

6 Dateisysteme (file systems)

Anforderungen:

- Große Menge von Informationen
- Dauer die Laufzeit eine P. hinaus (persistent)
- Gleichzeitiger Zugriff

Übliche Form: Dateien, welche über ein Dateisystem auf einem Datenträger angelegt werden.

Datenträger = Festplatte, Diskette, CD-ROM, Speicherstick, ...

Aber auch „roher“ Zugriff auf Datenträger: Swap-space, Datenbanken, Bandlaufwerke

Festplatte: Kleinste Einheit der „Block“ (auf Zylinder, Platte, Sektor, ...), typischerweise 512 Bytes groß

Einheitliche Präsentation von Dateisystem/Dateien für den Benutzer und für die P.

Übliche Dateisysteme

- FAT16, FAT32, NTFS (Windows)
- HFS+ (MacOSX)
- EXT2, JFS, XFS, ReiserFS (Linux)
- UFS, FFS (BSD, Unix)
- ISO9660 (CD-ROM), UFS (CD-RW, DVD)

Dateien

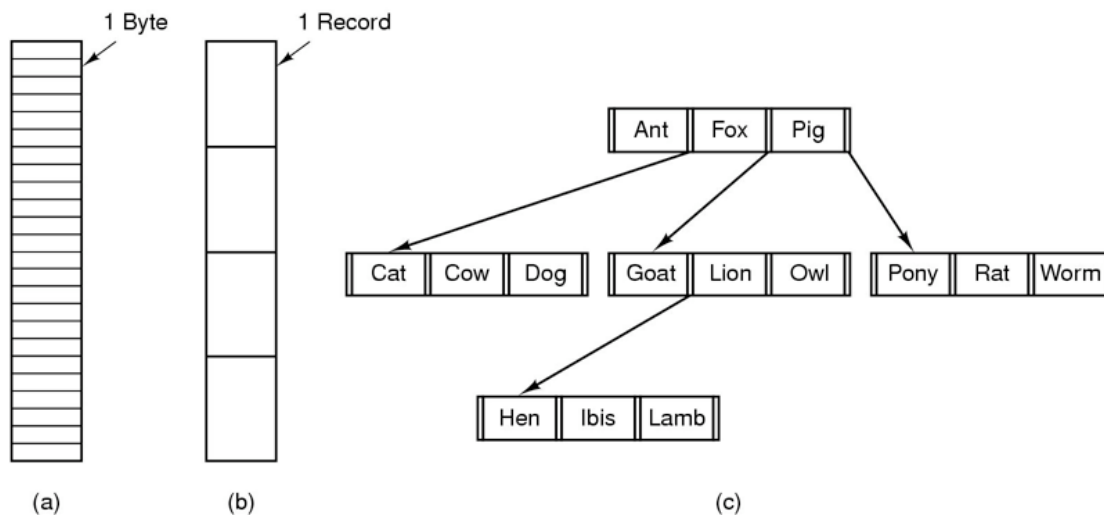
Eine Datei besteht aus 3 Elementen:

- Name (eigentlich: Pfad)
- Dateiattribute
- Eigentlicher Dateiinhalt

Dateisysteme erlauben für D. unterschiedliche Namen

- Längen von Namen (heute i.d.R 255 Zeichen)
- Sonderzeichen ggfls. erlaubt (in Unix ist lediglich der „/“ verboten)
- Unterscheidung Groß/Kleinschreibung (Unix: meist ja, Windows:nein)
- „Dateierweiterung“ (3-Buchstaben für den Dateityp) (Unix: der Punkt ist ein Zeichen wie jedes andere, Windows: Endung steuert Zuordnung zu Programmen)

Dateistrukturen



- (a) Datei als unstrukturierte Sequenz von Bytes (Unix, Windows)
 (b) Datei als Folge von Records (Mainframes, heute eher unüblich)
 (c) Datei als Baum (z.B. Resource-Fork bei MacOSX)

Zentrale Idee von Unix: „Alles ist als Datei modellierbar, auch Terminals, Speicher, Konsolen, Prozesse, IPC, Netzwerkverbindungen

- Minimierung der Anzahl verschiedener Systemaufrufe, einheitliche API
- Werkzeugcharakter von Unix

Dateitypen

- Reguläre Dateien (Textdateien, Binärdateien)
- Verzeichnisse
- Spezielle E/A-Dateien (IPC, Tastatur, ...)
- Verweise auf andere Dateien

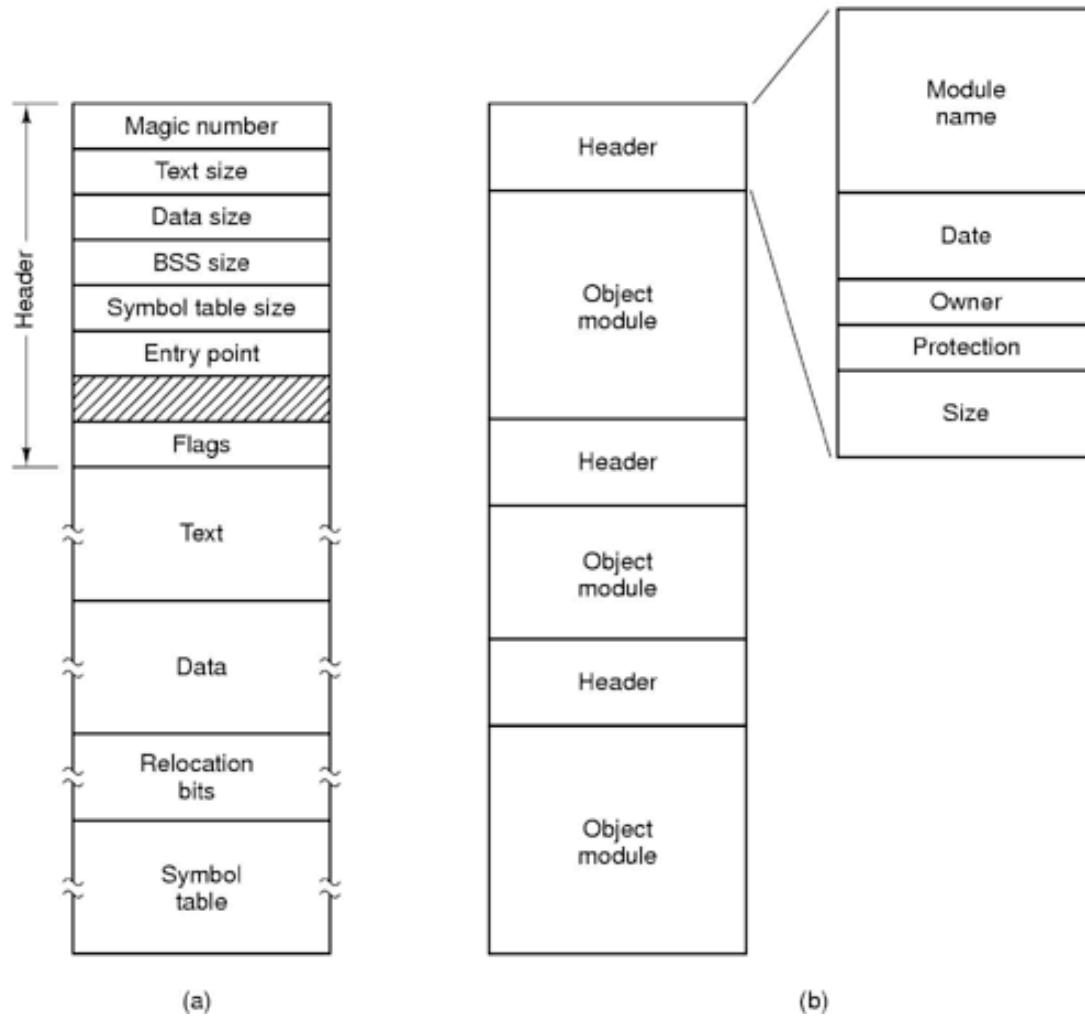
Textdateien:

- Folge von Zeilen unterschiedlicher Länge
- Zeilenendezeichen
- Inhalt interpretierbar gemäß Zeichensatz (ASCII, EBCDIC, Unicode, Latin-1)
- Mit Texteditor editierbar

Binärdateien:

- Interpretation anwendungsabhängig
- Verwendungszweck meist als Dateiendung angegeben (Krücke!)
- Hexeditor, Programm

Unterschied wird bei der Übertragung relevant (Zeichensatz/Zeilenende-konvertierung)



- a) Eine ausführbare Datei
b) Ein Archiv

Dateioperationen

- Create: Ankündigung des Schreibens. Setzen von Name und Defaultattributen
- Delete: Platz auf der Platte freigeben
- Open: Vorbereiten zum Lesen. Prüfen der Berechtigung. Obergrenze.
- Close: Freigeben der Ressource
- Read: Lesen an der aktuellen Position
- Write: Überschreiben oder anhängen von Daten
- Append: Öffnen zum anhängen von Daten
- Seek: Setzen des Zeigers für Read/Write
- Rename
- Get Attributes
- Set Attributes

Dateiattribute

Art und Ausprägung zwischen BS teils stark unterschiedlich.

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Verzeichnisse (Ordner, Directories)

Ein V. ist eine sequentielle Datei.

Simpelste Form: Es gibt nur ein einziges Verzeichnis.

Üblich: Hierarchischer Verzeichnisbaum

Windows: Wald von Verzeichnisbäumen mit Laufwerksbuchstaben

Unix: Ein einziger Verzeichnisbaum, Medien sind in den Baum „eingeklinkt“

Bennennung einer Datei mit Pfadnamen:

- Absolute Pfadnamen: /home/herwig/www/ lehre/betriebssysteme/vl06.pdf, erkennbar am Pfadtrenner zu Beginn
- Relative Pfadnamen: Basis ist das „aktuelle Verzeichnis“:

```
cd /home/herwig/incoming
cp vl06.pdf ../lehre/betriebssysteme/vl06.pdf
```

 Jeder Prozess hat sein eigenes Arbeitsverzeichnis.

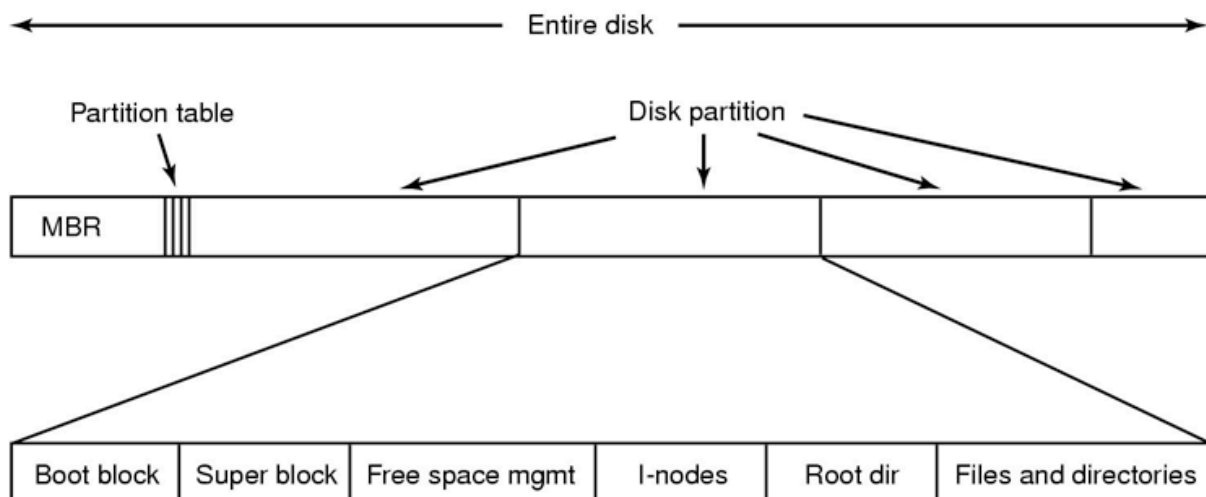
- . als Name für das aktuelle Verzeichnis
- .. als Name für das Vorgängerverzeichnis

Operationen auf Verzeichnissen

- Create: Neues V., enthält nur . und ..
- Delete: Löschen eines leeren Verzeichnisses
- Openidir: Öffnet Datei zum Lesen der Dateinamen
- Closedir
- Readdir: Liest das nächste Verzeichnis
- Rename
- Link: Erzeugen eines Verweises auf eine andere Datei
- Unlink

Frage: Warum kein Writedir?

Implementierung von Dateisystemen



- Master Boot Record: Wird zum Booten des Rechners benutzt
- Partitionstabelle: Enthält Start/Ende und Typ der Partitionen
- Bootblock: Enthält das Programm, welches beim Booten des Rechners geladen wird
- Superblock: Identifiziert die Eigenschaften des Dateisystems
- Informationen über freie Blöcke
- I-Nodes: Array von Dateiattributen
- Root dir: das Wurzelverzeichnis
- Files and directories: Die Dateien auf der Partition

Speicherung von Dateien (Analogie zur Speicherverwaltung!)

- Einfach: Dateien hintereinanderhängen
- Komplex: Dateien „virtuell“ auf Blöcke abbilden. Jede Datei hat eigenen „Adressraum“
- Seitentabelle gehört zu den Dateiattributen

Dateiattribute werden in den i-Nodes (Index-Knoten) gespeichert (nur die erste Seite einer mehrstufigen Seitentabelle). i-Nodes haben eine feste Größe (üblicherweise ein Block).

Konsequenzen:

- Eine Datei belegt mindestens einen Block
- BS verwaltet Liste freier Seiten
- Tiefe der Seitentabelle kann wachsen, wenn Datei wächst. Zu Beginn einstufig, dann beliebig mehrstufig. Üblicherweise bis zu 4-Stufig.
- Dateien können Löcher haben
- Unterschied zur Speicherverwaltung: Informationsänderungen müssen immer weggeschrieben werden. Also auch Änderungen an i-Nodes und Freiseitenliste.
- Ein Verzeichnis ist eine Liste von Namen mit dazugehörigen i-Nodes
- Jede Datei hat eine i-Node. Eine i-Node kann mehrere Namen haben (!!)

Großes Beispiel:

- Datei /home/herwig/vl06.pdf wird geöffnet zum Lesen (z.B. weil der Webserver sie via Netz verschicken will)
- Datei aktuell.pdf ist die gleiche Datei!
- Blöcke wissen ihren Zweck nicht. Sie können Verzeichnisse sein oder Seitentabellen enthalten oder Inhalte
- Aus dem Dateinamen ist der Dateityp nicht ersichtlich
- . und .. sind normale Links auf die entsprechenden i-Nodes
- I-Nodes hier Blöcke im besonderen Bereich (dies ist heute nicht mehr so)
- Datei vl06.pdf ist so groß, das zweite Indirektionsstufe benutzt wird

Konsistenz der Datenstrukturen enorm wichtig (Filesystemcheck).

Fragmentierung von Dateien.

Prozesse können Dateien gleichzeitig lesen/schreiben.

Sperren über Locks möglich (Ein P. kann anderen das Schreiben verbieten).

Link-Count einer Datei

- Jede i-Node enthält eine Zahl von Links, die auf sie zeigt
- Normal: Anzahl der Dateinamen mit der i-Node
- Jeder Prozeß, der Datei öffnet erhöht den Linkcount um 1
- Beim schließen der Datei wird der Link-Count um 1 verringert
- Beim löschen eines Dateinamens wird der Link-Count um eins verringert.
- Fällt der Link-count auf 0, so wird Datei gelöscht und i-Node freigegeben.

Folge: Die Datei wird nicht durch del/rm gelöscht!

Löschen i.Allg. nur durch Aufnahme der Blöcke in die Freelist. Der Inhalt bleibt bestehen, bis Blöcke überschrieben werden.

Leistungsfähigkeit eines Filesystems

- Buffer-Cache: Blöcke werden in einem Cache gehalten. Üblicherweise werden dazu sämtliche Blöcke aus der Speicherfreiliste verwendet (!)
- Verzögern der Schreiboperationen
 - o Falls Block wieder gelöscht wird, entfällt schreiben
 - o Optimieren der Plattenkopfbewegungen
 - o Verringern von Fragmentierungen
 - o Aber: Konsistenzerhaltung schwieriger
- Vorauslesen von Blöcken
- Journaling (Blöcke werden sequentiell geschrieben und von einem separaten Prozess aufgeräumt)

Netzwerkfilesystems

- NFS, AFS, SAMBA (Unix)
- SMB (Windows)

Benutzerrechte in Unix

- Jeder Benutzer meldet sich mit seinem Login (und Passwort) an.
- Jeder Benutzer gehört ausserdem zu einer Gruppe.
- Eine Datei gehört demjenigen, der sie anlegt und darf die Rechte verändern.
- Dateien darf man verschenken.
- Jeder Datei hat Zugriffsrechte für Benutzer, Gruppe and „alle anderen“

- Für jede Benutzergruppe lassen sich die Rechte „lesen“, „schreiben“ und „ausführen“ vergeben.

Binary	Symbolic	Allowed file accesses
111000000	rwX-----	Owner can read, write, and execute
111111000	rwXrwx---	Owner and group can read, write, and execute
110100000	rw-r-----	Owner can read and write; group can read
110100100	rw-r--r--	Owner can read and write; all others can read
111101101	rwXr-Xr-X	Owner can do everything, rest can read and execute
000000000	-----	Nobody has any access
000000111	-----rwx	Only outsiders have access (strange, but legal)

- Jeder Benutzer hat ein „Heimatverzeichnis“ (i.Allg. unter /usr/<username> oder /home/<username>)
- Die Benutzerrechte steuern nur das Recht, etwas mit dieser Datei zu machen aber nicht die Rechte zur Ausführungszeit!
- Besonderer Benutzer „root“. Dieser darf *alles*. (root ist auch die Bezeichnung für das oberste Verzeichnis)
- setuid-bit: Ist dieses gesetzt, so erhält man beim Ausführen einer Datei die Rechte dieses Benutzers. Problematisch: Programme, die root gehören und das setuid-bit gesetzt haben.
- Eine Datei kann zusätzlich ACL (access control lists) besitzen, welche eine feinere Granulierung als obiges Schema erlaubt.

Besondere Dateiart „Symbolic links“.

- Diese sind Verweise auf eine andere Datei. Referenziert wird über den Pfad/Dateinamen und nicht die gleiche i-Node.
- Konsequenzen: Können auf Dateien auf anderen Partitionen zeigen
- Beim Wegbewegen der Zieldatei ist der Link „kaputt“.
- Der Pfadname ist „Inhalt“ der Datei.

Anmerkungen:

- Moderne Dateisysteme benutzen Blöcke meist für mehrere kleine Dateien
- Die Anzahl der i-Nodes muss nicht mehr beim Einrichten des Dateisystems bestimmt werden und ist dynamisch

```
-bash-2.05b$ ls -la /
total 8997
drwxr-xr-x 19 root wheel      512 Oct 29 19:53 .
drwxr-xr-x 19 root wheel      512 Oct 29 19:53 ..
-rw-r--r--  2 root wheel      802 Apr  3  2003 .cshrc
-rw-r--r--  2 root wheel      251 Apr  3  2003 .profile
-r--r--r--  1 root wheel     4735 Apr  3  2003 COPYRIGHT
drwxr-xr-x  2 root wheel     1024 Oct 29 19:54 bin
drwxr-xr-x  3 root wheel      512 Oct 29 19:56 boot
drwxr-xr-x  2 root wheel      512 Jun  2 21:23 cdrom
lrwxr-xr-x  1 root wheel        10 Jun  2 21:25 compat -> usr/compat
drwxr-xr-x  3 root wheel    17920 Oct  2 20:59 dev
drwxr-xr-x  3 root wheel      512 Jun  4 09:07 disk1
drwxr-xr-x  2 root wheel      512 Jun  2 21:23 dist
drwxr-xr-x 16 root wheel     2560 Oct  2 21:52 etc
lrwxrwxrwx  1 root wheel         9 Jun  2 21:32 home -> /usr/home
-r-xr-xr-x  1 root wheel  2474551 Oct 29 19:49 kernel
-r-xr-xr-x  1 root wheel  4122347 Apr  3  2003 kernel.GENERIC
-r-xr-xr-x  1 root wheel  2473593 Oct  4 10:41 kernel.old
drwxr-xr-x  2 root wheel      512 Apr  3  2003 mnt
drwxr-xr-x  2 root wheel     4096 Oct 29 19:50 modules
drwxr-xr-x  2 root wheel     4096 Oct  4 10:41 modules.old
dr-xr-xr-x  1 root wheel      512 Nov  9 12:54 proc
drwxr-xr-x  3 root wheel      512 Jun  7 18:12 root
drwxr-xr-x  2 root wheel     2048 Oct 29 19:55 sbin
drwxr-xr-x  5 root wheel     1024 Jun  2 21:23 stand
lrwxr-xr-x  1 root wheel        11 Oct 29 19:53 sys -> usr/src/sys
drwxrwxrwt  5 root wheel      512 Nov  9 05:31 tmp
drwxr-xr-x 18 root wheel      512 Oct  4 10:52 usr
drwxr-xr-x 23 root wheel      512 Oct 30 10:33 var
-bash-2.05b$
```