Dateisysteme unter Windows und Unix im Vergleich

Ein Einführung

Priv.-Doz. Dr. Stefan Bosse
Universität Bremen, FB Mathematik & Informatik
19.3.2018
sbosse@uni-bremen.de

1. Inhalt

2. Einführung

2.1. Ziele

- Sie verstehen die **Grundbegriffe und Funktionen** eines Dateisystems, insbesondere Datei, Datensatz, Dateimetadaten, Dateitypen, Verzeichnishierarchie und Organisation, Dateiverknüpfungen, Benennungsregeln.
- Sie erlangen einen **Praxisbezug** mit konkreten Dateisystemen unter Windows (FAT/NTFS) und UNIX (UFS/EXT) von deren **Implementierung** und Interna
- Sie erkennen **Grenzen** von verschiedenen Dateisystemen (Dateigröße, Robustheit, usw.)
- Verständnis und praktische Übungen zum:
  - Blockspeichermodell der Datenträgerabstraktion
  - Linearspeichermodell der Dateiabstraktion
- Sie können die Funktion und Nutzen protokollierender Dateisysteme beschreiben
- Sie analysieren und interpretieren **Metadaten** der Dateisysteme FAT und MINIX mithilfe eines hexadezimalen Diskeditors (Übung)

2.2. Historischer Überblick

2.3. Betriebssysteme

- Dateisysteme sind eng verknüpft mit Betriebssystemen:
  - MS Windows: `FAT` (File Allocation Table), `NTFS` (NT Filesystem)
  - Linux, Android ($\in$ UNIX): `EXT2`,`EXT3` (Extended Filesystem)
  - Solaris ($\in$ UNIX): `ZFS` (Zettabyte Filesystem)
  - Aber unabhängig (CD/DVD): `Joliet`, `RockRidge`, ..
- Ein Dateisystem stellt eine der wichtigen **Abstraktionsmechanismen** dar
- Eigenschaften der Dateisysteme spiegeln Betriebssystemklasse wieder:
  - Einzelnutzer System
  - Mehrbenutzer System

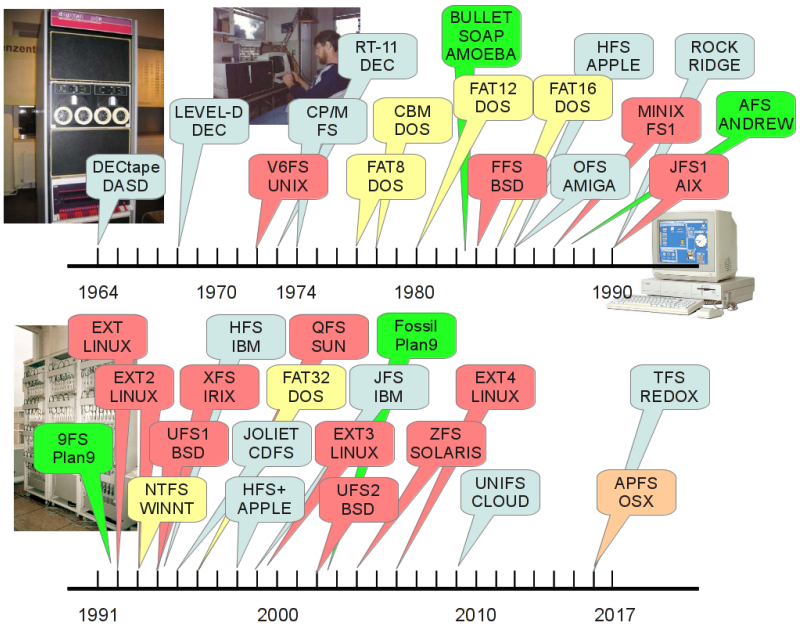
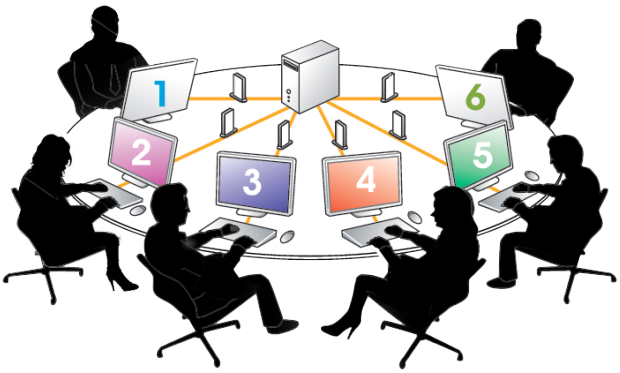


Abb. 1. Zeitliche Entwicklung einer Auswahl (!) von Dateisystemen. (Rot) UNIX (Gelb) DOS/Windows (Grün) Verteilte Betriebssysteme und Dateisysteme

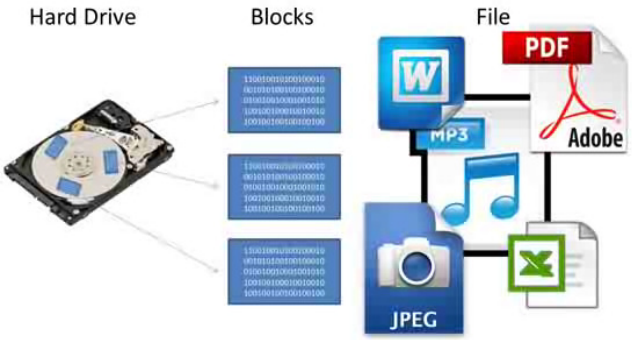
- Server System
- Mobiles System
- Netzwerk System
- Verteiltes System



2.4. Dateisysteme

Aufgabe: Organisation und Ablage von beliebigen digitalen Objekten

- Speicherung der Objektdaten auf **Langzeitspeichern** (z.B. Festplatte)
- Die Objekte werden auf verschiedenen Abstraktionsebenen auf binäre **Daten** abgebildet
- Objekte: *Text, Musik, Video, Foto, Zeichnung, Programmcode, ..*
- Jedes Objekt ist durch einen Satz von Attributen gekennzeichnet: *Name, Größe, Art, Einordnung in Containern, ..*



[A]

- Attribute sind **Metadaten**
- Organisation der Objekte in **Containern**
- Ein Objekt ist schließlich nur noch eine Nummer!

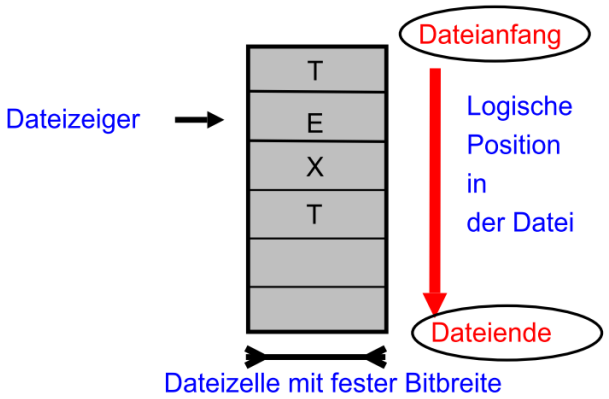
3. Funktion von Dateisystemen

Fragestellung:
Was ist allen Dateisystemen gemeinsam?

3.1. Dateien - Das Linearmodell

- Auf der Programmier- und Programmebene sind Dateien eine Folge von Bytes → Array/Tabellenstrukturierung

- Eine Datei kann gelesen (Lesezugriff) und verändert werden (Schreibzugriff)
- Es gibt einen Lese- und Schreibzeiger der eine Zelle des Arrays zum Lesen oder Schreiben auswählt




```
C:
fd=fopen("/tmp/x.dat",
   "r+");
n=fread(buf,1,10,fd);
n=fwrite(buf,1,10,fd);
fseek(fd,10,SEEK_SET);
fclose(fd);
```

Abb. 2. Logischer Aufbaue einer Datei (links) und programmatischer Zugriff (rechts, C)

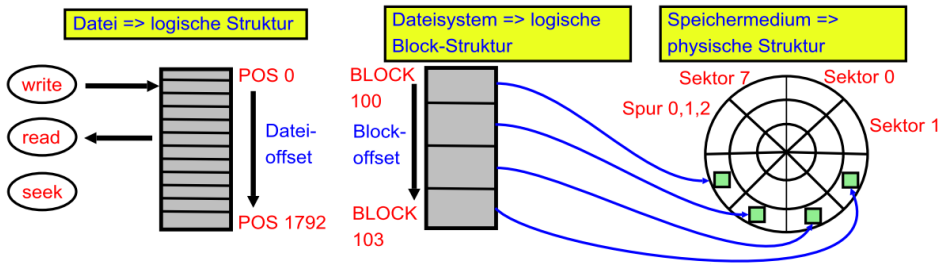
3.2. Dateien - Metadaten

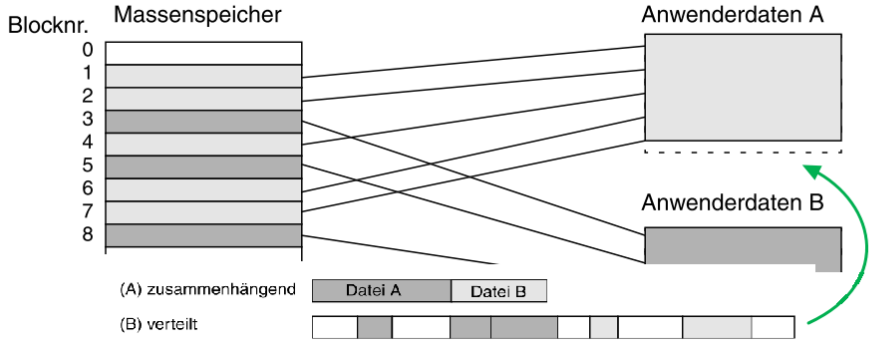
- Mit einer Datei sind eine Vielzahl von Attribute verknüpft → Metadaten
  - Name
  - Größe (Länge des Dateninhalts)
  - Datum (Erzeugung, Zugriff)
  - Typ (Text, Binär, Programmcode, usw.)
  - Zugriffsrechte (Operationen Lesen, Schreiben, Ausführen, usw.)
  - Eigentümer
  - Sperren (Modifikationsschutz bei Mehrfachzugriff)
- Welche Metadaten gespeichert werden hängt vom Dateisystem und Betriebssystem ab!
- Metadaten werden i.A. in Verzeichnisstrukturen untergebracht. Die Datei selber ist nur durch eine eindeutige Nummer referenziert.

3.3. Physische und Logische Ebene

Blockstrukturierung

- Für die technische Speicherung (Festplatte, DVD, USB-FLASH) werden Dateisysteme und die Dateien in Blöcke gleicher Größe zerlegt
- Das ermöglich eine vereinfachte und effizientere Ausnutzung des Datenträgers (siehe verteilte Belegung)





3.4. Organistationsstrukturen

Verzeichnisse und Baumstruktur

- Die meisten Dateisyssteme organisieren Dateien in Baum- oder Graphenstrukturen
- Diese Graphen bestehen aus Knoten: **Verzeichnisse** (Ordner) die als Container für Dateien und weitere Unterverzeichnisse dienen

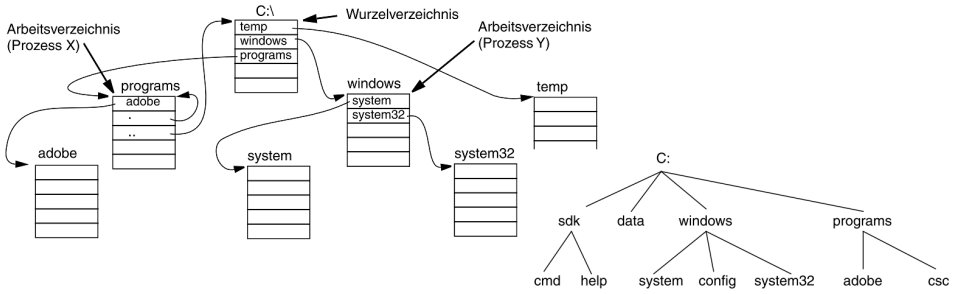


Abb. 3. Beispiel von Verzeichnissen und eines Verzeichnisbaums

Pfade

- Einzelne Dateien und Unterverzeichnisse sind dabei eindeutig über einem **Pfad** gekennzeichnet.
- Ein Pfad beginnt im Wurzelknoten und geht entlang des hierarchischen Verzeichnisweges; **Wurzelknoten**: *Windows* → `C:\`, *Unix* → `/`
- Einzelne Elemente des Pfades werden durch ein **Trennzeichen** zusammengefügt: *Windows* →`\`, *Unix* → `/`

Beispiele für Pfade

```
Windows
Pfadname: \Programme\java\j2re1.4.2_06\bin\java.exe
Dateiname: java.exe
Verzeichnispfad: \Programme\java\j2re1.4.2_06\bin\
Verzeichnisname: bin
```

```
Unix
Pfadname: /usr/jre1.6.0_31/bin/java
Dateiname: java
Verzeichnispfad: /usr/jre1.6.0_31/bin
Verzeichnisname: bin
```

4. Aufbau und Eigenschaften von Dateisystemen

Fragestellung:
Wie werden die Daten nun konkret organisiert????

4.1. UFS (Unix Filesystem)

Struktur

- Dateien werden in Blöcken im **Blockbereich** gespeichert
- Die Belegung kann verteilt sein
- Das Auffinden von Dateiblöcken geschieht durch Index Blöcke (I-nodes) im **Indexbereich**
- Der Superblock enthält alle relevanten Parameter des Dateisystems (Bereichsgrößen, Datum, Name, Blockgröße, ..)

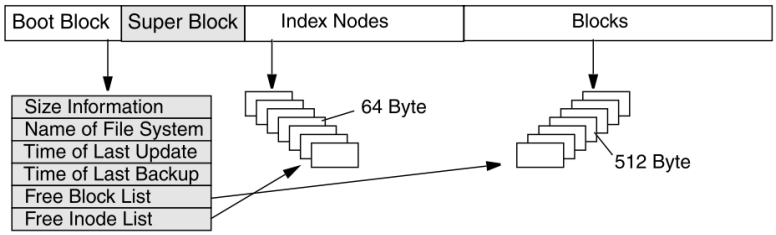


Abb. 4. Allgemeine Datenträgerunterteilung des UFS.

I-Nodes

- Eine I-Node ist eine Tabelle mit Blockreferenzen die zu einer Datei gehören.
- Eine I-Node hat eine feste Größe → Begrenzte Anzahl von Datenblockreferenzen!
- Daher Verkettung von vielen I-Nodes für große Dateien

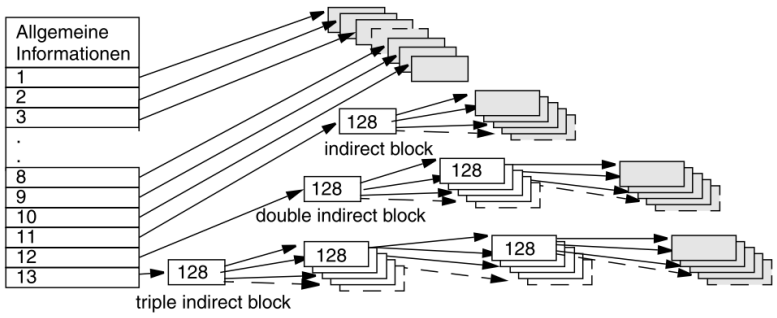


Abb. 5. Vernetzung und Tabellenstruktur von I-Nodes

4.2. FAT (File Allocation Table)

- Es gibt nur eine einzige globale Tabelle für alle Dateien mit Blockreferenzen die zu einer Datei gehören.

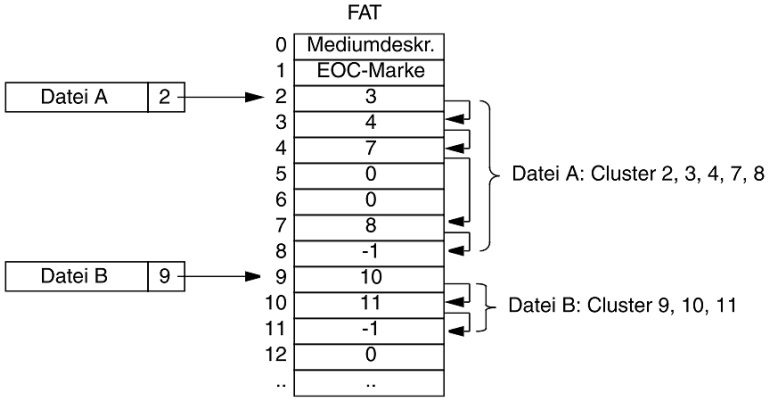
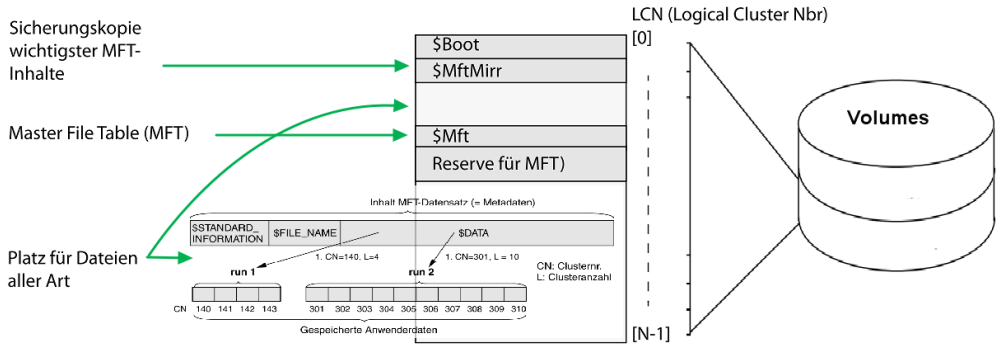


Abb. 6. Funktionsweise der FAT-Belegungstabelle

4.3. NTFS (New Technology Filesystem)

- Keine Trennung zwischen Index- und Datenbereichen
- Das Auffinden von Dateiblöcken geschieht mit Bereichsbäumen (B-trees) → komplex in der Implementierung, aber schnelle Suche!
- Alles ist eine Datei, auch Verzeichnisse → sehr viele Metadateien

- Es gibt eine erweiterbare globale Tabelle für alle Metadateien und Dateien: Masterfile Table MFT → Dateisystemjournal
- Eigenschaften: *Wiederherstellbarkeit*, *Zugriffsrechteverwaltung*, *Datenkomprimierung*, *Datenverschlüsselung*



4.4. Metadaten und Eigenschaften

Aufagbe!
(1) Windows: File Explorer öffnen → Nach C:\Windows\System32 wechseln → Detailinformation von Datei cmd.exe
(2) Unix: Terminal öffnen → cd /usr/bin → ls -l sh

Attribut	**DOS/WIN98 FAT**	**Windows NTFS**	**Unix EXT2/UFS**
Dateiname	+	+[1]	+
Namenslänge	11/255[3]	255	255
Case Sens.\UTF	-\-	+\+	+\?
Eigentümer	-	+	+
Erzeugungszeit	+	+	-/+
Zugriffszeit	-	+	+
Modifikationszeit	-	+	+
POSIX Rechte[2]	-	-	+
Links	-	+	+
Max. Dateien	65k-270M[3]/65k[4]	4G/4G[4]	4G/32k[4]
Dateigröße	<4GB[3]	>1TB	>16GB/512GB

[1] Primäres Attribut (direkt gespeichert) [2] Read/Write/Execute: Owner/Group/Others
[3] Abhängig von FAT Version und ggf. Volumegröße [4] Pro Verzeichnis

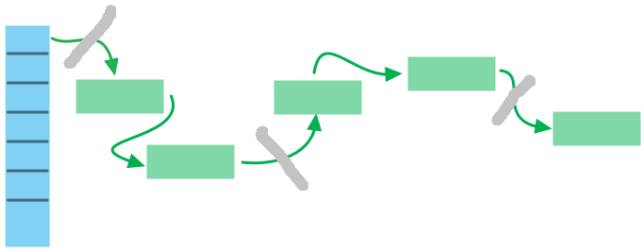
4.5. Fehler und Robustheit

Dateien werden bei den meisten Dateisystemen mit verketteten Listen gespeichert.

Speichern einer Datei bedeutet: 1. Datenblöcke speichern und verketten; 2. Verzeichnis aktualisieren; 3. I-Node Tabelle und Superblock aktualisieren.

Fehler

- Ist die Kette an einer Stelle beschädigt ist die ganze Datei fehlerhaft und inkorrekt.
- Wird das Verzeichnis nicht aktualisiert sind die Daten nicht auffindbar
- Ist das Verzeichnis ebenfalls eine Kette kann bei einer Beschädigung das gesamte Verzeichnis unbrauchbar werden
- Ist der Superblock beschädigt ist das gesamte Dateisystem unbrauchbar



4.6. Journaling, Replikation und Redundanz

- Um nach einem Crash das Dateisystem wieder schnell und konsistent herstellen zu können wird bei vielen modernen Dateisystemen ein **Journal** geführt.
- **Replikation** und Synchronisation von Superblöcken (der Zugang zum Dateisystem)
- **Replikation** von Dateisystemen (Synchronisierte Kopien auf mehreren physischen Datenträgern)
- **Backups**!!!

- Aufbau des Dateisystems selber kann verbessert werden um Inkonsistenzen zu vermeiden!
  KISS: Keep It Simple and Safe!
- Auswahl des richtigen Dateisystems!

5. Zusammenfassung

- Dateisysteme sind wesentlicher Bestandteil von Betriebssystemen: Service
- Dateisysteme speichern Dateien und bieten eine Organisationsstruktur mit Verzeichnissen und Bäumen
- Es gibt unterschiedliche Repräsentation von Dateien: Linearmodell; Blockmodell; Physisches Modell (Datenträger)
- Es gibt eine große Vielzahl von Dateisystemen die sich unterscheiden durch:
  - Speicherung der Dateien (Größe, Verteilung, Effizienz, Performanz)
  - Verwaltung von Verzeichnissen
  - Metadaten (Dateiattribute)
- Die Dateisysteme von UNIX und Windows unterscheiden sich deutlich auf der Implementierungsebene; Der Service ist aber ähnlich!
- Verkettung und Indextabellen sind wesentliche Datenstrukturierung

6. Referenzen und weierführende Literatur

Bücher - Vertiefung

1. Eduard Glatz, Betriebssysteme - Grundlagen, Konzepte, Systemprogrammierung. dpunkt, 2015. → Kapitel 9 Dateisysteme

2. Andrew Tanenbaum, Operating Systems (MINIX), Pearson Education, 1997. → Kapitel 5

Videos

A. Object Storage - the key to Cloud and Big Data, www.youtube.com/watch?v=Vl28V5tL3Wg

7. Praktische Übung

Verwendung von JavaScript und node.js

1. **Dateisystem**: Sie bekommen ein JavaScript Template das folgende Funktionen zur Verfügung stellt: Schleife, Datei erzeugen (Länge n Bytes), Verzeichnis erzeugen, Dateinamen der Länge m erzeugen.

   Untersuchen Sie unter Windows und Linux oder MacOs folgende Fragestellung:

   a. Erzeugen Sie eine leere Datei mit einem Namen, Finden Sie heraus wie lang der längste Name ist der gespeichert werden kann. Wird das theoretische Maximum vom jeweiligen Dateisystem erreicht? Wenn nein, was könnte der Grund sein?

   b. Erzeugen Sie ein Verzeichnis `test`. Wechseln Sie unbedingt in dieses Verzeichnis. Erzeugen Sie dort mit einer Schleife Unterverzeichnisse. Wie viele können erzeugt werden?

   c. Erzeugen Sie eine Date der Länge x. Wie groß kann x werden?

2. **Linearmodell**: Sie bekommen ein JavaScript Template das folgende Funktionen zur Verfügung stellt: Schleife, Datei erzeugen (leer), Lesen aus einer Datei (zeichenweise), Schreiben in eine Datei (zeichenweise), Dateizeiger positionieren, Datei schließen.

   a. Öffnen Sie eine Textdatei A zum Lesen. Erzeugen Sie eine zweite leere Datei B. Kopieren Sie mit den verfügbaren Operationen die Inhalte von A nach B.

3. **Logische Struktur**: Sie bekommen ein kleines MINIX Image (1.4MB). Lesen Sie im Buch [2] den Aufbau des MINIX Dateisystems und analysieren Sie das Dateisystem.