Lokale Dateisysteme

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

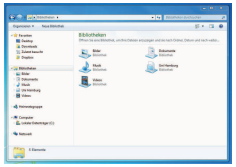
Universität Hamburg - Studentin der Wirtschaftsinformatik

11. März 2011

Inhalt der Präsentation

Was ist ein Dateisystem? Wozu dient es?

- Schicht zwischen Betriebssystem und Anwendungen
- Lokale Adressierung der Daten auf dem Datenträger
- Adresspfade durch Ordner und Dateinamen
- Verwaltung der Cluster, Sektoren und Blöcke im Speicher
- Speicherung der Metadaten einer Datei



Speichergrößen

Byte

Dezimalpräfixe		Binärpräfixe	
Name	Bedeutung	IEC-Name	Bedeutung
Kilobyte (kB)	10^3 Byte = 1.000 Byte	Kibibyte (KiB)	2^{10} Byte = 1.024 Byte
Megabyte (MB)	10^6 Byte = 1.000.000 Byte	Mebibyte (MiB)	2^{20} Byte = 1.048.576 Byte
Gigabyte (GB)	10^9 Byte = 1.000.000.000 Byte	Gibibyte (GiB)	2^{30} Byte = 1.073.741.824 Byte
Terabyte (TB)	10^{12} Byte = 1.000.000.000.000 Byte	Tebibyte (TiB)	2^{40} Byte = 1.099.511.627.776 Byte
Petabyte (PB)	10^{15} Byte = 1.000.000.000.000.000 Byte	Pebibyte (PiB)	2^{50} Byte = 1.125.899.906.842.624 Byte
Exabyte (EB)	10^{18} Byte = 1.000.000.000.000.000.000 Byte	Exbibyte (EiB)	2^{60} Byte = 1.152.921.504.606.846.976 Byte
Zettabyte (ZB)	10^{21} Byte = 1.000.000.000.000.000.000.000 Byte	Zebibyte (ZiB)	2^{70} Byte = 1.180.591.620.717.411.303.424 Byte

aus: http://de.wikipedia.org/wiki/Byte

Der Unterschied von Kilobyte zu Kibibyte ist $2,4\%$, von Zettabyte zu Zebibyte sind es bereits $18,1\%$.

File Allocation Table - Einführung

- 1980 für Microsoft Standalone Disc BASIC entwickelt
- Einteilung in Cluster von 512 bis 4096 Byte Größe
- Dateiattribute:
  - Schreibgeschützt
  - Versteckt
  - System
  - Archiv

FAT Dateisysteme sind aufgeteilt in:

Bootsektor	reservierte Sektoren	FAT	Stammverzeichnis	Datenbereich

File Allocation Table - Entwicklung

- Dateinamen im Schema 8.3 (bsp.: autoexec.bat)
- Keine usergebundenen Berechtigungen für Dateien und Odner
- Begrenzung der Einträge im Hauptverzeichnis
- Hauptverzeichnis mit fester Position und Größe in FAT16
- Mit FAT32 wird das Hauptverzeichnis flexibel - kann wachsen
- exFAT wurde speziell für Flash-Speicher entwickelt
- Kompatibilitätskonflikte mit anderen Systemen

noch immer Verwendung für Digitalkameras, Flash-Speicher, Memory Cards und 3,5" Disketten

File Allocation Table - Übersicht

	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}

VFAT

Ab Windows 3.11 wird VFAT optional und ab Windows 95 und höher voll unterstützt. Mit VFAT können lange Dateinamen verwendet werden, inklusive Dateipfad mit bis zu 255 Zeichen.

1. benötigt Update

New Technology Filesystem - NTFS

- **Vorteile gegenüber FAT**
  - Höhere Datensicherheit durch Journaling
  - Master File Table arbeitet schneller und effizienter
  - Lange Dateinamen mit fast allen Unicodezeichen
  - Dateinamenspfadlänge 32.767 Zeichen
  - Zugriffsrechteverwaltung
  - Max. Dateigröße 16 EiByte
- **Nachteile - allgemein**
  - Proprietäres Dateisystem - nicht frei
  - Fragmentierung der Festplatte durch verteilte Dateifragmente
  - Kompatibilität nur mit zusätzlichen Treibern gewährleistet

NTFS 1.0 bis 5.x

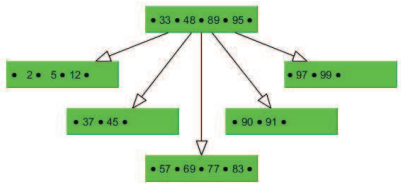
- NTFS 1.0 – Microsoft Windows NT 3.1
- NTFS 1.1 – NT 3.5/3.51
- NTFS 2.0 – NT 4.0
- NTFS 3.0 – 2000 (NT 5.0) mit 3.1-kompatiblem Datenträgerformat
- NTFS 3.1 – XP (NT 5.1)
- NTFS 3.1 – Server 2003 (NT 5.2)
- NTFS 5.x – Vista (NT 6.0), Server 2008

http://www.compu-seite.de/betriebssysteme/dateisysteme.htm

Einschub - B-Bäume

Operationen

Suchen	$O(k * h)$
Einfügen	$O(k * h)$
Löschen	$O(k * h)$
Split	$\Theta(k)$



Eigenschaften

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.

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

Hierachical Filesystem - HFS und HFS+

- **HFS**
  - Kam 1985 von Apple auf den Markt
  - Für Festplatten, Disketten und ROMs
  - Dateispeicherung in zwei Teilen (resource und data fork)
  - Sortierung in B-Trees
  - Dateinamen mit 255 Zeichen
  - Partitionsgröße bis 2 TByte
- **HFS+**
  - Kam 1998 auf den Markt
  - Partitionsgröße bis 8 EiByte
  - Dateinamen in Unicode

Extended Filesystem - Geschichte und Einführung

- Rémy Card implementiert April 1992 ext
- Entwicklung speziell für Linux
- Ablösung von Minix
  - Dateinamen mit max. 14 Zeichen
  - max. 64 MB Partitionen

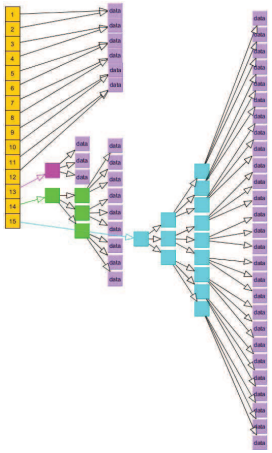
- 1993 kommt ext2 basierend auf Unix-Dateisystem-Struktur
  - Verwaltung über Blöcke, Inodes, Verzeichnisse
- Journaling und H-tree Sortierung kommen 2001 mit ext3
- Nachfolger ext4 kommt zunächst instabil in 2006
- 2008 bringt Theodore Ts'o stabiles ext4 mit vielen Erweiterungen

Extended Filesystem - Inode

Inode (engl. index node)

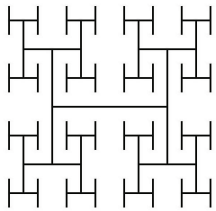
Metadaten

- 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
- Datei-Inhalt und Dateinamen sind *nicht* enthalten



aus: http://en.wikipedia.org/wiki/Inode

Einschub - H-Bäume



Eigenschaften

- Konstante Tiefe von eins oder zwei
- Hoher Verzweigungsgrad
- Nutzen einen Hashtable der Dateinamen
- Müssen nicht ausgeglichen werden

aus: http://en.wikipedia.org/wiki/Htree

Extended Filesystem - Überblick

	Start	Anzahl Dateien	Größe	Struktur
ext2	1983	10^{18}	16 TiB	bitmap, table
ext3	1997	Variable	16 TB	table, H-tree
ext4	2006	4 Milliarden	1 EiB	linked list, H-tree

Kompatibilität

Ein großer Vorteil der ext Dateisysteme ist ihre Kompatibilität. Während viele Dateisysteme zwar andere Dateisysteme lesen können, jedoch nicht schreiben, ist ext abwärtskompatibel. D.h. ext3 und ext4 sind in der Lage auf einem ext2 formatierten Datenträger zu schreiben.

Filesystem Hierachy Standard - FHS

Jede Datei gehört in eine der 4 Kategorien:

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

2 Gründe für den FHS:

- die Software

  und

- der User

können vorhersagen wo sich eine Datei befindet

Zusammenfassung und Ausblick

Microsoft nutzt *NTFS* mit Windows NT seit Juli 1993
- kontinuierliche Weiterentwicklung zu NTFS v.3.1
- parallele Entwicklung und Absage von WinFS (Windows Future Storage)

Apple nutzt *HFS(+)* seit 1985 in Mac OS
- ab Mac OS X 10.6 (Snow Leopard) werden weitere Dateisysteme unterstützt

Linux nutzt *ext* seit 1992
- Entwicklung von ext zu ext4 durch Erweiterung

Google
- Im Dezember 2010 hat Google angekündigt ihre Speicherinfrastruktur von ext2 auf ext4 upzugraden.
- Außerdem wird Android 2.3 (Gingerbread) ebenfalls auf ext4 laufen. (Theodore Ts'o)

Quellennachweis

genutzte Quellen

- de.wikipedia.org
- en.wikipedia.org
- compu-seite.de/betriebssysteme
- ext2.sourceforge.net/2005-ols/paper-html/node3.html
- ntfs.com
- Filesystem Hierachy Standard 2.3 - Rusty Russel, 2004

.. zum Schluß

Vielen Dank für die Aufmerksamkeit!

Gibt es noch Fragen??