Proseminar Speicher- und Dateisysteme: Lokale Dateisysteme

Christine Arndt
9arndt@informatik.uni-hamburg.de

17. März 2011

Inhaltsverzeichnis

Kapitel 1

Einführung

Begleitend zum Proseminar Speicher- und Dateisysteme der Abteilung Scientific Computing // Wissenschaftliches Rechnen ist dies die schriftliche Ausarbeitung zum Thema lokale Dateisysteme. Zunächst werden die Funktionen eines Dateisystems noch einmal aufgegriffen bevor dann verschiedene Dateisysteme im Detail erklärt werden: Beginnend mit den Microsoft Dateisystemen FAT (File Allocation Table) und dem NTFS (New Technology Filesystem), darauf das HFS (Hierarchical Filesystem), das im Moment von Apple genutzt wird und zuletzt das ext (Extended Filesystem), welches von Linux entwickelt wurde.

1.1 Funktionen von Dateisystemen

Verwaltung der Blöcke und Sektoren auf dem Datenträger
Datenträger sind ohne eine Formatierung für den Anwender unbrauchbar. Um Daten auf einem Datenträger ablegen zu können muß der Datenträger in kleinste Speichereinheiten unterteilt werden. Die Verwaltung der Daten erfolgt auf dieser Einteilung. Der File Allocation Table und das New Technology Filesystem benennen ihre kleinste Organisationseinheit Sektor, während das Hierarchical Filesystem und das extended Filesystem Blocks verwenden.

Schicht zwischen Betriebssystem und Anwendungen
Die gespeicherten Daten müssen für Anwendungen zum Lesen und Schreiben zur Verfügung gestellt werden. Man braucht also eine Verwaltung des Datenträgers, der die Daten zurückliefert.

Lokale Adressierung auf dem Datenträger
Das Dateisystem muß in der Lage sein, die auf dem Datenträger gespeicherten Daten zu finden. Die lokale Adressierung der belegten Blöcke/Sektoren erfolgt durch das Dateisystem.

Adresspfad durch Ordner und Dateipfad
Bei Installation neuer Programme bzw. beim Ablegen von Daten entstehen unentwegt neue Dateipfade die vom Hauptverzeichnis (/root) verschieden sind. Das Dateisystem organisiert nicht nur die Dateipfade, sondern verknüpft diese dann letztendlich auch mit den Daten, die in den Blöcken/Sektoren abgelegt sind.

Speicherung der Metadaten einer Datei
Darüber hinaus speichert das Dateisystem zu den Daten und ihren Adresspfaden die Metadaten von Dateien. Als Metadaten versteht man alle weiteren Eigenschaften einer Datei, die elektronisch erfaßt werden können. Hierzu zählen unter anderem: Datum der Dateierstellung, Datum und Uhrzeit der letzten Änderung, Dateigröße und Dateityp.

Kapitel 2

FAT - File Allocation Table

Der FAT - File Allocation Table wurde 1980 zuerst für die Nutzung von Microsoft Standalone Disc BASIC verwendet. Er hat seinen Namen daher, da eine Tabelle, also einen Table zur Allokation der Dateien nutzt. Die erste Version besaß 8 bit große table elements, die nächste ging dann direkt auf 12 und wurde FAT12 genannt. Datenträger werden in Cluster – eine Ansammlung von Speichersektoren – unterteilt. Cluster können leichter im Table gefunden werden als einzelne Sektoren. Allerdings bestimmt ihre Größe gleichzeitig die kleinste verfügbare Speichereinheit, da jeder Cluster nur eine Datei enthalten kann.
Neben den Metadaten können weitere Dateiattribute wie: System, Archiv, versteckt und schreibgeschützt gespeichert werden.
Der File Allocation Table besteht aus Bootsektor, reservierten Sektoren, dem FAT, dem Stammverzeichnis und dem tatsächlichen Datenbereich. Der Bootsektor befindet sich ganz zu Anfang des Dateisystems, dahinter folgen direkt die reservierten Sektoren. Danach kommt der File Allocation Table selbst, hierauf folgt dann das Stammverzeichnis, das in FAT12 und FAT16 noch Restriktionen unterliegt. Zum Schluß kommt dann der Datenbereich.

[0]

Abbildung 2.1: Aufbau des File Allocation Table

2.1 FAT12

Im FAT12 sind Cluster zwischen 512 und 4096 Byte groß. Bei 2^{12} Clustern sind somit Partitionsgrößen bis zu 16 MiB verwaltbar. Dateinamen werden im Schema 8.3, d.h. acht Buchstaben für den Dateinamen und drei Buchstaben für den Dateityp verwendet (bsp.: autoexec.bat). Das Hauptverzeichnis

[0] http://de.wikipedia.org/wiki/File_Allocation_Table#Aufbau

ist auf 14 Cluster beschränkt und es können somit maximal 224 Einträge hinterlegt werden. Ein Nachteil von FAT12 ist, dass keine usergebundenen Zugriffsberechtigungen für Dateien und Ordner gesteuert werden können.

2.2 FAT16

1983 kommt dann die nächste Version des File Allocation Table - FAT16 auf den Markt. 2^{16} Cluster können nun auf einer Festplatte verwaltet werden. Unter Windows NT macht dies Festplattengrößen bis 4 GB möglich. Dies setzt eine Clustergröße von 64 KB voraus. Allerdings gibt es weiterhin Einschränkungen im Hauptverzeichnis: es muß an fester Position bei fester Größe auf dem Datenträger vorliegen und kann nun 512 Einträge enthalten.

2.3 FAT32

FAT32 wurde dann 1997 mit Windows 95b eingeführt. Mit FAT32 können 2^{28} Cluster auf der Festplatte adressiert werden (4 reservierte Cluster). Bei einer Clustergröße von 32 KB konnten somit Festplatten von 8,8 TB verwaltet werden. Außerdem ist das Hauptverzeichnis nun flexibel und kann selbst nach der Formatierung noch wachsen.

2.4 exFAT

2006 wird exFAT eingeführt. Es soll das NTFS ersetzen, wo dieses nur schwierig bzw. nicht implementiert werden kann. Es wurde speziell für Flash Speicher entwickelt und wird von Windows Vista ab Service Pack 1 und von Windows XP ab Service Pack 2 und höher unterstützt.
Partitionsgrößen bis 64 ZiB bei einer Clustergröße von 32 MiB können verwaltet werden. Es gibt nun eine Tabelle, die freie Cluster indiziert. Nachteilig ist mangelhafte Unterstützung in anderen Systemen als Windows Vista, XP und 7. Da für SD-Speicherkarten und Memory-Sticks die Verwendung des exFAT vorgeschrieben ist, kommt es zu Kompatibilitätskonfilikten mit anderen Systemen.

VFAT

Ab Windows 3.11 wird VFAT optional und ab Windows 95 und höher fest unterstützt. VFAT erlaubt die Verwendung langer Dateinamen, inklusive Dateipfad bis zu 255 Zeichen.

Übersicht - FAT

	Start	Betriebssystem	Größe	Cluster
FAT12	1980	MS-DOS 2.0	16 MiB	2^{12}
FAT16	1983	Windows NT	4 GiB	2^{16}
FAT32	1997	Windows 95b	8,8 TB	2^{28}
exFAT	2006	Windows XP [1]	64 ZiB	2^{255}

Abbildung 2.2: Tabellenübersicht zur Entwicklung der File Allocation Table

FAT Dateisysteme sind relativ simple und somit verträgliche Dateisysteme, die zunächst auf Disketten verwendet wurden. FAT16 und FAT32 sind auch heute noch auf Flash Memory Cards, USB Sticks und Digital Cameras zu finden, obwohl viele Wechseldatenträger noch nicht von den Vorteilen von FAT32 profitieren.
Microsoft wechselt bereits Anfang der 90er Jahre von FAT auf NTFS. FAT bietet für Wechseldatenträger durch seine Einfachheit viele Vorteile, Funktionen wie Journaling und usergebundene Datenzugriffe werden jedoch mit dem Nachfolger NTFS erst verwirklicht.[2]

[1] benötigt Update
[2] Dieses Kapitel basiert auf: http://en.wikipedia.org/wiki/File_allocation_table

Kapitel 3

NTFS - New Technology Filesystem

Das New Technology Filesystem ist ebenfalls eine Entwicklung von Microsoft, die erstmals mit Windows NT 1993 auf den Markt kam. FAT gegenüber hat es einige Vorteile wie zum Beispiel Zugriffsschutz auf Dateiebene und größere Datensicherheit durch Journaling.
Beim Journaling werden alle Schreibbefehle zunächst in ein Journal geschrieben, so dass bei Systemabsturz ein konsistenter Zustand der Daten wiederhergestellt werden kann, auch wenn ein Schreibvorgang noch nicht beendet wurde.[3]
NTFS erbt einige Eigenschaften des HPFS, das zunächst von IBM und Microsoft gemeinsam entwickelt wurde. NTFS nutzt einen sogenannten Master File Table (MFT), der alle Informationen des Systems und die Einträge, welche Blöcke zu welcher Datei gehören, Zugriffsberechtigungen sowie Attribute, enthälte. Gespeichert werden folgende Attribute/Metadaten einer Datei:

- Dateigröße
- Datum der Dateierstellung
- Datum der letzten Änderung
- Freigabe
- Dateityp

und der Dateiinhalt selbst. Kleine Dateien und Verzeichnispfade können direkt in der MFT gespeichert werden, größere Dateien werden dann im Datenlauf gespeichert. Die MFT erhält bei der Formatierung einen festen Platz auf

[3] http://de.wikipedia.org/wiki/Journaling

der Festplatte von der sie standardmäßig 12,5% einnimmt. Ist dieser Speicheranteil erschöpft, beginnt die Speicherung auf der restlichen Platte. Ab jetzt werden die Dateien in Fragmenten zerstreut auf der Platte gespeichert. Dies nennt man Fragmentierung und führt zu erhöhtem Aufwand bei Lese- und Schreibvorgängen. Außerdem unterscheidet sich NTFS vom FAT Dateisystem durch effizientere und schnellere Speicherung, lange Dateinamen, die fast beliebige Unicode-Zeichen enthalten können, Dateipfadnamenslänge bis 32.767 Zeichen, die bereitserwähnte Zugriffsrechteverwaltung, maximale Dateigröße von (theoretisch) 16 EiB sowie Speicherung alternativer Datenströme. Wie viele andere Dateisysteme ist auch NTFS ein proprietäres Dateisystem. Seine schlechte Dokumentation komplizieren die Nutzung mit anderen Systemen, aus diesem Grund werden oftmals Drittanbieter Treiber benötigt.

Versionsübersicht - NTFS

NTFS 1.0	–	Windows NT 3.1
NTFS 1.1	–	Windows NT 3.5/3.51
NTFS 2.0	–	Windows NT 4.0
NTFS 3.0	–	Windows 2000 (NT 5.0)[4]
NTFS 3.1	–	Windows XP (NT 5.1)
NTFS 3.1	–	Server 2003 (NT 5.2)
NTFS 5.x	–	Windows Vista (NT 6.0), Server 2008[5]

NTFS wurde Juli 1993 auf den Markt gebracht. Seither arbeitet Microsoft an der Weiterentwicklung von NTFS, aktuell an der Version 6.x.
Außerdem gab es ein Projekt Namens Windows Future Storage (kurz: WinFS), dass 2003 erstmals vorgestellt wurde. Es sollte NTFS ersetzen und arbeitete basierend auf dem System relationaler Datenbanken. Allerdings gab es Probleme mit der Performanz, so dass im Juni 2006 das WinFS Entwicklungsteam schließlich erklärte, dass WinFS nicht als separates Produkt auf den Markt kommen würde. Einzelne Aspekte und Strukturen des bereits entwickelten Systems konnten jedoch in andere Projekte eingebettet werden.[6]

[4] mit 3.1-kompatiblem Datenträgerformat
[5] http://www.compu-seite.de/betriebssysteme/dateisysteme.htm
[6] http://en.wikipedia.org/wiki/WinFS

Kapitel 4

HFS - Hierachical Filesystem

1985 bringt Apple das Hierachical Filesystem (kurz: HFS) auf den Markt. Anders als das FAT-Dateisystem nutzt Apple damals schon eine B-Baum-Struktur, die hier kurz erläutert sei.

B-Baum

B-Bäume sind eine besondere Form des binären Suchbaums, in dem jeder Knoten mehr als 2 Kinder haben kann. Die Operationen; Suchen, Einfügen, Löschen und Split, können in $O(log*n)$ ausgeführt werden, dies wird dadurch begünstigt, dass B-Bäume auch nach jeder Operation wieder ausbalanciert vorliegen. Neben seiner hohen Operationseffizienz zeichnen sich B-Bäume durch einen hohen Verzweigungsgrad aus. Außerdem erfüllt ein B-Baum folgende Eigenschaften:

Eigenschaften[7]

1. Jeder Pfad von der Wurzel zu einem Blatt hat die Länge $h - 1$.
2. Jeder Knoten außer der Wurzel und den Blättern hat mindestens $k + 1$ Kinder. Die Wurzel ist ein Blatt oder hat mindestens 2 Kinder. Mit $k \in N$ sei der minimale Grad des Baumes.
3. Jeder Knoten hat höchstens $2k + 1$ Kinder.
4. Jedes Blatt mit der Ausnahme der Wurzel als Blatt hat mindestens k und höchstens $2k$ Einträge.

[7] Grundlagen von Datenbanken - Dr. Norbert Ritter - WiSe 10/11 - Kapitel 7

HFS

Das Hierarchical Filesystem ist ein proprietäres Dateisystem, jedoch gibt es detailierte, freizugängliche Dokumentationen, so dass es von den meisten modernen[8] Dateisystemen gelesen werden kann. Es sollte den Vorgänger MFS ersetzen und war für Disketten, Festplatten und Read-Only-Memories gedacht. Die Besonderheit des Dateisystems ist die Art wie Daten gespeichert werden. Jede gespeicherte Datei wird in zwei voneinander getrennten sogenannten Forks gespeichert, die unter dem gleichen Namen angesprochen werden. Die data fork, die die Datenquelle der Datei ist, und die Resource fork, die dann die zugehörigen Metadaten und beispielsweise Icons enthält. Dateinamen konnten 255 Zeichen lang sein. Eine Festplatte enthielt maximal 65.535 Dateien und war in der Größe auf 2 TB begrenzt.[9]

HFS+

HFS+ wurde dann 1998 auf den Markt gebracht. Es ist in der Lage Platten bis zu einer Größe von 8 EiB zu verwalten. Außerdem konnten Dateinamen auf HFS+ Unicodezeichen enthalten.

Apple nutzt seit 1985 das Hierarchical Filesystem und ist nicht davon abgewichen. Noch heute können Versionen von Mac OS HFS Partitionen lesen, allerdings können sie nicht mehr zum Booten verwendet werden. Seit Mac OS X 10.6 (Snow Leopard) werden auch andere Dateisysteme außer HFS+ unterstützt.[10]

[8] http://de.wikipedia.org/wiki/HFS_(Dateisystem)
[9] http://en.wikipedia.org/wiki/Hierarchical_File_System
[10] http://en.wikipedia.org/wiki/HFS_plus

Kapitel 5

ext - Extended Filesystem

Das erste Extended Filesystem wurde im April 1992 von Rémy Card implementiert. Es war das erste Dateisystem, das speziell für Linux entwickelt wurde und das vormalige Minix Filesystem ersetzen sollte. Zwei große Nachteile des Minix Dateisystems waren die Dateinamenslänge von maximal 14 Buchstaben und die begrenzte Partitionsgröße. Mit ext konnten diese beiden Begrenzungen überwunden werden. Nach dem ersten ext Filesystem gab es einen Wettbewerb um den Nachfolger, den schließlich ext2 auf Grund besserer Brauchbarkeit gewann.[11]

5.1 ext2 - Second extended Filesystem

1993 kommt ext2 heraus und ist bis heute weit verbreitet. Die ext Dateisysteme zeichnen sich vor allen Dingen durch die Verwaltung von Blocks, Inodes[12] und Verzeichnissen aus, die auf die Struktur von Unix Dateisystemen zurückgeht. Der Datenträger wird untergliedert in Blocks mit Größen von 1, 2 oder 4 KiB. Diese Blocks werden in Gruppen verwaltet. Das System ist ähnlich wie Cluster und Sektoren in FAT. Allerdings dienen die Gruppenblocks besser zur Vermeidung von Fragmentierung, zu der es dennoch auch in den Extended Filesystemen kommt. Außerdem gibt es einen Superblock, der sich ganz zu Anfang des Dateisystems befindet und alle wichtigen Informationen über die Konfiguration des Dateisystems enthält. Darüberhinaus enthält der Superblock eine Art Lageplan der Blockgruppen in dem hinterlegt wird welche Blockgruppe welche Inodes enthalten und welche noch frei sind. Aus Sicherheitsgründen liegen mehrere Kopien des Superblocks in den Blöcken verteilt vor, da das Dateisystem ohne Superblock unbrauchbar wäre.[13]

[11] http://en.wikipedia.org/wiki/Extended_file_system
[12] siehe Abschnitt 5.1. Inode
[13] http://de.wikipedia.org/wiki/Extended_file_system

Inode

I-Node ist englisch und steht für index node. Ein I-Node ist ein kleines Metadatenfragment einer Datei, das eine nummerierte Referenz auf den Dateinamen und den gesondert gespeicherten Inhalt einer Datei hält. Jedes Objekt in einem ext-Dateisystem wird durch einen I-Node repräsentiert. Im I-Node sind dann Pointer auf die Blöcke in denen die Daten der Datei gespeichert sind enthalten.
Darüber hinaus enthält der I-Node folgende Metadaten:

- Zugriffsrechte
- Eigentümer ID
- Dateityp (Verzeichnis, Link, ..)
- Größe der Datei
- Anzahl der Verweise auf die Datei
- Letzte Inode-Änderung, letzter Zugriff auf Datei und letzte Änderung der Datei
- Verweise auf Cluster auf den Datei-Inhalt

Dateiinhalt und Dateiname sind jedoch *nicht* enthalten. In jedem I-Node gibt es 15 Pointer, 12 davon zeigen direkt auf Datenblöcke, einer zeigt indirekt auf Datenblöcke. Der 14. zeigt auf Pointer, die indirekt auf Datenblöcke zeigen und wird 'doppelt' indirekter Pointer genannt. Der 15. Pointer ist ein dreifach indirekter Pointer, er zeigt auf Pointer, die auf Pointer zeigen, die indirekt auf Pointer zeigen.[14]

[14] http://en.wikipedia.org/wiki/Inode

Folgendes Schaubild soll diese Struktur veranschaulichen:

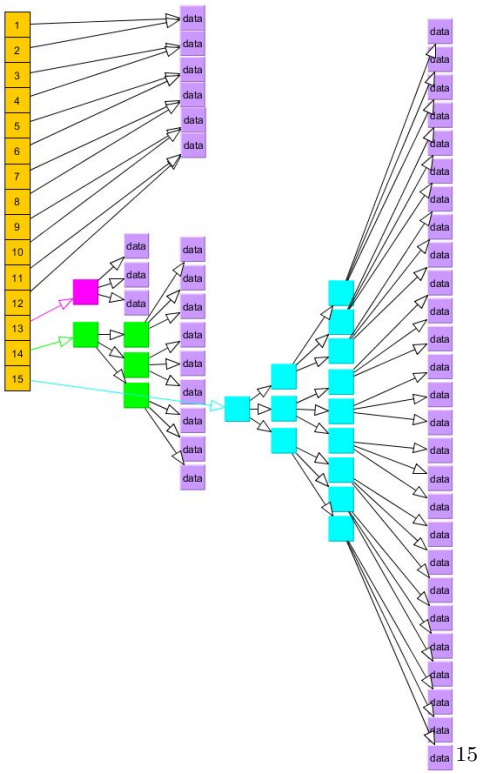

Abbildung 5.1: Schaubild zu I-Node-Pointer-Structure

5.2 ext3 - Third extended Filesystem

Mit ext3 kommt 2001 dann maßgeblich die Journaling-Funktion hinzu. Allerdings werden hier noch keine Checksummen zur Kontrolle verwendet, dies wird erst mit ext4 eingeführt. Außerdem wird nun in ext3 mit der Datenstruktur H-Tree gearbeitet. Ein großer Vorteil von ext3 ist die Tatsache, dass man ohne Back-up oder Datenwiederherstellung von ext2 auf ext3 updaten kann. Je nach Standpunkt ist die Schwierigkeit der Daten-Recovery als Vor- oder Nachteil des Dateisystems zu betrachten. Beim Löschen von Daten werden die Inodes zerstört, was zur Folge hat, das der Datenverlust nahezu unwiderbringlich ist.[16]

[15] http://de.wikipedia.org/wiki/Inode
[16] http://en.wikipedia.org/wiki/Ext3

H-Tree

H-Trees sind eine Datenstruktur ähnlich dem B-Baum. Der H-Tree ist ein *hashed* B-Baum, der eine Hashtabelle der Dateinamen verwendet. Weitere Vorteile sind:

- hoher Verzweigungsgrad
- konstante Tiefe von eins oder zwei
- er muß nicht ausgeglichen werden.[17]

5.3 ext4 - Fourth extended Filesystem

The fourth extended Filesystem wird erstmals 2006 vorgestellt. Allerdings ist die Version unstabil und muß überarbeitet werden. 2008 bringt Theodore Ts'o ext4 dann stabil heraus. ext4 nutzt Anstelle von Blöcken Extents um die Dateien zu verwalten. Ein Extent kann 128 MB mappen. Wenn ein Inode mehr als 4 Referenzen auf Extents halten muß, werden H-Trees zur Indizierung verwendet. Extens sind ein weiterer Fortschritt zur Vermeidung der Fragmentierung. Ein weiteres neues Feature von ext4 ist der sogenannte Multiblock Allocator. Anstatt Block für Block mit Daten zu beschreiben, sucht der Multiblock Allocator, zum Beispiel bei der Speicherung einer 100 MB großen Datei, den entsprechenden Platz 'am Stück', so dass die Daten hintereinander geschrieben werden können und Lese- und Schreibaufwand sehr viel geringer sind.[18]

Übersicht - Extended Filesystems

	Start	Anzahl Dateien	Größe	Struktur
ext2	1993	10^{18}	16 TiB	bitmap, table
ext3	1997	Variable	16 TB	table, H-tree
ext4	2008	4 Milliarden	1 EiB	linked list, H-tree

Abbildung 5.2: Tabellenübersicht zur Entwicklung der Extended Filesystems

Linux hat eine kontinuierliche Entwicklung von ext über ext2 und ext3 zu ext4 gemacht und ist dabei stets seinen Wurzeln treu geblieben. Der Hauptentwickler von ext4, Theodore Ts'o[19], wechselte im Januar 2010 von

[17] http://ext2.sourceforge.net/2005-ols/paper-html/node3.html
[18] http://en.wikipedia.org/wiki/Ext4
[19] http://en.wikipedia.org/wiki/Theodore_Ts%27o

der Linux Foundation zu Google und sorgte so dafür, dass Google ein Upgrade der Datenbankstruktur von ext2 auf ext4 durchführte. Außerdem wird beschlossen, dass die nächste Version von Android OS 2.3 (Gingerbread) auf einem ext4 Dateisystem laufen soll.

5.4 FHS - Filesystem Hierachy Standard

Der Filesystem Hierachy Standard ist eine Richtline für Softwareentwickler und Integratoren, die mit Linuxsystemen arbeiten. Er ist dazu gedacht, die Interoperabilität von Computerprogrammen zu fördern, so das User und Software in der Lage sind vorherzusehen in welchem Verzeichnis sich eine bestimmte Datei befindet. Hierzu unterteilt man die Dateien in 4 unterschiedliche Kategorien:

	shareable	**unshareable**
static	/usr	/etc
	/opt	/boot
variable	/var/mail	/var/run
	/var/spool/news	/var/lock

Abbildung 5.3: 4 Kategorien des Filesystem Hierachy Standard

Zunächst unterscheidet man static und variable, wobei static Dateien alle die jenigen sind, die sich ohne den Eingriff eines Systemadministrator nicht ändern. Weiterhin unterscheidet man shareable und unshareable Dateien. Als shareable gelten alle Dateien, die über ein Rechnernetz von anderen Rechnern genutzt werden können.
Die Entwicklung des Standards began 1993 und liegt seit 2004 in der Version 2.3 vor.[20]

Anmerkungen

Die Gewichtung der einzelnen Themen und Dateisysteme ist leider im Vortrag und somit auch in der Ausarbeitung nicht so ausgefallen, wie ich es mir vorgestellt hatte. Nachdem ich zu Anfang relativ viel Recherche für die ersten Themen unternommen habe, ist für das eigentliche Schwerpunktthema, Extended Filesystem, wenig Zeit und Kapazität geblieben. Ich finde dies selbst sehr bedauerlich, da ich das Thema sehr spannend finde und es gerne weiter ausgedehnt hätte. Außerdem hätte ich mich im Nachhinein wohl auch auf die Entwicklung eines einzigen Dateisystems beschränkt, da jedes einzelne Dateisystem sehr interessant und ergiebig ist.

[20] Filesystem Hierachy Standard 2.3 - Rusty Russel, 2004

Quellennachweise

- de.wikipedia.org
  - http://de.wikipedia.org/wiki/File_Allocation_Table#Aufbau
  - http://de.wikipedia.org/wiki/Journaling
  - http://de.wikipedia.org/wiki/HFS_(Dateisystem)
  - http://de.wikipedia.org/wiki/Extended_file_system
  - http://de.wikipedia.org/wiki/Inode
- en.wikipedia.org
  - http://en.wikipedia.org/wiki/File_allocation_table
  - http://en.wikipedia.org/wiki/WinFS
  - http://en.wikipedia.org/wiki/Hierarchical_File_System
  - http://en.wikipedia.org/wiki/HFS_plus
  - http://en.wikipedia.org/wiki/Extended_file_system
  - http://en.wikipedia.org/wiki/Ext3
  - http://en.wikipedia.org/wiki/Ext4
  - http://en.wikipedia.org/wiki/Inode
  - http://en.wikipedia.org/wiki/Theodore_Ts%27o
- http://www.compu-seite.de/betriebssysteme/dateisysteme.htm
- http://ext2.sourceforge.net/2005-ols/paper-html/node3.html
- http://ntfs.com/
- http://kernelnewbies.org/Ext4
- Filesystem Hierachy Standard 2.3 - Rusty Russel, 2004
- Grundlagen von Datenbanken - Dr. Norbert Ritter - WiSe 10/11 - Kapitel 7