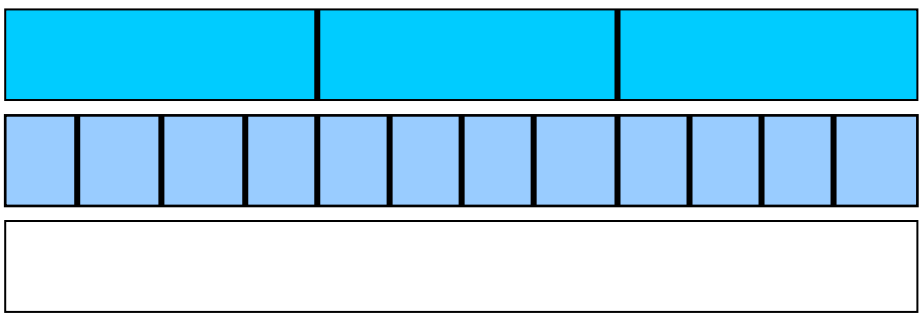
Dateisysteme leisten die komplette im vorherigen Absatz beschriebene Organisation der Daten auf einem Speichermedium und bilden die Schnittstelle zwischen dem Betriebssystem, welches auf Aktionen des Nutzers reagiert, und der Firmware des Speichermediums. Sie sorgen dafür, dass eine Datei eine für den Nutzer einfache und gut merkbare Adresse hat. Beispielsweise ist der Dateipfad C:\\Test\helloworld.c viel leichter zu merken als Cylinder 1017, Head 7, Sector 58 bzw. die entsprechende Blockadresse 1025634.

2. Grundlegendes Konzept

Dieser Abschnitt wird ausgehend von der Theorie eines Massenspeichers erläutern, wie Organisation der Daten in der Theorie funktioniert und abschließend Beispiele für die Aufteilung einer Festplatte für das Dateisystem beinhalten.

2.1. Massenspeicher

Ein Massenspeicher ist rein technisch betrachtet nur ein physikalisches Medium mit Speicherplatz. Folgende Grafik verdeutlicht die für die Nutzung mit einem Dateisystem wichtige Struktur eines Massenspeichers.



Der untere weiße Balken stellt den physikalischen Platz auf dem Medium dar, welcher typischerweise in Blöcke mit einer Größe von 512 oder 4096 Bytes aufgeteilt wird. Bei optischen Medien beträgt diese Blockgröße standardmäßig 2048 Bytes. Um die Zugriffsgeschwindigkeit zu erhöhen und den Aufwand des Dateisystems zu reduzieren werden allerdings nicht diese einzelnen Blöcke adressiert, sondern immer 4 oder 8 Blöcke zu einem Cluster zusammengefasst. Cluster sind die nach außen für das Dateisystem erkennbare Struktur, die es adressieren kann.

2.2. Datei

Eine Datei ist eine Menge von Clustern, die zusammengehörende Informationen enthalten. Die Cluster können, müssen aber nicht, an mehreren Orten auf dem Medium liegen. Damit zu jeder Datei klar ist, wo sich diese Cluster befinden, um sie auszulesen, gibt es eine Tabelle, die zu jeder Datei einige Beschreibungen enthält, die sogenannten Metadaten. Diese Tabelle enthält immer eine Angabe über die Länge der Datei, also wie viele Cluster von der Datei belegt sind. Außerdem ist, wie oben bereits erklärt, wichtig zu wissen, wo die einzelnen Cluster liegen, was ebenfalls in der Metadaten – Tabelle gespeichert wird. Dafür gibt es verschiedene Verfahren, die ich in Kürze erläutern werde. Die Beschreibungstabelle für die Dateien enthält je nach Art des Dateisystems noch weitere Informationen über die Datei. Das ist zum Beispiel der Dateityp. Zusätzlich können auch Zugriffsrechte für verschiedene Benutzer abgespeichert werden, sowie Datums- und Uhrzeitangaben, wie das Erstelldatum, das Datum der letzten Änderung der Datei, oder die Uhrzeit des letzten Zugriffs.

Für die Speicherung der Clusteradressen gibt es vier verschiedene Verfahren. Das einfachste Verfahren ist die Speicherung des Startclusters der Datei sowie die Länge der Datei, zum Beispiel Startcluster 5093 und Länge 12, sagt aus, dass die Datei im Cluster 5093 beginnt und Cluster 5104 das letzte Cluster ist. Nachteil dieses Verfahrens ist, wenn Cluster 5105 bereits belegt ist und nun die Datei vergrößert werden soll. Da nicht einfach die Längenangabe um die Anzahl der zusätzlichen Cluster erhöht werden kann, muss die komplette Datei an einen neuen Ort geschrieben werden, was die Abspeicherung der Änderungen deutlich verlangsamt. Zusätzlich kann es passieren, dass noch sehr viel Speicherplatz frei ist, dieser aber so verteilt ist, dass für eine große Datei kein ausreichend langer Bereich an aufeinanderfolgenden freien Clustern gefunden werden kann, sodass die Datei nicht abgespeichert werden kann.
Genau dieses Problem behebt das zweite Verfahren, bei welchem nur die Adresse des Startclusters in der Metadaten – Tabelle gespeichert wird und immer am Ende eines Clusters steht die Adresse des Folgeclusters. So kann freier Speicherplatz beliebig genutzt werden. Aber auch dieses Verfahren ist nicht ohne Nachteile. Durch die schlangenartige Abspeicherung müssen bei einem Zugriff auf das Ende der Datei immer erst alle vorhergehenden Cluster gelesen werden, um zu das gewünschte Cluster auslesen zu können. Sollte nun ein Cluster aufgrund eines Fehlers beschädigt sein, ist es möglich, dass die gesamte Datei nicht mehr gelesen werden kann.

Die Probleme beider Verfahren werden vom dritten möglichen Verfahren gelöst. Dieses speichert alle Clusteradressen der Datei einzeln in der Metadaten – Tabelle. So kann auf jedes Cluster schnell und unabhängig von allen anderen Clustern zugegriffen werden. Das Problem dieses Verfahrens ist die große Menge an Metadaten, die gespeichert werden müssen, da ja für jedes Cluster ein eigener Eintrag in den Metadaten vorhanden sein muss.

Aktuell angewandt wird deshalb das vierte Verfahren, die Speicherung in Extents. Ein Extent ist dabei eine Kombination aus Startcluster und Länge des Dateiabschnitts. In den Metadaten stehen dann insgesamt mehrere Extents, von denen aus insgesamt dann alle Cluster der Datei adressiert werden können. Der Metadaten – Eintrag für eine Datei kann dadurch zum Beispiel wie folgt aussehen: Startcluster 67439, Länge 39; Cluster 149228, Länge 8; Cluster 34929, Länge 3. So wird erreicht, dass freier Speicherplatz gut genutzt werden kann, auf jeden Bereich der Datei schnell zugegriffen werden kann und das bei nur wenigen Einträgen in der Metadaten – Tabelle.

2.3. Verzeichnisse

Zur sinnvollen Benutzbarkeit für Menschen fehlt dem Dateisystem jetzt noch der Ort, wo der Dateiname abgelegt wird. Dies geschieht in Verzeichnissen, welche man als spezielle Dateien ansehen kann. In einem Verzeichnis stehen immer Paare aus Dateiname und der Adresse der Metadaten in der Metadaten – Tabelle, die zu dieser Datei gehören. Anhand der Datei Helloworld.c, die in einem beliebigen Verzeichnis liegt, hier ein beispielhafter Weg, die Datei zu lesen:

In dem Verzeichnis steht der Eintrag „Helloworld.c | 0x00000f73“

Daher wird nun an Position 0x00000f73 in den Metadaten nach Informationen über die Datei gesucht. Dort findet sich dann wie oben erläutert die Dateilänge, z.B. 35 Cluster und anschließend die Extents mit den Clusteradressen, z.B. „3457,29 ; 3489,6“, wodurch nun mit dem eigentlichen Lesen der Datei begonnen werden kann. Zwischenzeitlich sind aber anhand der Metadaten auch schon die Berechtigungen des angemeldeten Benutzers überprüft worden, also ob dieser die Datei überhaupt lesen oder verändern darf. Wenn nicht, wird die Datei natürlich nicht weiter gelesen.

2.4. Aufteilung der Festplatte

Auf der Festplatte müssen also nun die Dateien, die Verzeichnisse und die Metadaten abgespeichert werden. Dazu kommen noch weitere Bereiche, wie der Bootblock, welcher direkt ausführbaren Maschinencode enthält, der beim Systemstart gelesen wird,

eine Block Availability Map (BAM), also eine Liste, die für jedes Cluster ein Bit enthält, dass aussagt, ob das Cluster frei oder benutzt ist und wenn die Festplatte mehrere Partitionen enthält außerdem noch eine Partitionstabelle. Zu möglichen Aufteilungen folgen nun drei Beispiele.

Das erste Beispiel ist eine Festplatte, die zwei Partitionen enthält. Auf ihr befindet sich der Bootblock (1), die Partitionstabelle (2) und die beiden Partitionen (grüne und orange Bereiche). Beide Partitionen haben jeweils eine Metadaten – Tabelle (3), eine BAM (4) und freien Platz für Dateien und Verzeichnisse (5)

Beispiel 1, Festplatte mit 2 Partitionen, Aufteilung nicht Maßstabsgetreu

Das zweite Beispiel stellt die einfachste Aufteilung einer Festplatte dar. Wie im ersten Beispiel liegt am Anfang der Festplatte der Bootblock (1), dann folgt die Metadaten – Tabelle (3), die BAM (4) und der restliche Platz wird für Dateien und Verzeichnisse (5) genutzt.

Beispiel 2, Einfache Aufteilung, Aufteilung nicht Maßstabsgetreu

Das dritte Beispiel ist in der Praxis oft anzutreffen. Die Aufteilung der Festplatte erfolgt wie in Beispiel 2, nur liegen die Bereiche anders. Für einen schnelleren Zugriff sind Metadaten – Tabelle (3) und BAM (4) in der Mitte des Speicherbereichs abgelegt. Davor und danach liegen Bereiche für Dateien und Verzeichnisse (5). Außerdem zeigt sich hier eins der Probleme moderner Dateisysteme. Es gibt einen zweiten Teil der Metadaten – Tabelle an einem anderen Ort auf der Festplatte (3.1). Dies geschieht, wenn der Platz der ursprünglichen Metadaten – Tabelle nicht mehr ausreicht, zum Beispiel weil der Benutzer zu viele Dateien abgespeichert hat, für deren Metadaten dann ein neuer Bereich angelegt werden muss. Dieses Phänomen nennt sich Metadaten – Fragmentierung und macht Dateizugriffe langsamer.

Aus dem gleichen Grund werden Verzeichnisse nicht an einem speziellen Ort abgelegt, sondern frei verteilt gespeichert, denn ein Nutzer kann beliebig viele Unterverzeichnisse mit beliebig vielen Dateieinträgen und weiteren Unterverzeichnissen anlegen (bis zu vom Dateisystem vorgegebenen Grenzen für z.B. Gesamtzahl aller Dateien).

3. Struktureller Aufbau

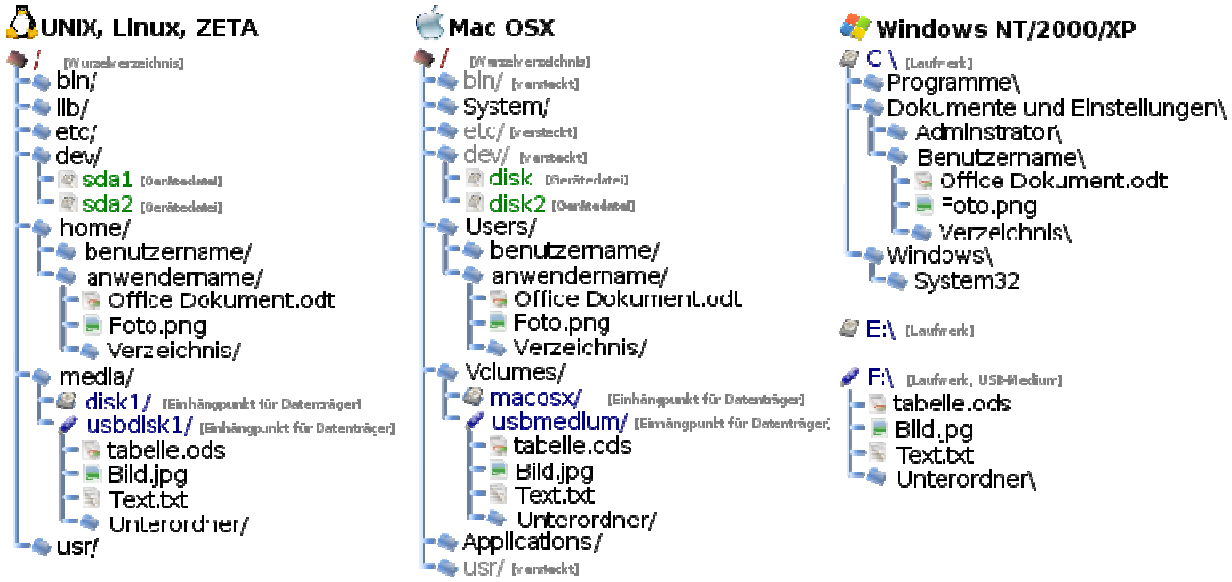
Dieses Kapitel erläutert den Unterschied zwischen linearen, hierarchischen und virtuellen Dateisystemen.

3.1. Lineare Dateisysteme

Lineare Dateisysteme sind heutzutage für den normalen Heimnutzer nicht mehr vorhanden. Sie wurden vor allem früher bei Lochbändern und Lochkarten eingesetzt und finden sich heutzutage nur noch auf Magnetbändern in der professionellen Datensicherung. Im Gegensatz zur oben erklärten Theorie, kennt ein lineares Dateisystem keine Verzeichnisse. Es liegen alle Dateien in einem einzigen Verzeichnis, welches direkt auf die Datei verweist. Mit der über die Jahre zunehmenden Anzahl an Dateien wurde es immer schwieriger, mit nur einem Verzeichnis zu arbeiten. Dies war der Grund für die Entwicklung von hierarchischen Dateisystemen.

3.2. Hierarchische Dateisysteme

Hierarchische Dateisysteme sind die heute üblichen jedem bekannten Dateisysteme. Die wichtige Eigenschaft hierarchischer Dateisysteme ist die Möglichkeit Unterordner zu erstellen und so eine für Nutzer sinnvolle Dateistruktur möglich zu machen. Folgende Grafik stellt beispielhafte Verzeichnisbäume von hierarchischen Dateisystemen unter verschiedenen Betriebssystemen dar.



Quelle :
http://de.wikipedia.org/w/index.php?title=Datei:Filesystem.svg&filetimestamp=20100727221414

3.3. Virtuelle Dateisysteme

Virtuelle Dateisysteme stehen eine Ebene über den normalen hierarchischen Dateisystemen. Ein virtuelles Dateisystem existiert nicht statisch auf der Festplatte

sondern nur virtuell und verwaltet mehrere andere Dateisysteme, ohne, dass der Nutzer davon etwas sieht. Für den Nutzer liegen alle Dateien in einem Dateisystem.

4. Beispiele für Dateisysteme

Nachdem nun die theoretischen Grundlagen erläutert sind, werden in diesem Abschnitt Name und Besonderheiten der gängigsten und wichtigsten Dateisysteme vorgestellt.

4.1. Apple

Seit 1986 verwenden alle Apple-Geräte das Hierarchical File System, kurz HFS sowie daraus entstandene Weiterentwicklungen wie das HFS+ und HFSX. HFS+ ist das Standarddateisystem unter Mac OS X.

4.2. Microsoft

Ausgehend von FAT12 waren die FAT – Dateisysteme lange Zeit die Standarddateisysteme für alle Windows – Betriebssysteme. FAT16 und FAT32 sind dabei jeweils Weiterentwicklungen, die die Verwaltung von größeren Dateien und einer größeren Anzahl von Dateien als ihre Vorgänger ermöglichten. Allerdings hat auch die modernste Variante FAT32 ihre Grenzen und kann zum Beispiel nur maximal 4GiB große Dateien verwalten. Somit wurde das New Technology File System, kurz NTFS entwickelt, welches seit Windows XP in Verwendung ist. Teilweise als Dateisystem für Windows 8 bezeichnet wird das neue ReFS (Resilient File System), welches allerdings nicht für Heimcomputer sondern nur für Windows – Serversysteme ausgelegt ist und dort durch zusätzliche Features wie Wartbarkeit im laufenden Betrieb und automatische Fehlerkorrektur bieten soll.

4.3. Linux

Für Linux gibt es eine Vielzahl von Dateisystemen, wobei die Extended File System – Familie die größte und die am stärksten verbreitete ist. Auf ext folgten die Weiterentwicklungen ext2, ext3, ext3cow und ext4, wobei ext2 über lange Zeit das Standardsystem unter Linux war und ext3 und ext4 heute weit verbreitet sind. Weitere wichtige Dateisysteme sind XFS, ein robustes und schnelles 64-Bit Dateisystem und JFS, ein Jornaling – Dateisystem. Für die Zukunft befindet sich btrfs in der Entwicklung, welches die Beschränkungen der gängigen ext3 und ext4 Dateisysteme aufheben soll und außerdem viele weitere Funktionen für schnelleren Zugriff und höhere Dateisicherheit enthalten soll.

4.4. Solaris und BSD

Die Unix – Betriebssysteme Solaris und BSD nutzen vor allem das Unix File System, kurz UFS. Für Solaris wurde außerdem das Zetta File System ZFS entwickelt, für BSD ist das Berkeley Fast File System (FFS) auch verbreitet. Gerade für die Unix – Betriebssysteme gibt es außerdem viele weitere Dateisysteme, die vor allem jeweils für eigene Betriebssysteme entwickelt wurden.

4.5. Optische Datenträger und andere Speichermedien

Auch für optische Datenträger, wie CDs, DVDs und Blue-rays gibt es standardisierte Dateiformate. Für CDs ist dies das Format nach ISO9660, was auch Compact Disk File System (CDFS) genannt wird. Da CDFS aber Beschränkungen zum Beispiel bei Dateinamen hat wurde von Microsoft das Dateisystem Joliet entwickelt, was längere Dateinamen mit Unicodezeichen ermöglicht. DVDs und Blue-rays werden meisten im Universal Disk Format (UDF) formatiert.

Aufgrund der besonderen Eigenschaften, insbesondere im Bezug auf die Abnutzung einzelner Speicherzellen, gibt es für Flash – Speichermedien besondere Dateisysteme, wie z.B. exFAT als Weiterentwicklung von FAT32 oder Extreme Flash File System (ExtremeFFS), für Solid State Drives von SanDisk.

5. Dateisystemzugriff und Operationen eines Dateisystems

Datenträgerseitig ist nun klar, wie das Dateisystem aufgebaut ist und wie es funktioniert. Dieser Abschnitt widmet sich nun der Benutzerseite und der Frage, wie denn jetzt das Dateisystem weiß, dass der Benutzer gerade eine Datei öffnen möchte.

5.1. Computerebenen und Kernel

Ein Computer lässt sich grundsätzlich in 3 Ebenen unterteilen. Die Anwenderebene in der z.B. Benutzerprogramme laufen, die Betriebssystemebene, welche unter anderem für das Dateisystem zuständig ist, und die Hardwareebene. In einem klassisch aufgebauten Betriebssystem mit monolithischem Kernel wandelt der Kernel die Nutzeranfragen in Hardwarebefehle um und befiehlt z.B. der Festplatte, das Verzeichnis auszulesen, welches der Benutzer soeben angeklickt hat.

5.2. Kernelbefehle

Um dem Kernel eine gewünschte Aktion zu signalisieren gibt es diverse Befehle, die von Anwendungsprogrammen aufgerufen werden können. Wichtig sind z.B. Erzeugen,

Löschen, Öffnen, Schließen und Lesen eines Verzeichnisses, sowie die Möglichkeit ein Verzeichnis zu wechseln. Unter Unix ist zum Beispiel der Befehl um ein Verzeichnis zu erzeugen „mkdir“, um ein Verzeichnis zu öffnen „opendir“ und für den Verzeichniswechsel „chdir“. Für Dateien gibt es ähnliche Befehle, zum Beispiel das Erzeugen (creat), das Löschen(rm), das Öffnen (open), das Schließen, das Lesen (read) und das Schreiben (write). Außerdem gibt es noch weitere Befehle, wie das Umbenennen einer Datei oder eines Verzeichnisses (mv), eine Möglichkeit, Dateien zu kopieren oder auch den Befehl, den Datenträger zu formatieren.
Was der Kernel und das Dateisystem alles tun müssen um einfach nur eine Datei zu öffnen lässt sich an einem einfachen Beispiel gut darstellen. Angenommen, ein Nutzer oder ein Programm möchte die Datei präsentation.pdf öffnen, die den Dateipfad „/path/topräsentation.pdf“ hat. Dafür muss zuerst das Ausgangsverzeichnis, unter Unix zum Beispiel das Wurzelverzeichnis geöffnet werden. Anschließend folgt das Suchen von „path“ im Wurzelverzeichnis, die Überprüfung, ob der Benutzer Zugriffsrechte für „path“ besitzt, und schließlich das Öffnen von „path“. Danach müssen dieselben Schritte noch einmal für „to“ ausgeführt werden. Nun ist man bereits im Verzeichnis, wo „präsentation.pdf“ liegt, es muss aber erneut erst der Eintrag für „präsentation.pdf“ im Verzeichnis gesucht werden, die Zugriffsrechte müssen überprüft werden und dann kann schließlich „präsentation.pdf“ geöffnet werden. Dies sind bereits zehn einzelne Schritte, die ausgeführt werden müssen um einfach nur die Datei zu öffnen.

5.3. Netzwerkdateisysteme

Sämtliche oben genannten Befehle sind nicht nur auf einen lokalen Rechner beschränkt. Auch im lokalen Netzwerk angeschlossene Speichermedien können mit den Kernelbefehlen angesteuert werden und verhalten sich dann wie lokale Speichermedien. Dafür müssen die Speichermedien allerdings mit einem Netzwerkdateisystem formatiert sein, welches die Kommunikation über das Netzwerk unterstützt. Beispiele für Netzwerkdateisysteme sind unter anderem das Distributed File System (DFS), das Network File System (NFS), das Quick File System (QFS) und das x File System (xFS).

6. Sicherheitsaspekte

Wie bei fast jeder Computerkomponente spielt auch bei Dateisystemen die Sicherheit eine wichtige Rolle. Die beiden Bereiche der Sicherheit bei Dateisystemen sind Zugriffsschutz und Datensicherheit und werden im Folgenden erläutert.

6.1. Zugriffsschutz

Eine wichtige Aufgabe des Dateisystems ist die Verwaltung der Zugriffsrechte für Dateien und Ordner. Dies regeln moderne Dateisysteme mit in der Metadaten – Tabelle gespeicherten Informationen über die Zugriffsrechte. Für verschiedene Benutzer und Benutzergruppen kann jeweils festgelegt werden, welche Rechte sie besitzen. Normalerweise gibt es 3 Stufen des Zugriffs, gar keinen Zugriff, einen schreibgeschützten Zugriff, und vollen Zugriff. Beim schreibgeschützten Zugriff kann ein Benutzer die Datei zwar öffnen, aber keine Änderungen an ihr speichern und die Datei auch nicht löschen. Insbesondere im Datenbankbereich können weitere Abstufungen vorgenommen werden, sodass Nutzer zum Beispiel neue Datensätze hinzufügen, von ihnen hinzugefügte löschen, aber nicht andere verändern können. Manche Dateisysteme bieten weitere Möglichkeiten, wie zum Beispiel in NTFS das Verstecken von Dateien, sodass diese für normale Benutzer ohne Administratorrechte nie sichtbar sind.

Der zweite Bereich des Zugriffsschutzes ist die Verschlüsselung der Daten durch das Dateisystem, sodass selbst bei einem vermutlich ungewollten Zugriff auf den Datenträger, die Daten nicht auslesbar sind. Verschlüsselung kann entweder durch dateisystemeigene Werkzeuge erfolgen, wie zum Beispiel Triple Des und AES beim Dateisystem NTFS, oder unabhängig vom Dateisystem mit „fremden“ Werkzeugen, wie zum Beispiel TrueCrypt.

6.2. Datensicherheit

Oberste Prämisse bei der Datensicherheit ist, dass ein Dateisystem weder Daten verlieren noch verfälschen darf. Dabei gibt es verschiedene Bereiche, die Probleme verursachen können, die vom Dateisystem zu lösen sind. Das erste Problem liegt beim Multitasking, wo es vorkommen kann, dass zwei Programme gleichzeitig verschiedene Dateien öffnen möchten. Aufgabe des Dateisystems ist es nun, beide Öffnungsvorgänge sauber zu trennen und für jedes Programm die richtige Datei zu öffnen. Etwas komplexer wird es, wenn zwei Programme gleichzeitig auf dieselbe Datei zugreifen möchten, oder ein zweites Programm auf eine Datei zugreifen möchte, die bereits von einem anderen Programm geöffnet ist. Würden beide Programme gleichzeitig vollen Zugriff erhalten, könnten Änderungen des einen Programms, durch überschreiben der Datei durch das andere Programm verloren gehen, wodurch das Dateisystem gegen die Prämisse verstoßen hätte, dass es keine Daten verlieren darf. Lösung für dieses Problem

sind sogenannte locks. Ein lock ist eine Sperre einer Datei, sodass diese, wenn sie bereits geöffnet ist, nicht erneut geöffnet werden kann. Bei Datenbanken, wo es große Dateien gibt, würde eine Sperre der ganzen Datei die Leistung erheblich beeinflussen. Dafür gibt es dort oft das Verfahren, dass nur immer der Bereich der Datei gesperrt ist, mit dem das Programm arbeitet und andere Programme auf andere Bereiche der Datei weiterhin zugreifen können.

Das zweite Problem eines Dateisystems ist ein plötzlicher Stromausfall, wodurch Schreibvorgänge eventuell nicht vollständig ausgeführt werden können und so ebenfalls Daten verloren gehen können. Hier gibt es verschiedene Ansätze, die dafür sorgen sollen, dass ein Stromausfall keinen Datenverlust verursacht. So ist es zum Beispiel möglich, die Hardware zu optimieren, indem man in ihr Kondensatoren verbaut, also Kurzzeit – Energiespeicher, die den Stromausfall aus Computersicht um kurze Zeit verzögern können, damit die Schreibvorgänge beendet werden können. Ein zweiter Ansatz ist die Optimierung der Software auf kleine Arbeitsschritte, was vor allem für Programme, die große Datenmengen schreiben wollen, wie Renderprogramme, sinnvoll ist. So sind 1000 kleine Schreibvorgänge besser als 10 große, da erstens kleine Schreibvorgänge bei einem Stromausfall eher noch beendet werden können und zweitens wenn ein Schreibvorgang fehlschlagen sollte, nur 0,1% der Daten anstatt 10% der Daten beschädigt sind. Dritter Ansatz ist das Journaling, welches in vielen modernen Dateisystemen implementiert ist. Allgemein heißt Journaling, dass alle Schreibvorgänge erst in einen gesonderten Bereich „notiert“ werden und danach richtig ausgeführt werden. Nach einem Systemabsturz kann nun durch überprüfen des Journals festgestellt werden, welche Daten eventuell beschädigt sind und überprüft und repariert werden sollen. Beim Journaling muss zwischen Metadaten – Journaling und Full – Journaling unterschieden werden. Während Metadaten – Journaling nur Änderungen an den Metadaten überwacht und dafür sorgt, dass diese korrekt sind, überwacht Full – Journaling alle Schreibvorgänge auf einem Datenträger. Vierter Ansatz ist die Copy-on-Write Funktionalität eines Dateisystems. Dies bedeutet, dass wenn eine Datei geändert werden soll, die gesamte Datei zusammen mit den Änderungen an einen neuen Speicherort geschrieben wird und der Metadaten – Eintrag entsprechend geändert wird. Sollte der Schreibvorgang fehlschlagen, liegt auf jeden Fall noch die vorherige Version der Datei vor, wodurch Copy-on-Write einen Komplettverlust einer Datei praktisch unmöglich macht. Allerdings verlangsamt natürlich das Kopieren der gesamten Datei bei jeder Änderung gerade bei größeren Dateien Schreibvorgänge deutlich. Ein weiterer

Ansatz ist noch, das Speichermedium als einen Endlosspeicher anzusehen, und vom Anfang bis zum Ende zu beschreiben und danach wieder am Anfang anzufangen, sodass immer möglichst viele alte Daten nicht sofort überschrieben werden und dadurch wiederhergestellt werden können.

7. Ausblick

Natürlich sind Dateisysteme jederzeit im Wandel und werden ständig weiterentwickelt, weshalb dieses Kapitel abschließend einen kurzen Ausblick auf die Zukunft geben soll. Das Zettabyte File System, kurz ZFS, darf dabei auf jeden Fall nicht fehlen. Es läuft auf Solaris und FreeBSD und kann ca. 281 Billionen Dateien auf einem bis zu ca. 18 Millionen Terabyte großen System verwalten und bietet viele Funktionen, die heutige Dateisysteme kaum haben, wie zum Beispiel ein integriertes RAID, Prüfsummen um Dateien auf Beschädigungen zu überprüfen, Snapshot-Backups, welche die Daten im laufenden Betrieb sichern, Data – Deduplication, wodurch redundante Daten physisch nur einmal gespeichert werden, und eine Poolfunktionalität, die es erlaubt, diverse Speichermedien in ein Dateisystem einzubinden. Die oben genannten Grenzen beruhen allerdings sogar nur auf aktuellen Implementationen, wo von den 128 Bit, auf denen das Dateisystem basiert, nur 64 Bit genutzt werden. Theoretisch kann ZFS mit 128 Bit alle je erzeugbaren Datenmengen in einem ZFS – System speichern. Jedoch ist ZFS eher für Server entwickelt und außerdem auf Heimcomputern aufgrund des schwierigen Umgangs mit 128bit Werten in 32 Bit bzw. 64 Bit Systemen eher langsamer. Ähnliches soll das neue btrfs für Linux leisten, welches z.B. ebenfalls Snapshots, Prüfsummen und RAID bietet, allerdings im Gegensatz zu ZFS, was schon seit 2006 nutzbar ist, immer noch in der Entwicklung steckt und bisher kaum mit Linux – Distributionen veröffentlich wurde. Von Microsoft Seite wird für Windows an ReFS gearbeitet, was als Nachfolger für NTFS dienen soll, aber nur für Dateiserver gedacht ist und nicht als Hauptdateisystem für einen Heimcomputer geeignet ist. Insgesamt sind also vor allem für Server Neuerungen zu erwarten, während Heimnutzer in nächster Zeit wohl bis auf btrfs unter Linux keine neuen Dateisysteme erwarten können.

8. Abbildungen

Sofern nicht anders angegeben sind Grafiken eigene Werke, die mit Powerpoint und/oder Word erstellt wurden.

9. Quellen

http://de.wikipedia.org/wiki/Dateisystem

http://de.wikipedia.org/wiki/Liste_von_Dateisystemen

http://de.wikipedia.org/wiki/Cluster-Dateisystem

http://kris.koehntopp.de/artikel/diplom/node21.html

http://de.wikipedia.org/wiki/Filesystem_in_Userspace

http://de.wikipedia.org/wiki/B%2B-Baum

http://www.heise.de/newsticker/meldung/TrueCrypt-zur-Verschluesselung-von-Dateisystemen-in-Version-4-0-erschienen-144764.html

http://de.wikipedia.org/wiki/Betriebssystemkern

http://de.wikipedia.org/wiki/Assembler_(Informatik)

http://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2010_2011/sds-1011-schoebel-btrfs-praesentation.pdf

http://www.hardwareluxx.de/index.php/artikel/hardware/storage/14705-zfs-und-die-zukunft-der-dateisysteme.html

http://www.vorlesungen.uni-osnabrueck.de/informatik/ShellProg/3_Dateisystem.rtfd/index.html

http://de.wikipedia.org/wiki/Master_File_Table

http://de.wikipedia.org/wiki/Cylinder_Head_Sector

http://de.wikipedia.org/wiki/Verzeichnisstruktur

http://www.itwissen.info/definition/lexikon/master-file-table-MFT.html

http://de.wikipedia.org/wiki/Cluster_(Festplatte)

http://de.wikipedia.org/wiki/Btrfs

http://de.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

http://www.pro-linux.de/artikel/2/1456/1,einfuehrung-und-features.html

http://de.wikipedia.org/wiki/ZFS_(Dateisystem)

http://de.wikipedia.org/wiki/XFS_(Dateisystem)

http://de.wikipedia.org/wiki/Journaling-Dateisystem

http://de.wikipedia.org/wiki/ReFS