FH Bielefeld
University of
Applied Sciences

Vorlesung

Betriebssysteme

Teil 9
Speicherverwaltung
und
Dateisysteme

FH Bielefeld
University of
Applied Sciences

Ziele der Vorlesung

- Wiederholung und Abschluss des Themas Speicherverwaltung

- Dateisysteme kennenlernen
  - Details der File Allocation Table
  - Prinzipien der Unix-Dateiverwaltung
- Übungsaufgabe

FH Bielefeld
University of
Applied Sciences

Seitenersetzung

Beispiel für LRU-Algorithmus:

Referenz-string	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
optimal																				
Rahmen 0	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
Rahmen 1		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
Rahmen 2			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
FIFO																				
Rahmen 0	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
Rahmen 1		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
Rahmen 2			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
LRU																				
Rahmen 0	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
Rahmen 1		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
Rahmen 2			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7

optimal: 9 Ersetzungen
FIFO: 15 Ersetzungen
LRU: 12 Ersetzungen

FH Bielefeld
University of
Applied Sciences

Seitenersetzung: Beladys Anomalie

- Die Zahl der Seitenrahmen wird erhöht (mehr Speicher):

Referenz-string		0	1	2	3	0	1	4	0	1	2	3	4
FIFO	Rahmen 0	0	0	0	3	3	3	4	4	4	4	4	4
	Rahmen 1		1	1	1	0	0	0	0	0	2	2	2
	Rahmen 2			2	2	2	1	1	1	1	1	3	3

-> 9 Seitenfehler

Referenz-string		0	1	2	3	0	1	4	0	1	2	3	4
FIFO	Rahmen 0	0	0	0	0	0	0	4	4	4	4	3	3
	Rahmen 1		1	1	1	1	1	1	0	0	0	0	4
	Rahmen 2			2	2	2	2	2	2	1	1	1	1
	Rahmen 3				3	3	3	3	3	3	2	2	2

-> 10 Seitenfehler

- Eine Erhöhung der Seitenrahmen verringert nicht zwangsläufig die Zahl der Seitenfehler! (Beladys Anomalie)

Seitenersetzung: Second Chance Strategie

- Abart des FIFO-Algorithmus, die verhindert, dass häufig benutzte Seiten ausgelagert wird.
- Algorithmus: **R-Bit** der ältesten Seite wird via **Timer** geprüft:
  - R-Bit nicht gesetzt, dann wird die Seite ausgelagert.
  - R-Bit gesetzt, dann wird das Bit gelöscht und die Seite an den Anfang der FIFO-Liste verlagert. Die nächste Seite in der FIFO-Liste wird geprüft.
- Ist bei allen Seiten das R-Bit gesetzt, degeneriert der Algorithmus zum FIFO-Algorithmus, da die älteste Seite mit gelöschtem R-Bit durchgeschoben wird.
- Eine Abart von *Second Chance* mit einem **Ring-Puffer** statt einer FIFO-Liste wird ***Clock-Algorithmus*** genannt.
  - Ring-Puffer Verwaltung ist effizienter als Umhängen in Listen

FH Bielefeld
University of
Applied Sciences

Seitenersetzung: Working-Set Strategien

- Die meisten Prozesse starten mit einer Seite und laden dann bei Bedarf Seiten nach (*Demand Paging*):
  - *Nach einer gewissen Zeit* hat er dann aufgrund der Lokalität der Referenzen die wichtigsten Seiten in den Speicher geladen und produziert **nur noch wenige Seitenfehler**.
  - Menge der Seiten, die ein Prozess zu einem bestimmten Zeitpunkt benutzt heißt **Working Set**.
- Bei Working Set Modellen
  - **merkt sich das Betriebssystem den Working Set** eines Prozesses beim Auslagern und
  - lädt ihn komplett wieder ein, bevor der Prozess weiterarbeitet um so Seitenfehler zu vermindern.
- Strategien, die Seiten laden, noch bevor sie gebraucht werden, werden auch ***Prepaging*-Verfahren** genannt.

FH Bielefeld
University of
Applied Sciences

Entladestrategien (Cleaning)

- Legt den Zeitpunkt fest, wann eine modifizierte Seite auf die Paging-Area geschrieben wird
- Varianten:
  - **Demand-Cleaning**: Bei Bedarf
  - Vorteil: Seite lang im Hauptspeicher
  - Nachteil: Verzögerung bei Seitenwechsel
  - **Precleaning**: Präventives Zurückschreiben, wenn Zeit ist
  - Vorteil: Frames in der Regel verfügbar
  - **Page-Buffering**: Listen verwalten
  - Modified List: Wird zwischengepuffert
  - Unmodified List: Für Entladen freigegeben
  - Heute üblich (siehe Windows)

FH Bielefeld
University of
Applied Sciences

Realisierung virtueller Speichertechnik

- Das Betriebssystem bzw. der Memory-Manager muss mehrere Strategien implementieren. Hierzu gehört die
- Abrufstrategie (Fetch Policy, Varianten sind Demand Paging oder Prepaging), die Speicherzuteilungsstrategie (Placement Policy),
- die Austauschstrategie (Replacement Policy = Seitenersetzungs- bzw. Verdrängungsstrategie)
- und die Aufräumstrategie (Cleaning Policy).

Speicherverwaltung unter Unix

- Frühere Unix-Systeme bis zu BSD 3 nutzten ausschließlich Swapping
  - Ein Prozess namens swapper (daemon) mit PID 1 übernahm das Swapping bei bestimmten Ereignissen bzw. zyklisch im Abstand von mehreren Sekunden
- Ab BSD 3 wurde Demand Paging ergänzt, alle anderen Unix-Derivate (System V) haben es übernommen
  - Ein sog. page daemon wurde eingeführt (PID 2)
  - Im page daemon ist der Seitenersetzungsalgorithmus nach einem Clock-Page Algorithmus implementiert
  - Heute: Variationen je nach Unix-Derivat

FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter Linux

- Bei 32-Bit-Linux:
  - Virtuelle Adressen mit 32 Bit Länge, 1 GB für den Kernel und die Seitentabellen, restliche 3 GB für den User-Prozess
- Bei 64-Bit-Linux:
  - 48-Bit-Adressen und Adressraum der Größe 2^{48} Byte
- Adressumsetzung:
  - Linux verwendet dreistufige Seitentabellen bis Version 2.6.10, ab Linux-Version 2.6.11 sogar vierstufige Seitentabellen
  - Evtl. Mapping auf zweistufige oder sonstige Seitentabelle, wenn Hardware es nicht kann

FH Bielefeld
University of
Applied Sciences

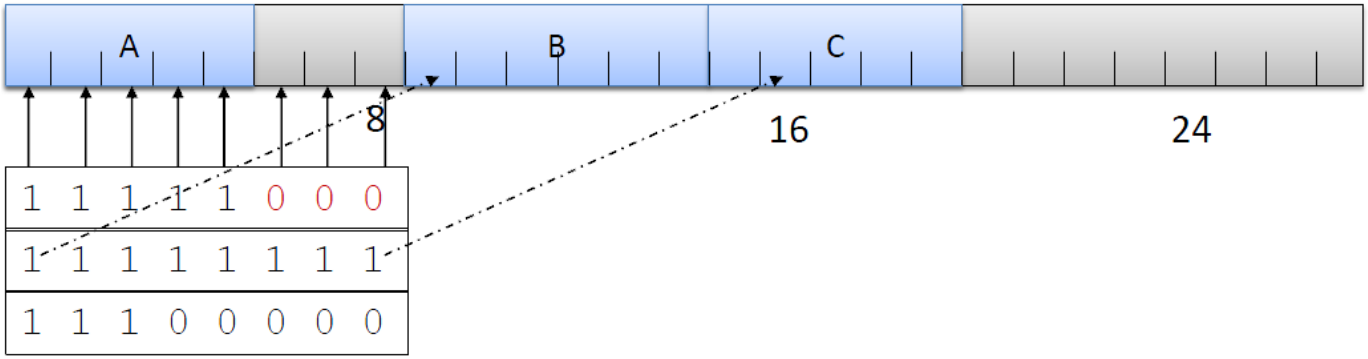
Speicherverwaltung unter Linux

- Fetch-Policy:
  - Als Einlagerungsstrategie wird Demand Paging ohne Prepaging und ohne Working Set verwendet
- Replacement- und Cleaning-Strategie:
  - Replacement über eine Art Clock-Page-Algorithmus
  - Verwaltung mehrerer Listen mit Seitenrahmen (Page Buffering)
  - Mehrere Kernel-Threads zur Listenbearbeitung:
    - kswapd überprüft periodisch die Listen und lagert bei Bedarf um
    - bdflush (ab 2.6 pdflush) schreibt periodisch veränderte („dirty“) Seiten auf die Paging-Area
- Placement-Policy:
  - Speicherbelegung erfolgt über Buddy-Technik

FH Bielefeld
University of
Applied Sciences

Speicherbelegungsstrategien

- Vermeidung von Fragmentierung ist anzustreben
- Die Belegung des Hauptspeichers wird in Speicherbelegungstabellen verwaltet
- Die Realisierung kann z.B. als Bit Map erfolgen:
- Jedem Rahmen wird ein Bit zugeordnet
  - 0 = frei
  - 1 = belegt
- Freie Hauptspeicherbereiche erkennt man dann an nebeneinander liegenden Nullen



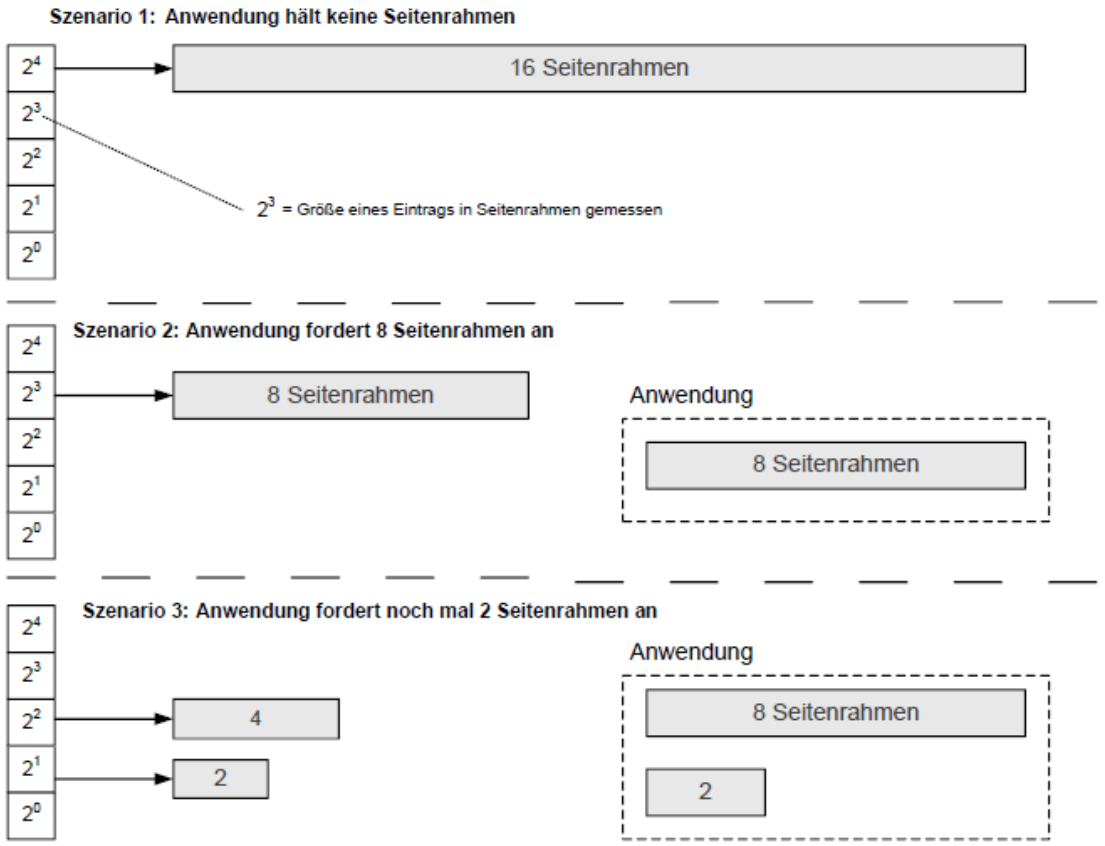
FH Bielefeld
University of
Applied Sciences

Speicherbelegungstrategien: Suche nach freien Seiten

- Vergabestrategien
- Sequentielle Suche, erster geeigneter Bereich wird vergeben (First-Fit)
- Optimale Suche nach dem passendsten Bereich, um Fragmentierung möglichst zu vermeiden (Best-Fit)
- Buddy-Technik: Schrittweise Halbierung des Speichers bei einer Hauptspeicheranforderung
  - Speichervergabe:
  - Suche nach kleinstem geeigneten Bereich
  - Halbierung des gefundenen Bereichs solange bis gewünschter Bereich gerade noch in einen Teilbereich passt
  - Bei Hauptspeicherfreigabe werden Rahmen wieder zusammengefasst:
  - Zurückgegebenen Bereich mit allen freien Nachbarbereichen (und deren Partnern) verbinden und zu einem Bereich machen

FH Bielefeld
University of
Applied Sciences

Speicherbelegungstrategien: Buddy-Technik

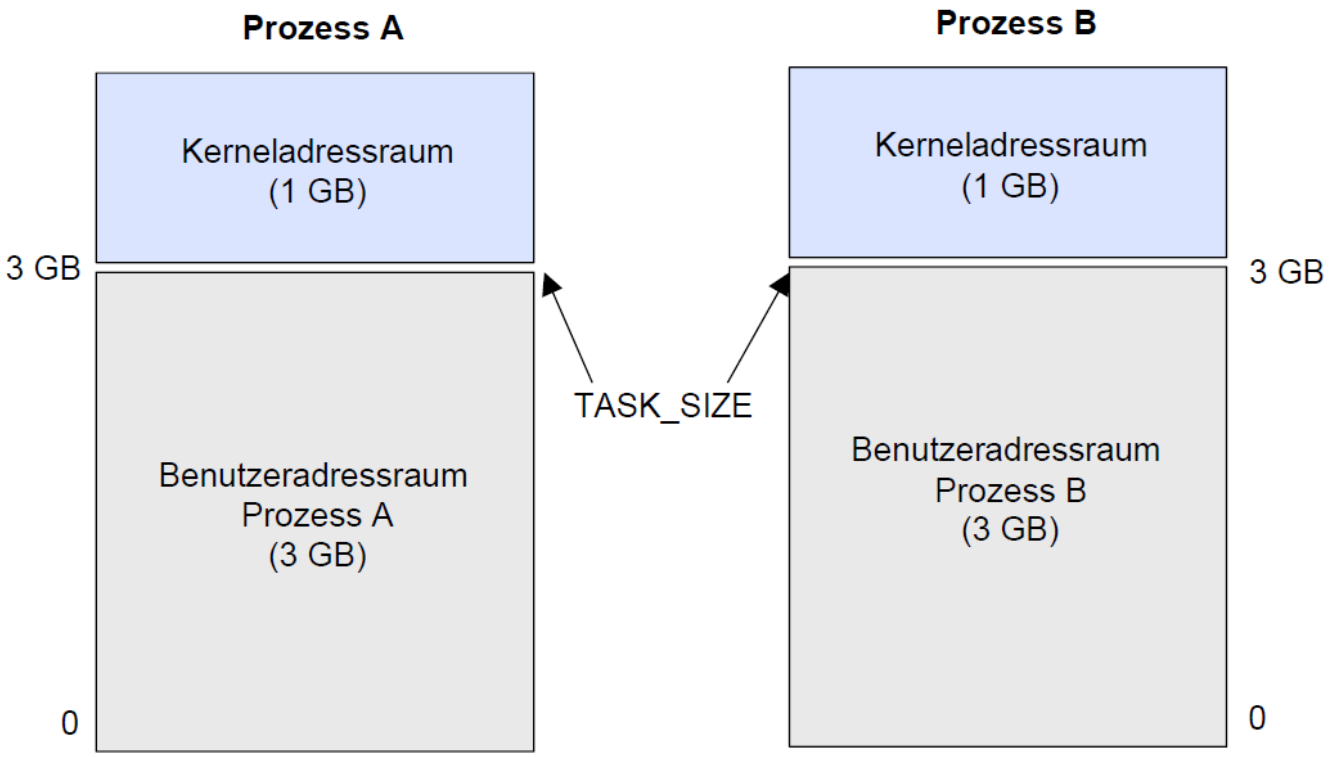


- Reduziert externe Fragmentierung auf Kosten einer verstärkten internen Fragmentierung!

FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter Linux

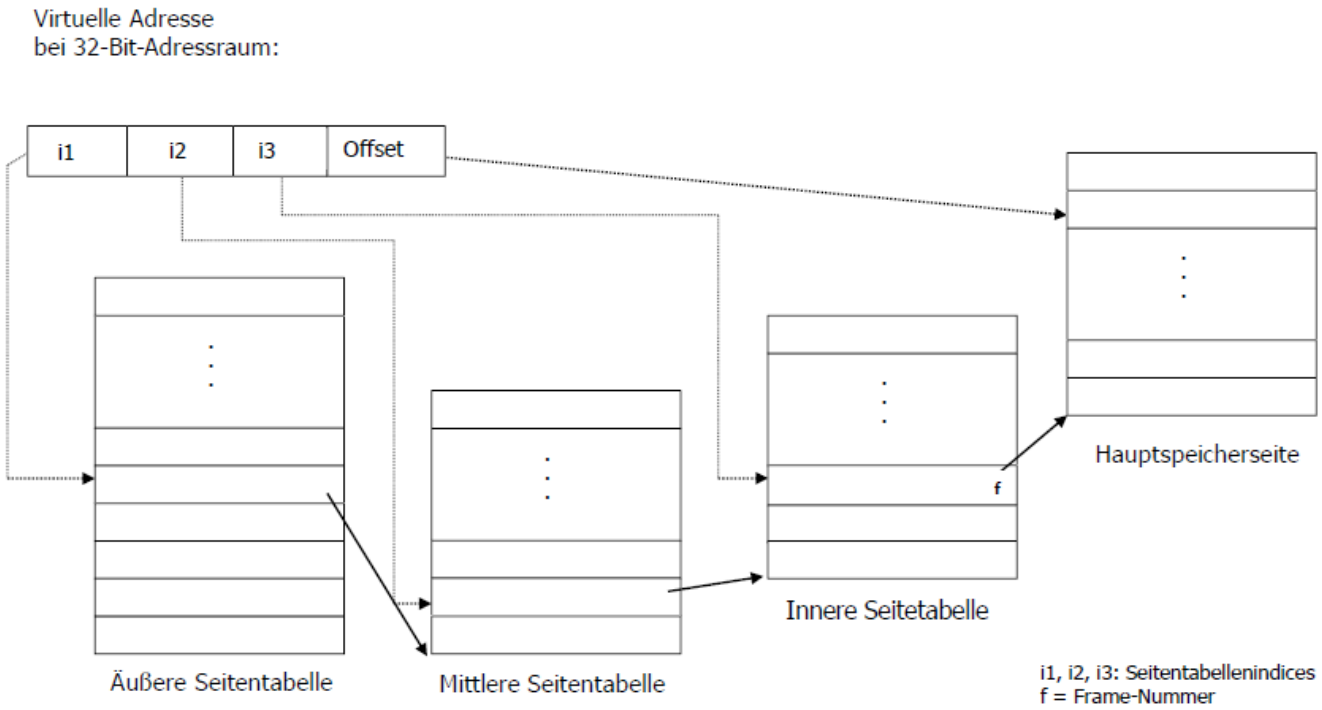
- Adressraumbelegung bei 32-Bit-Architektur



FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter Linux (32-Bit)

- Virtuelle 32 Bit Adressen, hier: dreistufige Seitentabellen
- Abbildung bei Intel-Pentium auf zweistufiges Verfahren (Pentium unterstützt nur zwei Stufen)



Speicherverwaltung unter 32-Bit-Windows

- Virtuelle Adressen mit 32 Bits Länge, also 4 GB Adressraum, 2 GB davon für den User-Prozess und der Rest für den Kernel
  - linearer Adressraum ohne Segmentierung
- Seitengröße abhängig von Prozessorarchitektur:

Prozessor-architektur	Größe der Small Page	Größe der Large Page
X86	4 KB	4 MB
x64 (AMD)	4 KB	2 MB
IA64 (Intel)	8 KB	16 MB

Speicherverwaltung unter 32-Bit-Windows

- Fetch-Policy:
  - Nutzung von Demand Paging
  - Ab Windows 2003 wird auch Prepaging verwendet

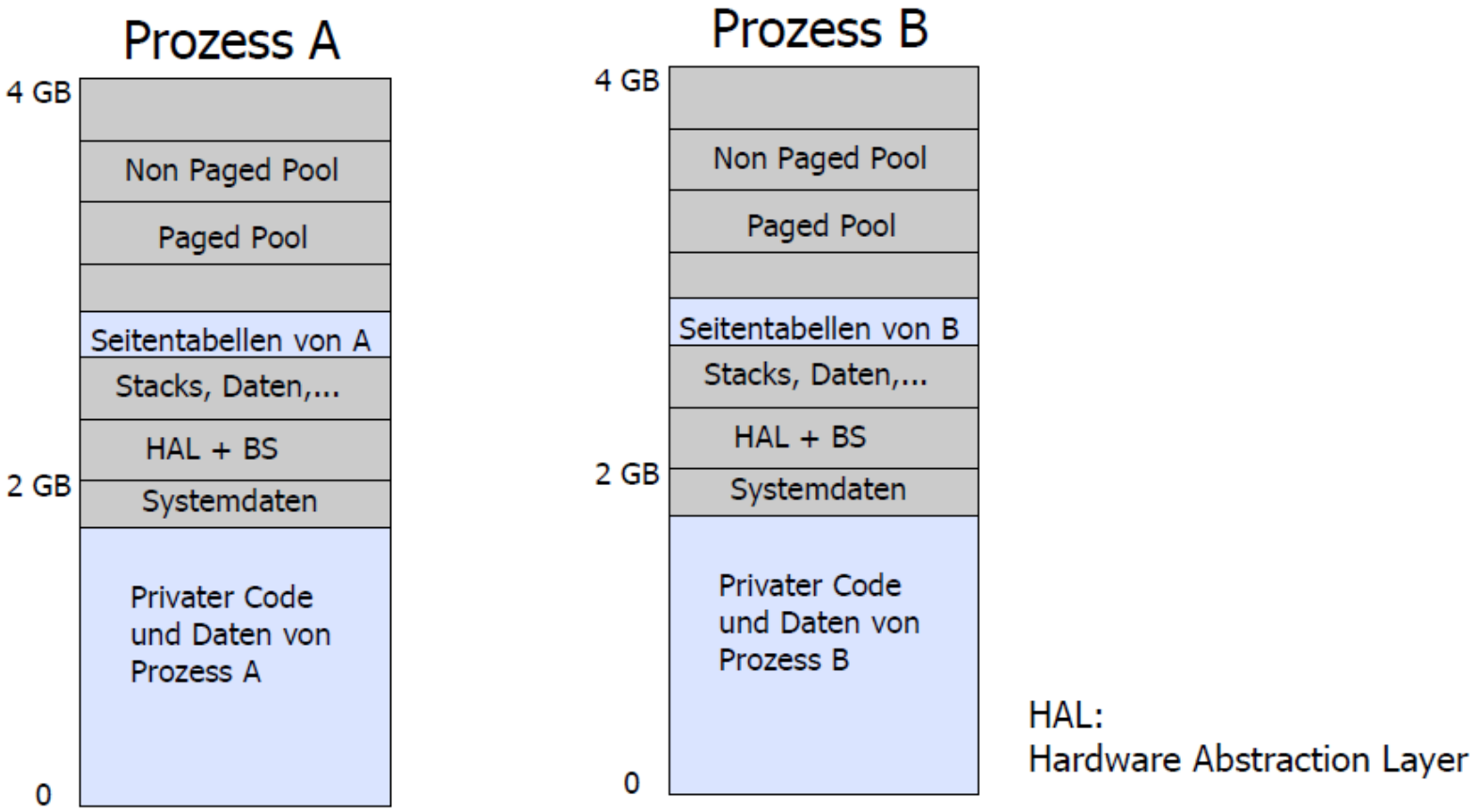
- Replacement- and Cleaning-Policy:
  - Kombination aus lokaler und globaler Ersetzungsstrategie
  - Eigenes Working-Set-Verfahren
  - FIFO bei Multiprozessormaschinen
  - Clock-Page bei Einprozessormaschinen
  - Mehrere Auslagerungslisten werden verwaltet
  - Mehrere Threads bearbeiten die Listen

- Placement-Policy:
  - Nicht näher erläutert

FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter 32-Bit-Windows

- Aufbau eines virtuellen Adressraums (vgl. Tanenbaum)



FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter 32-Bit-Windows

- Working Sets
  - Jeder Prozess hat einen Working Set mit einer veränderbaren Größe (Minimum 50 Seiten, Maximum 345 Seiten je nach vorhandenem Speicher)
  - Bei einem Seitenfehler wird nicht über den maximalen eigenen Working Set eines Prozesses eingelagert
  - Ausnahme:
    - Ein Prozess „paged“ stark und andere nicht, dann wird der „pagende“ Prozess erhöht, aber nicht mehr als die verfügbaren Seitenrahmen - 512, so dass immer noch ein paar Seitenrahmen frei bleiben

FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter 32-Bit-Windows

- Ein zyklisch arbeitender Working Set Manager Thread versucht zusätzlich nach einem komplizierten Verfahren freie Seitenrahmen zu besorgen
- Ein Seitenrahmen (Frame) ist
  - entweder einem (oder mehreren) Working Set(s) zugeordnet
  - oder genau einer von vier Listen, in denen Windows freie Seitenrahmen verwaltet

Speicherverwaltung unter 32-Bit-Windows

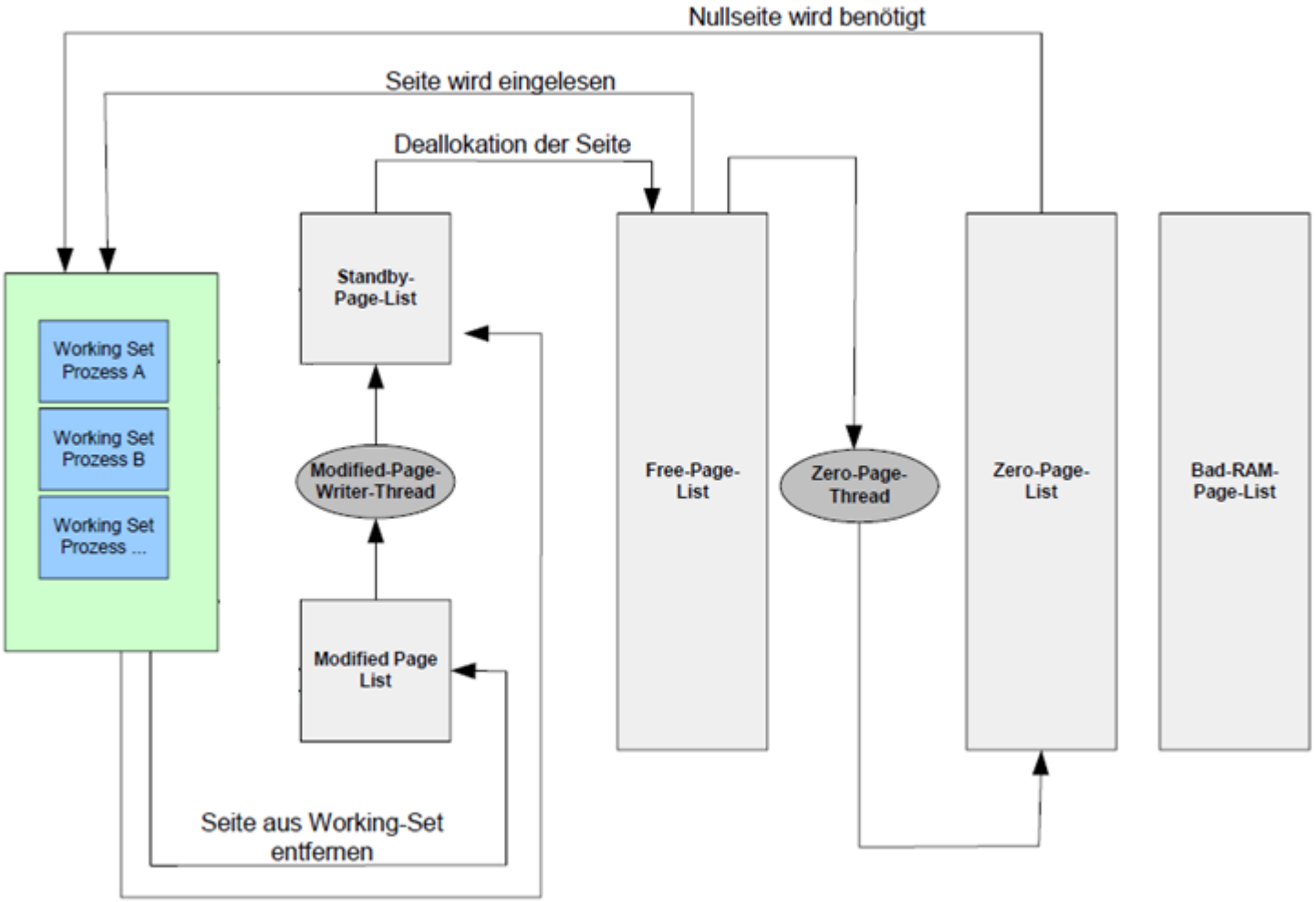
- Die Listen im Einzelnen:
- Modified-Page-List
  - Seiten, die bereits für die Seitenersetzung ausgewählt wurden, aber noch nicht ausgelagert wurden und auch dem nutzenden Prozess noch zugeordnet sind
- Standby-Page-List
  - Wie modified page list, mit dem Unterschied dass sie „clean“ sind, also eine gültige Kopie auf der Paging Area haben
- Free-Page-List
  - Frames, die bereits „clean“ sind und keinem Prozess mehr zugeordnet sind
- Zero-Page-List
  - Wie die free page list und zusätzlich mit Nullen initialisiert
  - Weitere Liste hält defekte Speicherseiten (Bad-RAM-Page-List)

Speicherverwaltung unter 32-Bit-Windows

- Einige Threads arbeiten an der Verwaltung dieser Listen mit
- Swapper-Thread:
  - Läuft alle paar Sek., sucht nach Prozessen, die schon länger nichts tun (idle) und legt deren Frames in die Modified- oder Standby-Page-List
- Modified-Page-Writer-Thread:
  - Laufen periodisch und sorgen für genügend saubere Seiten durch Umschichtung von der Modified-Page-List in die Standby-Page-List (vorher wird auf Platte gesichert)
- Zero-Page-Thread:
  - Läuft mit niedriger Priorität, löscht Frames aus der Free-Page-List und legt sie in die Zero-Page-List

FH Bielefeld
University of
Applied Sciences

Speicherverwaltung unter 32-Bit-Windows



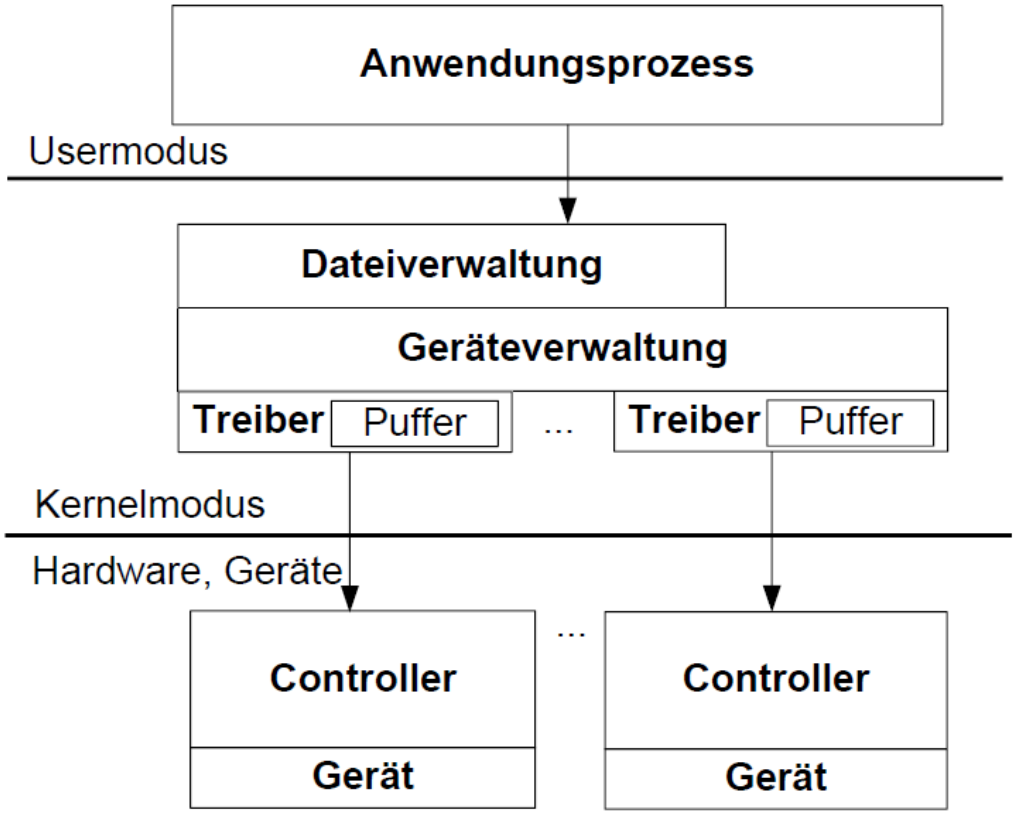
FH Bielefeld
University of
Applied Sciences

Zusammenfassung

- Früher: …, Partitionierung, Swapping
- Heute zumeist:
  - Virtueller Speicher:
    - Segmentierung: Speicherbereiche variabler Größe
    - Paging:
      - Lineares Speichermodell: Abbildung virtuelle → physikalische Adressen, Seiten werden dynamisch geladen und entladen → Seitenersetzungsstrategien
      - Probleme bei großer Anzahl virtueller Seiten (→ TLB, inverted PT)
  - Standard: Kombination von Paging und Segmentierung
- Unterstützung durch Hardware:
  - MMU, siehe Pentium

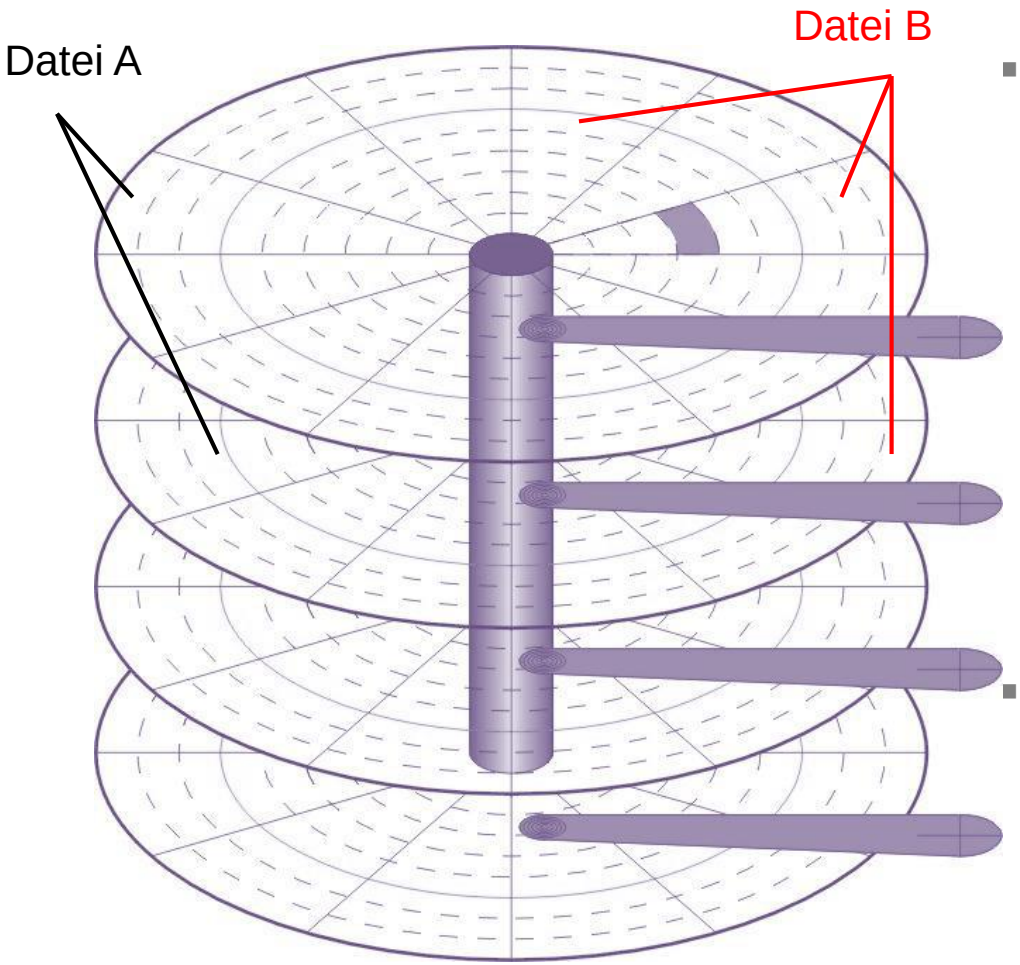
FH Bielefeld
University of
Applied Sciences

Treiber und Dateisysteme im Betriebssystem



FH Bielefeld
University of
Applied Sciences

Dateisysteme: Grundlagen Datenorganisation



- Festplatte als Speichermedium:
  - Besteht aus mehreren magnetischen **Scheiben**, Zugriff über **Schreib-/ Leseköpfe**
  - Scheiben eingeteilt in **Spuren** (konzentrische Kreise), übereinander liegende Spuren: **Zylinder**
  - Benachbarte Spuren: **Zonen**
  - Bereiche der Spuren / Zylinder: **Sektoren**
- Zusammenhänge Daten (z.B. Datei) können physikalisch verteilt sein über Medium (*Fragmentierung*)

FH Bielefeld
University of
Applied Sciences

Block Size Linux

```
martin@redstar: ~/test
--getro                   get read-only
--getdiscardzeroes        get discard zeroes support status
--getss                   get logical block (sector) size
--getpbsz                 get physical block (sector) size
--getiomin                get minimum I/O size
--getioopt                get optimal I/O size
--getalignoff             get alignment offset in bytes
--getmaxsect              get max sectors per request
--getbsz                  get blocksize
--setbsz <bytes>          set blocksize
--getsize                 get 32-bit sector count (deprecated, use --getsz)
--getsize64               get size in bytes
--setra <sectors>         set readahead
--getra                   get readahead
--setfra <sectors>        set filesystem readahead
--getfra                  get filesystem readahead
--flushbufs               flush buffers
--rereadpt                reread partition table

martin@redstar:~/test$ sudo blockdev --getbsz /dev/sda1
4096
martin@redstar:~/test$ sudo blockdev --getpbsz /dev/sda1
512
martin@redstar:~/test$
```

Dateisysteme: Dateioperationen

- Beispiele für Dateioperationen:

Beschreibung	Unix (POSIX)	Win32-API-Funktion
Erzeugen oder Öffnen einer Datei	`open`	`CreateFile`
Zerstören einer Datei	`unlink`	`DeleteFile`
Schließen einer Datei	`close`	`CloseHandle`
Daten aus Datei lesen	`read`	`ReadFile`
Daten in Datei schreiben	`write`	`WriteFile`
Lesezeiger setzen	`lseek`	`SetFilePointer`
Dateiattribute ermitteln	`stat`	`GetFileAttributes`
Bereich einer Datei gegen Mehrfachzugriff sperren	`fcntl`	`LockFile`
Gesperrten Bereich freigeben	`fcntl`	`UnlockFile`

FH Bielefeld
University of
Applied Sciences

Dateisysteme: Spezielle Dateien

Neben **regulären Dateien** gibt es noch weitere Dateiarten:

- **Gerätedateien**:
  - Einblendung von Geräten als spezielle Dateien im Dateisystem (z.B. **/dev/sda1** , verweist auf den Partition 1 der Festplatte a.)
  - Block- oder zeichenorientierte Gerätedateien möglich.
- **Prozess Dateien**:
  - Das **/proc** Dateisystem gibt Informationen zu den laufenden Prozessen aus. (z.B. **/proc/<PID>/pagemap**)
  - Die Dateien werden zur Laufzeit vom Betriebssystem dynamisch erzeugt und geben aktuelle Prozessinformationen aus.
  - Einfache Schnittstelle zwischen Betriebssystem und systemnaher Programmierung.
- **Pipe-Dateien**:
  - Dienen der Kommunikation zwischen Prozessen

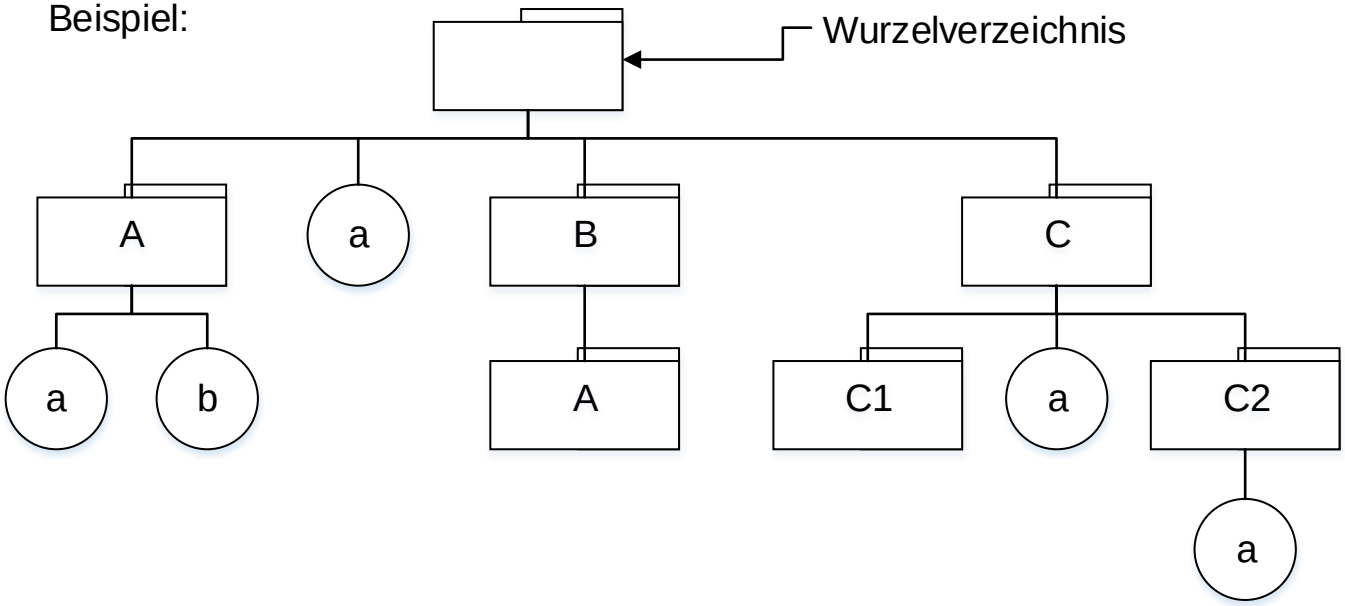
FH Bielefeld
University of
Applied Sciences

Dateisysteme: Verzeichnisse

Hierarchische Verzeichnisse:

- Erleichtern den Überblick über das Dateisystem
- Ein **hierarchischer Namensraum** wird erzeugt.
- Daten werden als Dateien in den „Blättern“ des Verzeichnisbaums abgelegt.
- Jede Datei kann über einen **Pfad** eindeutig im Verzeichnisbaum identifiziert werden.

Beispiel:



FH Bielefeld
University of
Applied Sciences

Dateisysteme: Pfadnamen

- Der Pfadname einer Datei oder eines Verzeichnisses wird aus der **Kette der Knotennamen** gebildet, die durch ein spezielles Zeichen getrennt sind.
  - ***Absolute*** Pfadnamen (voll qualifizierte Namen): Der gesamte Name von der Wurzel bis zur Datei wird angegeben.
  - ***Relative*** Pfadnamen: Der Pfadname wird ab dem aktuellen Verzeichnis (Arbeitsverzeichnis) angegeben.
- *Beispiele*:
  - Absoluter Pfadname, Unix: **/home/user1/Document/text.doc**
  - Absoluter Pfadname, Windows: **C:\Dokumente und Einstellungen\user1\text.doc**
- Die Verzeichniseinträge ‚.' und ‚..' bezeichnen das aktuelle Verzeichnis und das Vorgängerverzeichnis.

FH Bielefeld
University of
Applied Sciences

Dateisysteme: Links unter Unix

- Zusätzlich zu dem hierarchischen Baum können unter Unix Querverweise (*links*) erzeugt werden.
- **Direkter** Querverweis (*hard link, physical link*):
  - Es wird ein **Eintrag im Verzeichnis** erstellt, der auf die ursprüngliche Datei zeigt.
  - Die Datei enthält ein **Zählattribut**, dass anzeigt, wie viele Verweise auf sie zeigen.
  - Beim **Löschen** einer direkten Querverbindung wird der **Zähler herunter gezählt**.
  - Datei wird erst dann gelöscht, wenn kein Link mehr existiert.
  - Direkte Querverbindung und Datei müssen auf dem **selben Datenträger** liegen!
- **Logischer** Querverweis (*symbolic link*):
  - Es wird eine neue Datei vom Typ „Link“ erstellt, die den Pfadnamen der ursprünglichen Datei enthält.
  - Link und Datei dürfen auf unterschiedlichen Datenträgern liegen.

FH Bielefeld
University of
Applied Sciences

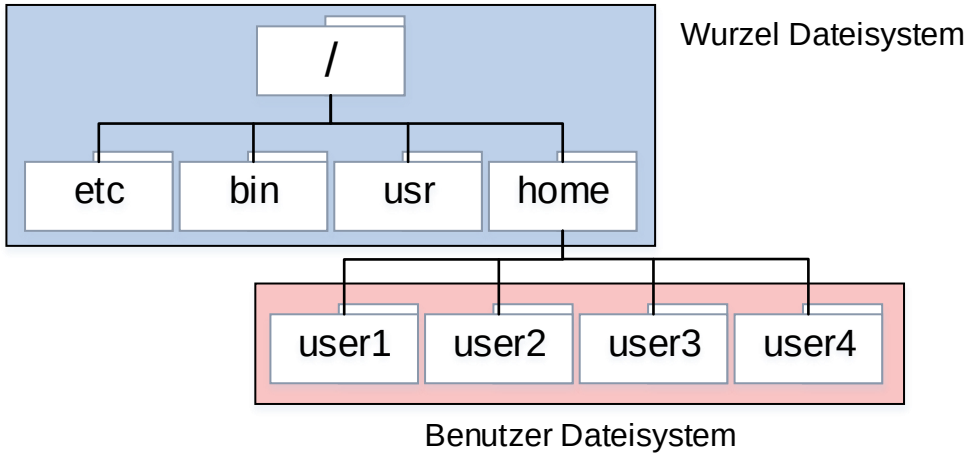
Dateisysteme: Links unter Unix (Forts.)

```
martin@redstar: ~/test
martin@redstar:~/test$ echo bla > lala.txt
martin@redstar:~/test$ stat lala.txt
  File: `lala.txt'
  Size: 4               Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 3210        Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  martin)   Gid: ( 1000/  martin)
Access: 2012-12-11 14:48:23.470332005 +0100
Modify: 2012-12-16 16:46:07.516933174 +0100
Change: 2012-12-16 16:46:07.516933174 +0100
 Birth: -
martin@redstar:~/test$ ln -P lala.txt lalalink
martin@redstar:~/test$ stat lala.txt
  File: `lala.txt'
  Size: 4               Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 3210        Links: 2
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  martin)   Gid: ( 1000/  martin)
Access: 2012-12-11 14:48:23.470332005 +0100
Modify: 2012-12-16 16:46:07.516933174 +0100
Change: 2012-12-16 16:46:28.736932509 +0100
 Birth: -
martin@redstar:~/test$
```

FH Bielefeld
University of
Applied Sciences

Dateisysteme: Erweiterung

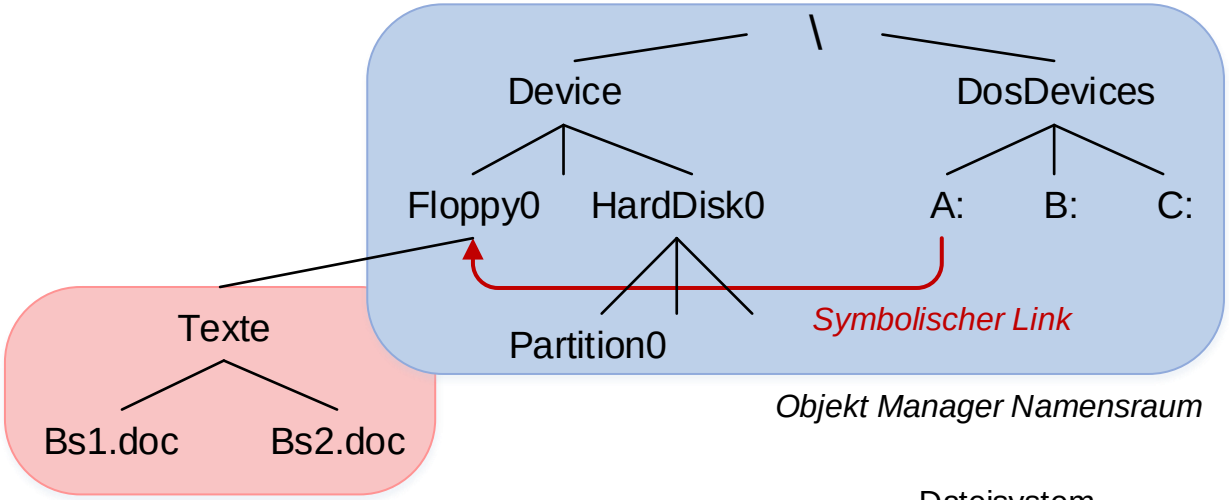
- Unix Dateisystem ist **hierarchisch** aufgebaut:
  - Zerfällt meistens in verschiedene Teilsysteme, die sich auf verschiedenen Laufwerken befinden können.
- Über den Befehl **mount** können weitere Geräte (Laufwerke) eingebunden werden.
- Beispiel: Benutzer-Verzeichnisse liegen auf einer zweiten Festplatte.



FH Bielefeld
University of
Applied Sciences

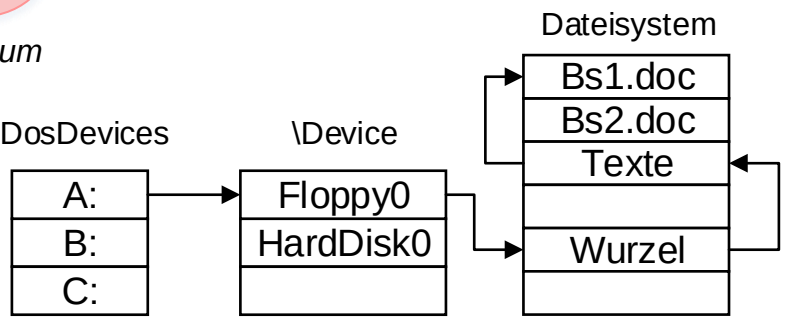
Dateisysteme: Windows Namensraum

- Windows besitzt einen globalen Namensraum, der alle Objekte (z.B. Dateien, Kommunikationskanäle...) enthält.
- Im Namensraum sind logische Querverbindungen möglich.



Beispiel: Öffne

A:\Texte\Bs1.doc



FH Bielefeld
University of
Applied Sciences

Dateisysteme: Realisierung

Generelle Umsetzung: Datei wird **fragmentiert**.

Gründe:

- **Naiver Ansatz**:
  - Datei wird am Stück (in aufeinanderfolgenden Sektoren) gespeichert.
  - Problem: Was passiert, **wenn die Datei wächst**?
- **Fragmentierte** Speicherung:
  - Datei wird in **Speichereinheiten** zerlegt.
  - Speichereinheiten werden über den Datenträger verteilt gespeichert.
  - Information zu belegten Speichereinheiten muss **verwaltet** werden.
  - Verbreitete Lösungen: FAT, UNIX i-Nodes

FH Bielefeld
University of
Applied Sciences

Themenübersicht

- Einleitung: Dateisysteme
  - Datenorganisation: Dateien, Verzeichnisse
  - Pfade, Verweise (Links)
- Beispiele für Dateisysteme:
- **File Allocation Table (FAT)**
- Unix File System (UFS)
- NT File System (NTFS)

FH Bielefeld
University of
Applied Sciences

FAT: Eigenschaften

FAT (*File Allocation Table, FAT*):

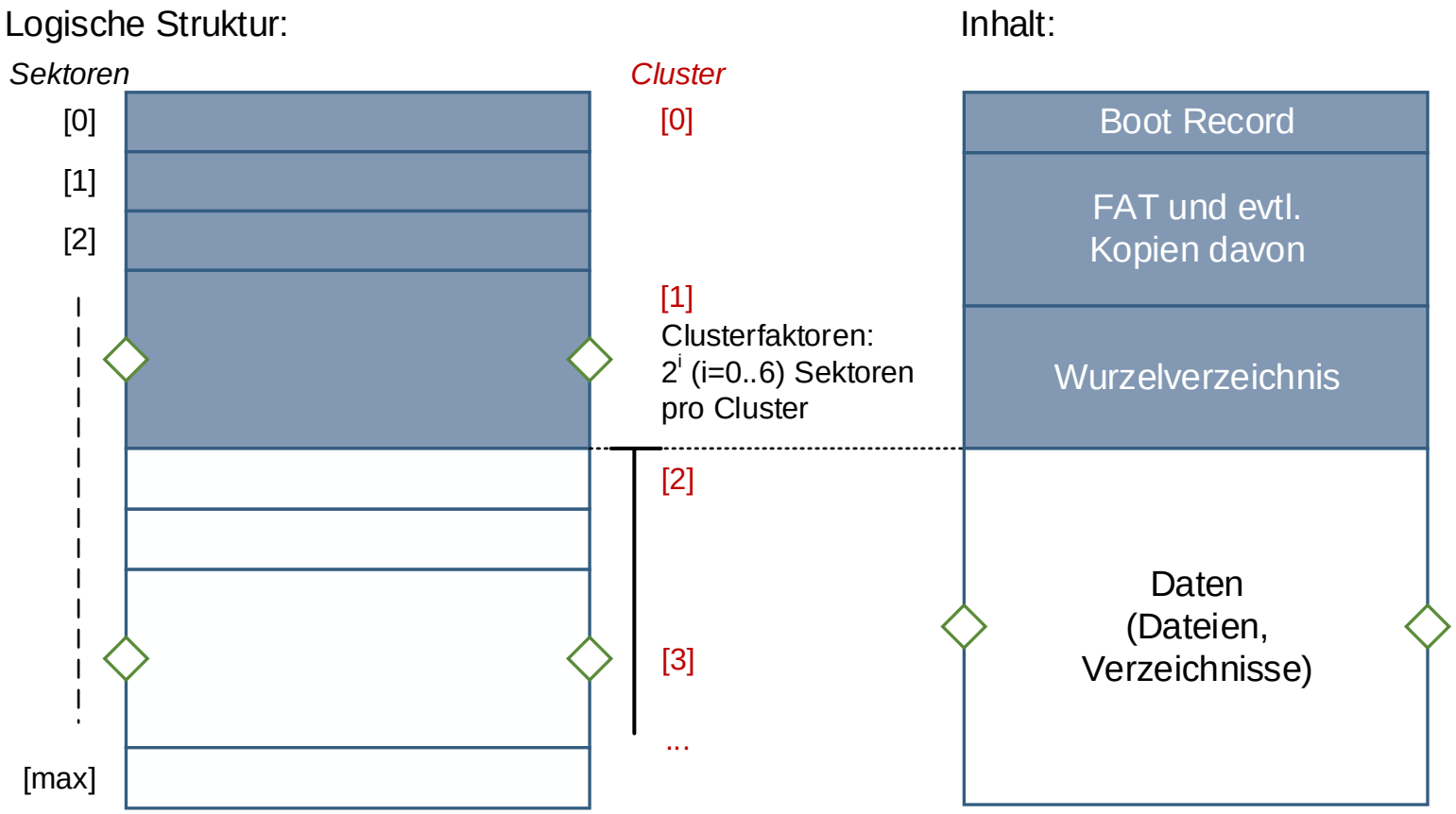
- Informationen über den belegten Speicher werden in einer **Tabelle** abgelegt.
- Traditionelles Dateisystem unter **Microsoft** DOS und den ersten Windows Versionen (entwickelt ~1980).
- Aktuelle Bedeutung: Flash-Speicherkarten, Wechseldatenträger

Wichtige Eigenschaften:

- Beschränkung auf **8 Zeichen** für den **Dateinamen** und **3 Zeichen** für die **Dateierweiterung**.
- Je nach Größe des Speichermediums gibt es **FAT-12** (max. 16 MiB Kapazität), **FAT-16** (max. 2 GiB Kapazität) oder eine **FAT-32** (max. 2 TiB Kapazität) Version. (512B Sektorgröße, Clustergröße=64 Sektoren)
- Es gibt **keine Zugriffsregelung** über eine **Benutzerkennung**.

FH Bielefeld
University of
Applied Sciences

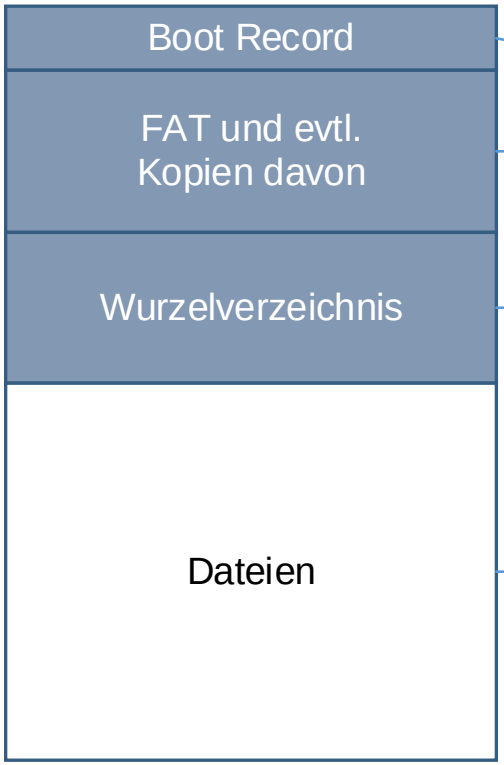
FAT: Organisation Festplatte



Clustergröße = Clusterfaktor * Sektorengröße

FH Bielefeld
University of
Applied Sciences

FAT: Organisation Festplatte (Forts.)



Boot Record: Enthält Grunddaten über die Platte und Routine für Betriebssystem-Ladeprozess.

FAT: Enthält Informationen über freie und belegte Cluster (ein Eintrag pro Cluster). Da essentiell für das System, mit **Kopie**. Sollte im **Hauptspeicher** gehalten werden.

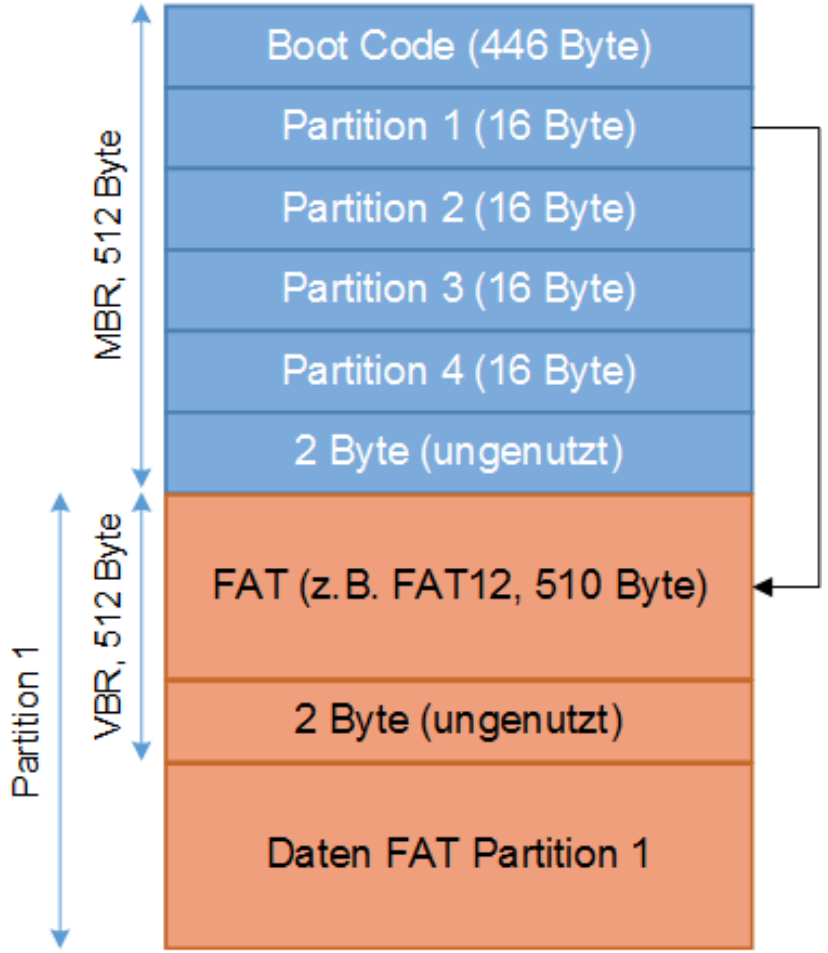
Wurzelverzeichnis: Enthält Liste aller Dateien und Verzeichnisse im Wurzelverzeichnis (außer bei FAT-32).

Dateien: Enthält Anwenderdateien sowie alle tiefer liegenden Datei- und Verzeichniseinträge

FH Bielefeld
University of
Applied Sciences

FAT: Partitionen & Boot Records

- Manchmal ist das Speichermedium partitioniert:
  - Festplatte → *immer*
  - USB-Stick, SD-Karte → *nie*
- Dann sind dem Dateisystem noch Meta-Daten vorangestellt:
  - *Master Boot Record*
  - *Volume Boot Record*



FH Bielefeld
University of
Applied Sciences

FAT: Genereller Aufbau

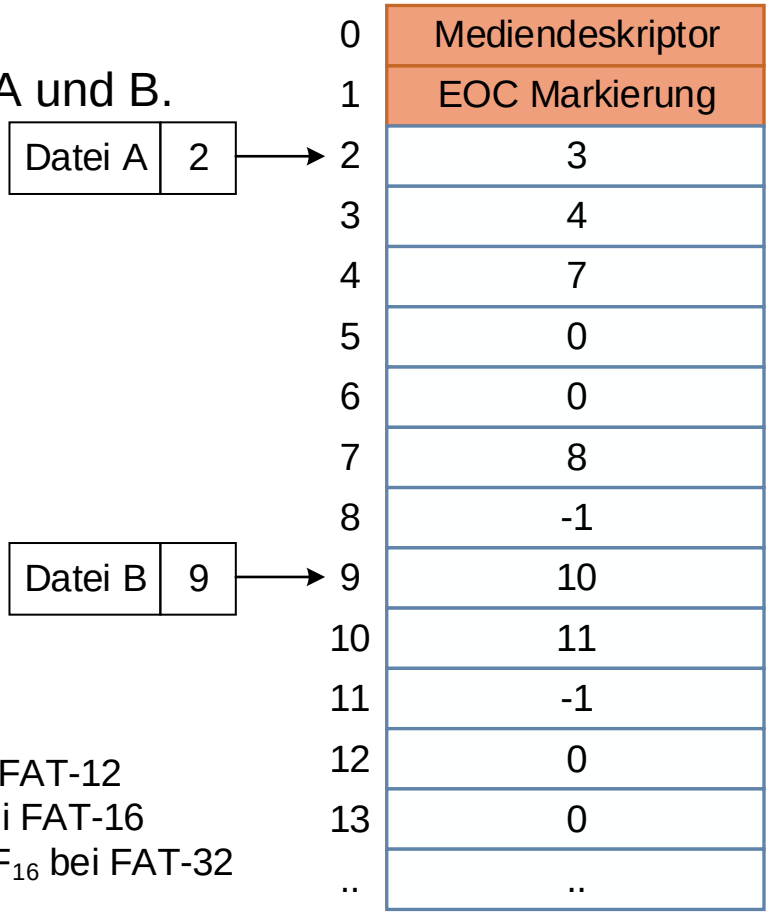
- *File Allocation Table* (FAT) ist ein Abbild der **Clusterbelegung**:
  - Für jeden Cluster existiert ein Eintrag.
  - Einträge verkettet zu einer Liste, der durch die Datei belegten Cluster.
- Länge eines Eintrags:
  - **12 Bit** (für FAT-12) -> 4095 Cluster,
  - **16 Bit** (für FAT-16) -> 65535 Cluster oder
  - **32 Bit** (28 genutzte Bits, für FAT-32) -> 270 Mio. Cluster breit.
- Für einige **Verwaltungsinformation** (z.B. Formatierungsfehler eines Clusters) stehen definierte Einträge zur Verfügung, so dass sich die Zahl der adressierbaren **Cluster um 10 verringert**.
- **Eintrag 0** enthält einen **Mediendeskriptor**, **Eintrag 1** eine **EOC** (*end of cluster chain*) Markierung.
- Damit ist **Cluster 2** der **erste nutzbare** Cluster in der FAT.

FH Bielefeld
University of
Applied Sciences

FAT: Einträge

Beispiel: Es existieren zwei Dateien, A und B.

- A belegt die Cluster 2, 3, 4, 7, 8
- B belegt die Cluster 9, 10, 11

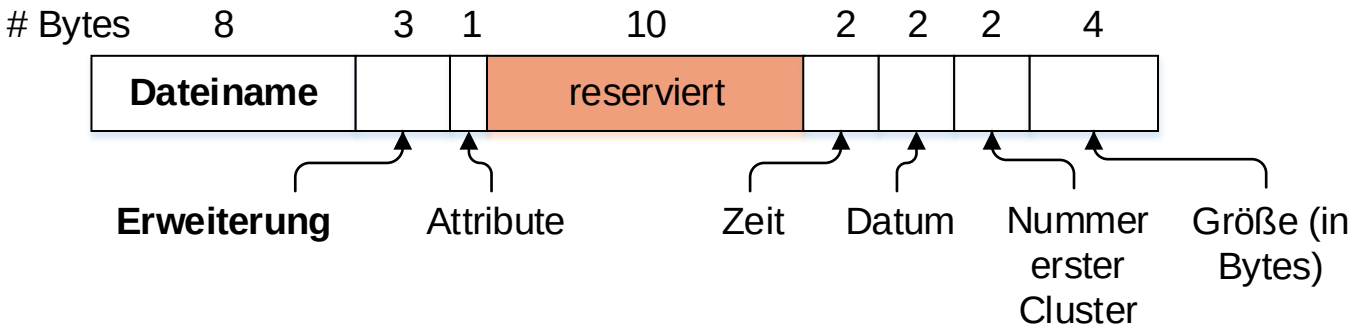


-1: FFF_{16} bei FAT-12
$FFFF_{16}$ bei FAT-16
$0FFFFFFF_{16}$ bei FAT-32

FH Bielefeld
University of
Applied Sciences

FAT: Verzeichniseinträge

Prinzipieller Aufbau der **Verzeichniseinträge**:

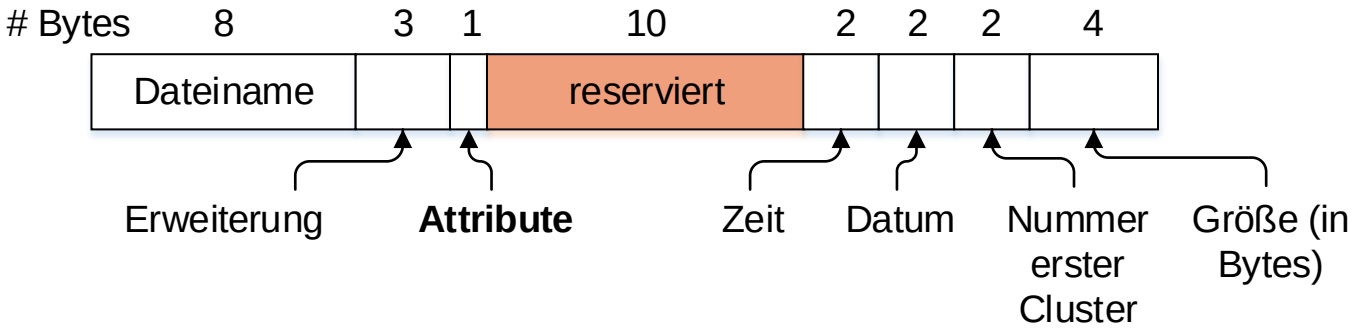


Bedeutung:

- **Dateiname**, **Erweiterung**: 8 Byte Name der Datei + 3 Byte Erweiterung. Wird mit Leerzeichen aufgefüllt.
  - Erstes Byte 0: Der Eintrag ist **frei**, war noch nie benutzt.
  - Erstes Byte 0xE5: Der Eintrag wurde **gelöscht** (kann für *undelete* benutzt werden.)
  - Erstes Byte 0x2E („.“), zweites Byte Leerzeichen („ “): Der Eintrag ist der erste Eintrag eines **Unterverzeichnisses**.
  - Erste beiden Bytes 0x2E („..“): Der Eintrag verweist auf das **übergeordnete** Verzeichnis.

FH Bielefeld
University of
Applied Sciences

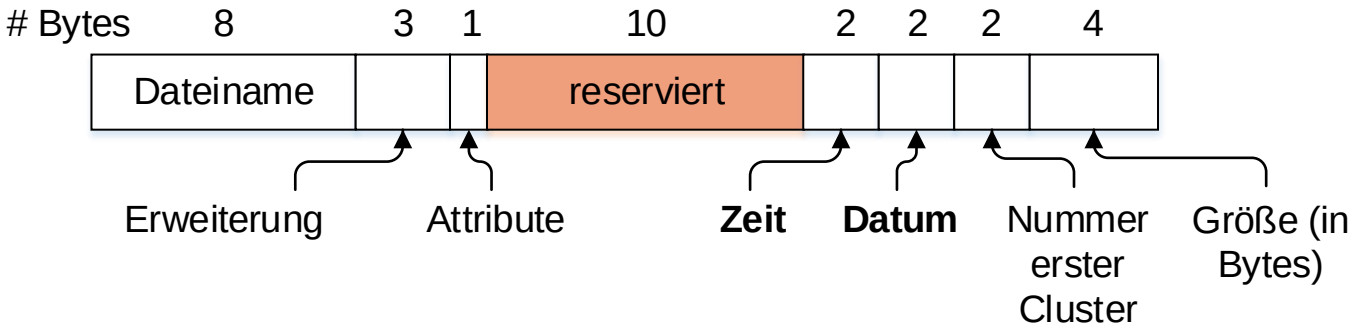
FAT: Verzeichniseinträge (Forts.)



- **Attribute**:
  - Bit 0: Read-only Flag
  - Bit 1: Hidden Flag
  - Bit 2: System Flag
  - Bit 3: Volume Label (Der Eintrag enthält nur den Namen des Mediums, sollte nur in einem (einzigen) Wurzelverzeichniseintrag gesetzt sein!)
  - Bit 4: Unterverzeichnis Flag. Die Verzeichniseinträge sind im Cluster „Erste Clusternummer“ abgelegt.
  - Bit 5: Archiv Flag
  - Bit 6+7: Nicht benutzt.

FH Bielefeld
University of
Applied Sciences

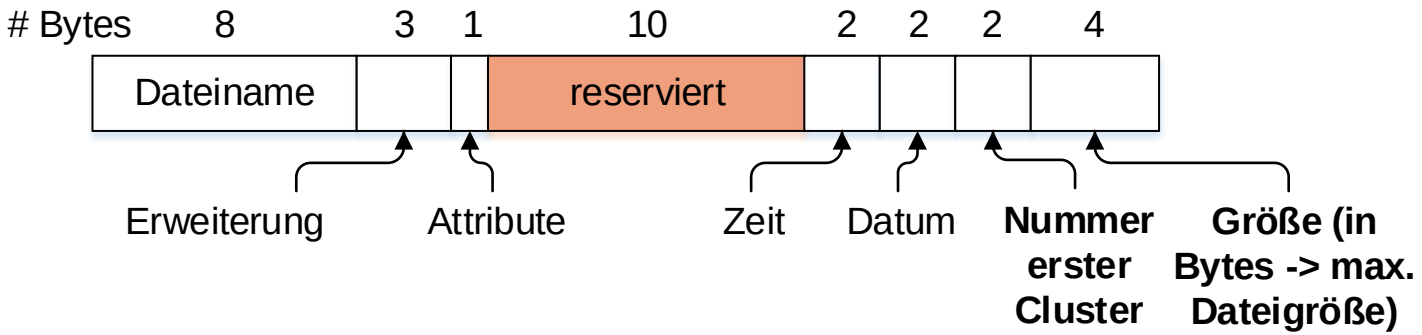
FAT: Verzeichniseinträge (Forts.)



- **Zeit**: Zeit der Erstellung oder letzten Änderung (zuerst LSB).
  - 5 Bit für die Stunden (0..23)
  - 6 Bit für die Minuten (0..59)
  - 5 Bit für die Sekunden (0..29), in 2 Sekunden Schritten
- **Datum**: Datum der Erstellung oder letzten Änderung
  - 7 Bit für das Jahr (0..127) ab 1980
  - 4 Bit für den Monat (1..12)
  - 5 Bit für den Tag (1..31)

FH Bielefeld
University of
Applied Sciences

FAT: Verzeichniseinträge (Forts.)



- **Erste Clusternummer**:
  - Bei einer **Datei**: **Erster Cluster** mit den Daten der Datei, die anderen Cluster können aus der FAT geholt werden.
  - Bei einem **Verzeichnis**: **Cluster**, in dem die **Verzeichniseinträge** liegen.
  - Bei FAT32 liegen die MSB im reservierten Bereich (Bytes 20 und 21).
- **Größe** in Bytes: Dateigröße.

Hinweis: 16 und 32 Bit Werte sind im *little endian mode* abgelegt, d.h. zuerst das LSB, dann das MSB!

FAT: Cluster- und Partitionsgrößen

- Platte kann mehrere FAT Partitionen beinhalten
- Maximale Größe einer FAT Partition ist vom Clusterfaktor abhängig.
- Partitionsgröße in Abhängigkeit von der Clustergröße (Tanenbaum):

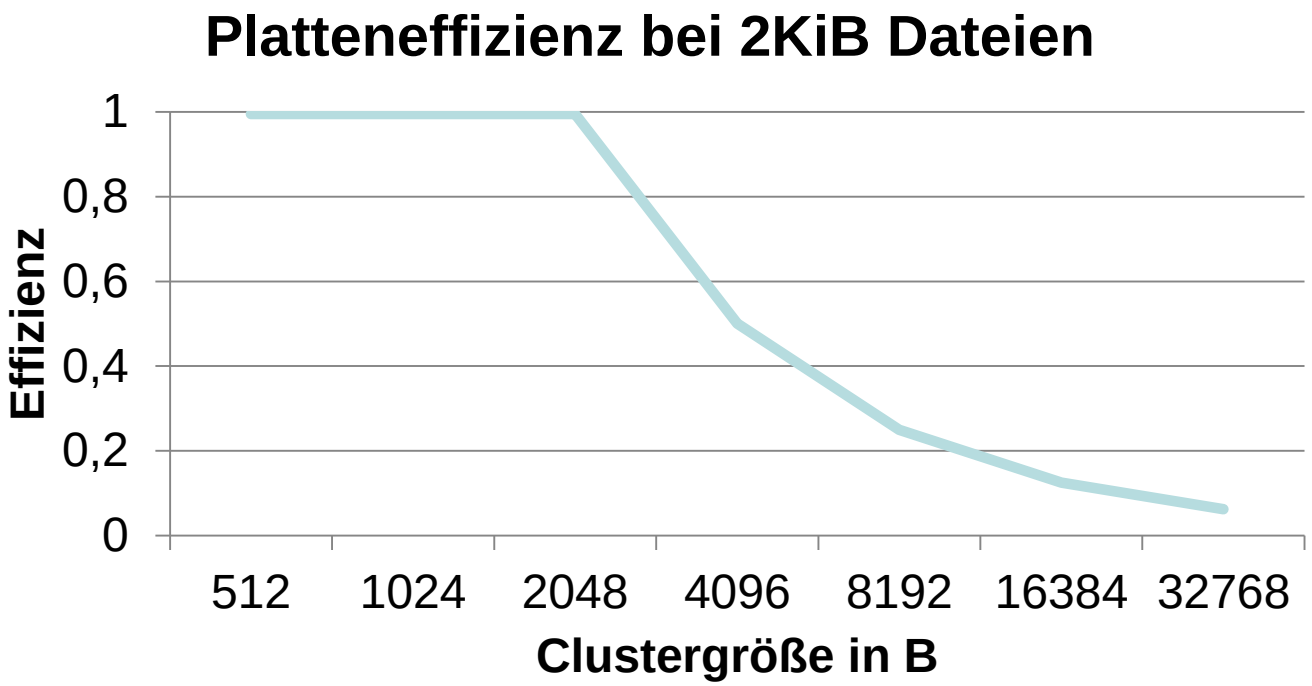
Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Figure 4-32. Maximum partition size for different block sizes. The empty boxes represent forbidden combinations.

FH Bielefeld
University of
Applied Sciences

FAT: Platteneffizienz

- Die Platteneffizienz ist Abhängig von der mittleren Dateigröße.
- Bei vielen kleinen Dateien kann eine kleine Clustergröße vorteilhaft sein!



FH Bielefeld
University of
Applied Sciences

FAT: Weiterentwicklungen

- VFAT: Dateinamen mit bis zu 255 Zeichen
  - Mehrere Verzeichniseinträge nehmen den Dateinamen auf
- exFAT:
  - Weiterentwicklung speziell für Wechseldatenträger (USB Sticks, SD-Karten)
  - Erstmals unter Windows XP (2006) unterstützt
  - Patentiert, closed source
  - Standard für SDXC Speicherkarten mit mehr als 32GB

FH Bielefeld
University of
Applied Sciences

Themenübersicht

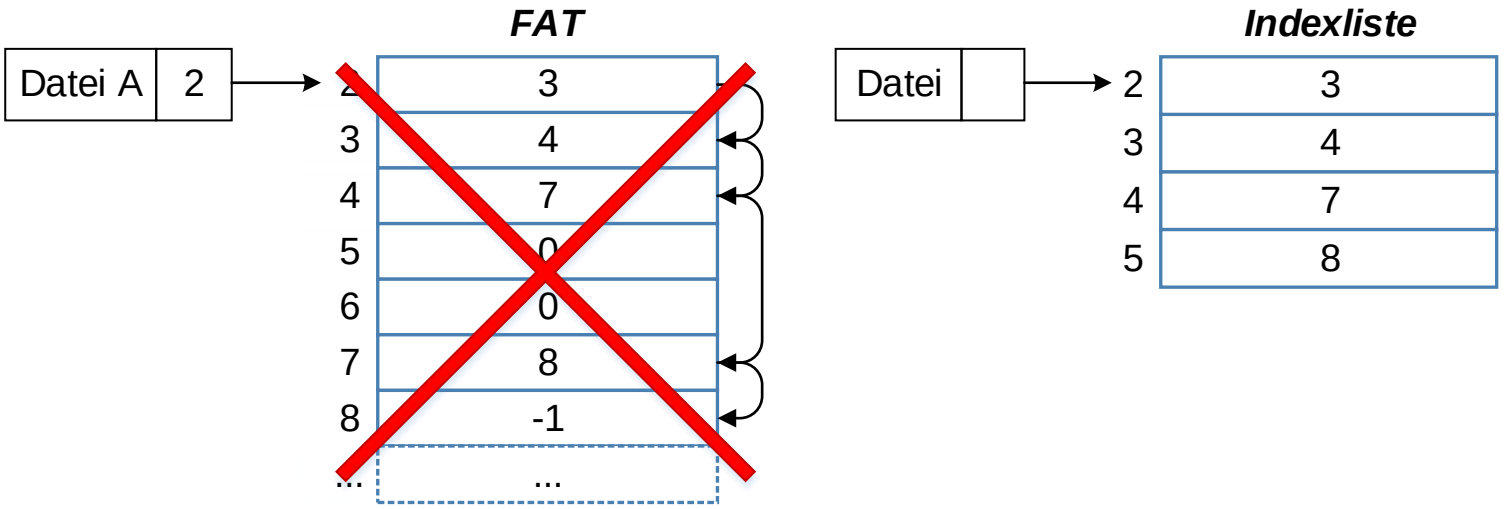
- Einleitung: Dateisysteme
  - Datenorganisation: Dateien, Verzeichnisse
  - Pfade, Verweise (Links)
- Beispiele für Dateisysteme:
  - File Allocation Table (FAT)
  - **Unix File System (UFS)**
  - NT File System (NTFS)

FH Bielefeld
University of
Applied Sciences

UFS: Indexbezogene Verwaltung

Idee der **Index-Knoten** (*I-Nodes* oder *Inode*):

- Jede Datei besitzt ihre eigene Indexliste über ihre belegten Blöcke.

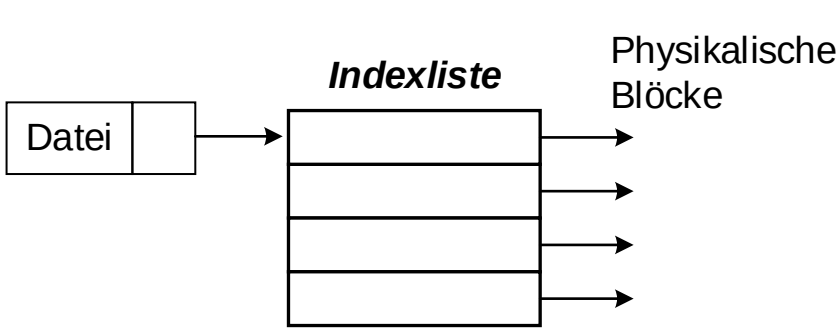


- Die Indexliste muss nur dann gelesen werden, wenn eine Datei geöffnet wird.
- Ein **I-Node** enthält alle Metainformationen über die Datei sowie die Indexliste.
- Es existiert eine gemeinsame Liste mit freien Blöcken.

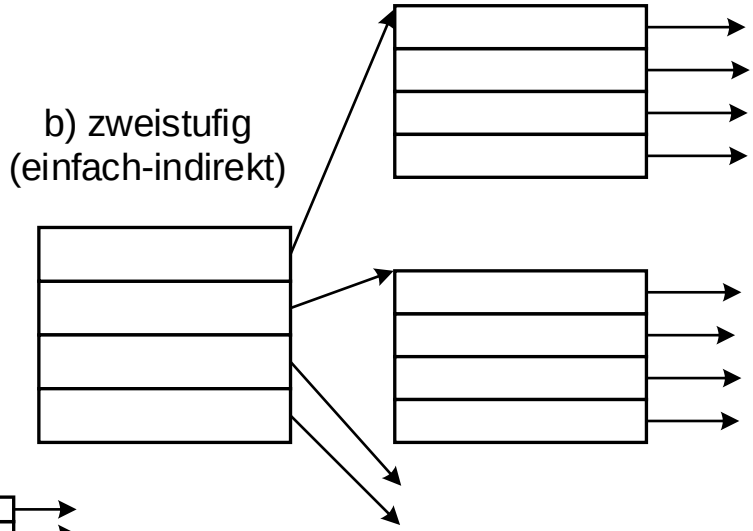
FH Bielefeld
University of
Applied Sciences

UFS: Mehrstufige Übersetzung

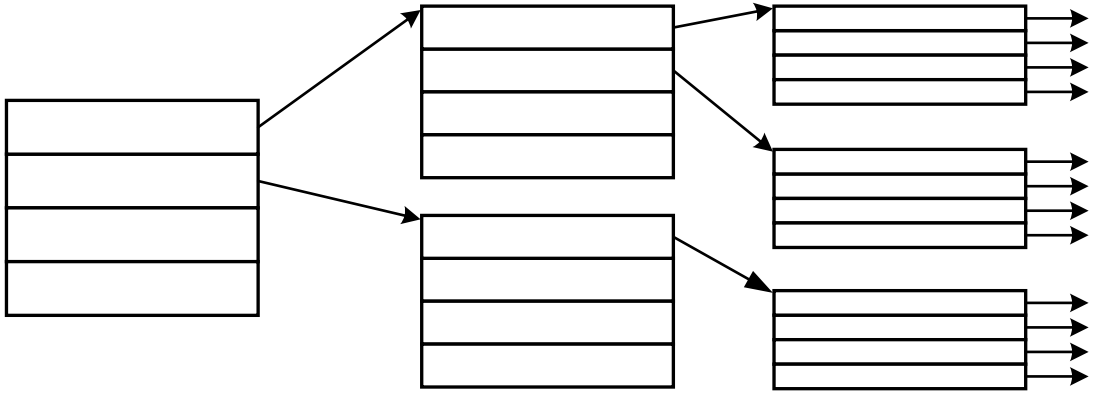
Ein I-Node kann nur eine feste Anzahl von Festplattenblöcken referenzieren



a) einstufig



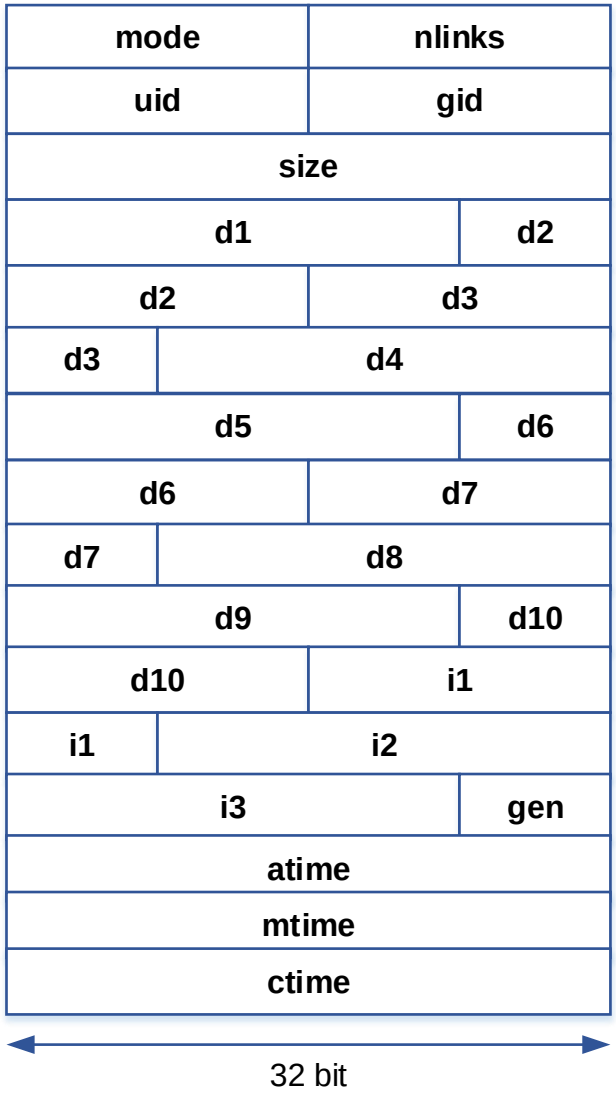
b) zweistufig
(einfach-indirekt)



c) dreistufig (zweifach-indirekt)

UFS: I-Node in Unix, System V

In Unix System V ist ein I-Node 64 Byte groß:



- **mode:** Dateityp, Schutzbits, setuid, setgid Bits.
- **nlinks:** Anzahl der Verzeichniseinträge auf diesen I-Node.
- **uid:** UID des Besitzers
- **gid:** GID des Besitzers
- **size:** Dateigröße in Byte (32bit -> max. Dateigröße 4 GiByte).
- **d1-d10:** 10 direkte Blockadressen.
- **i1-i3:** eine einfache, eine zweifache und eine dreifache Indirektion.
- **gen:** wird bei Zugriff inkrementiert.
- **atime,mtime,ctime:** Zeit des letzten Zugriffs, der letzten Änderung und der Letzten Änderung des I-Nodes.

FH Bielefeld
University of
Applied Sciences

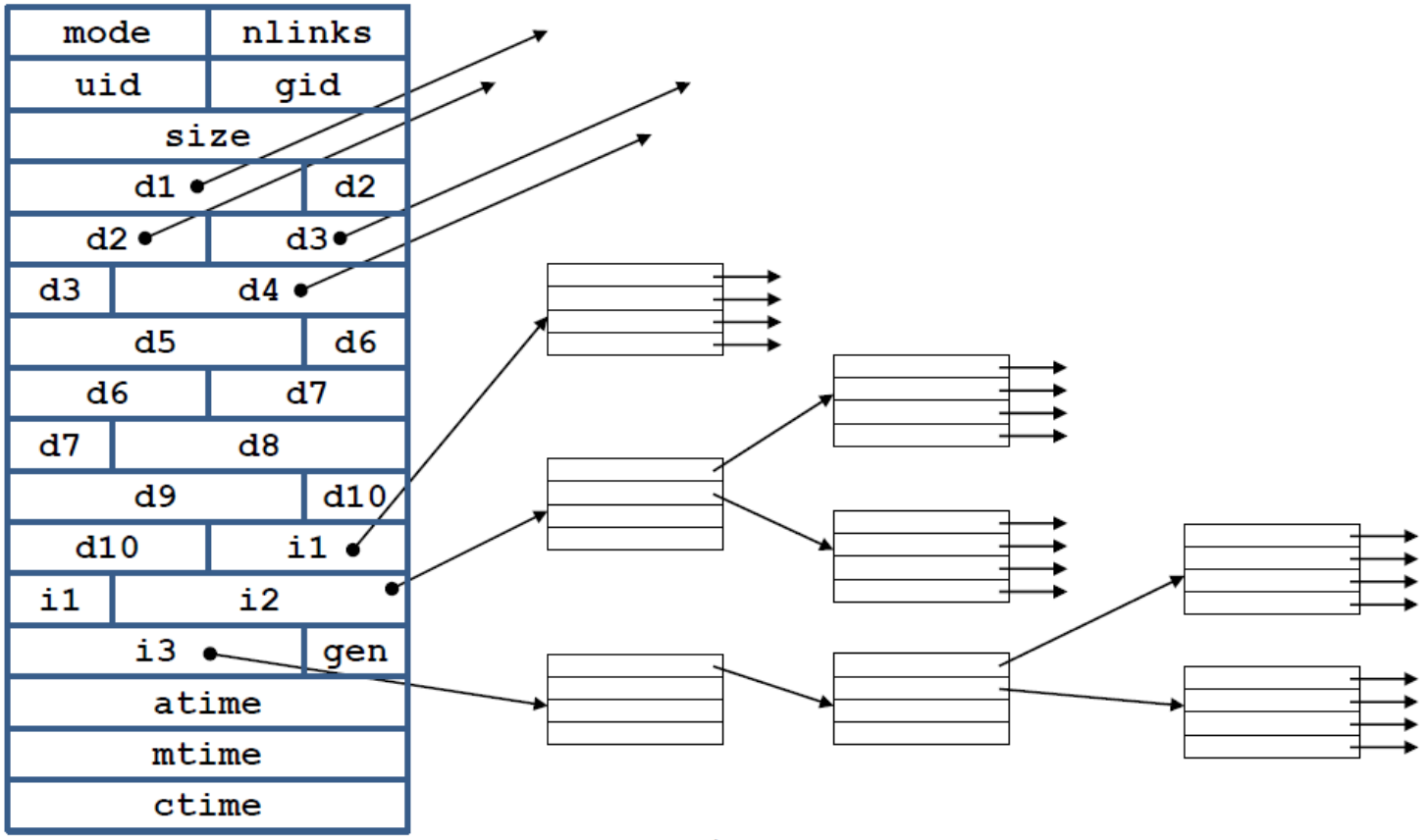
UFS: Beispiel für I-Node Inhalte

```
martin@redstar: ~/test
martin@redstar:~/test$ echo kurt > lala.txt
martin@redstar:~/test$ stat lala.txt
  File: `lala.txt'
  Size: 5               Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 3210        Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  martin)   Gid: ( 1000/  martin)
Access: 2012-12-11 14:48:23.470332005 +0100
Modify: 2012-12-11 14:48:53.490333249 +0100
Change: 2012-12-11 14:48:53.490333249 +0100
 Birth: -
```

FH Bielefeld
University of
Applied Sciences

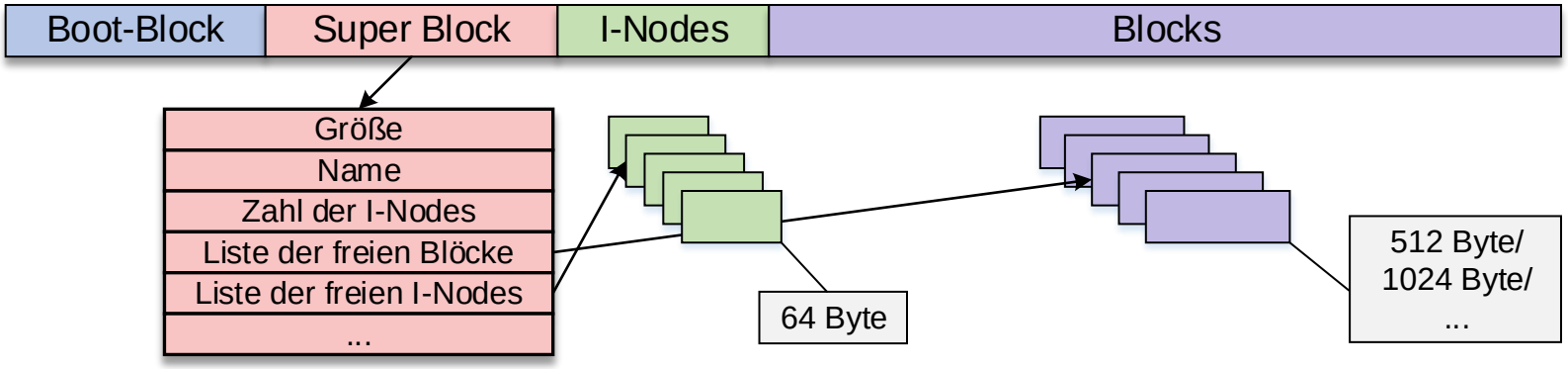
UFS: I-Node Indirektionsblöcke

Ein Indirektionsblock enthält (Blockgröße/Adresslänge) Adressen auf andere Blöcke.



FH Bielefeld
University of
Applied Sciences

UFS: Organisation Partition

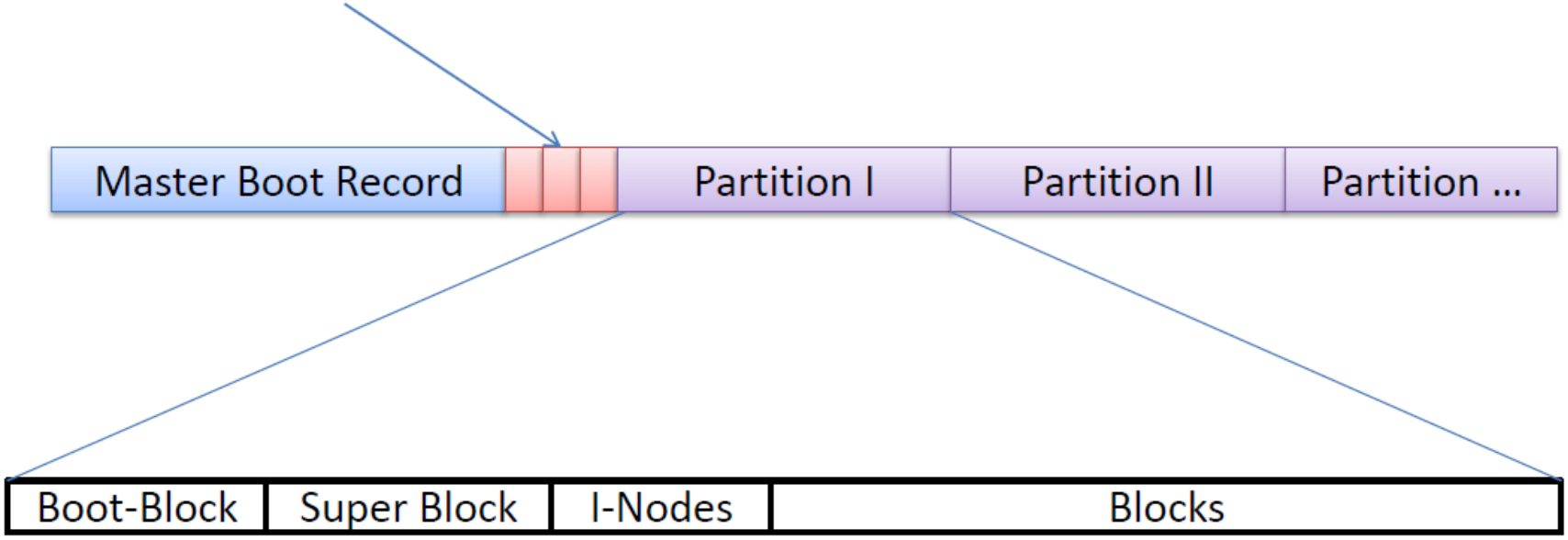


- **Boot-Block** (Block 0): Enthält Daten zum Laden des Betriebssystems
- **Super-Block**: Enthält Basisinformationen über den Datenträger. *Ein zerstörter Super-Block macht das Dateisystem unlesbar!*
- **I-Nodes**: Liste der I-Nodes im Dateisystem. Freie I-Nodes werden mit einer 0 im *mode* Feld gekennzeichnet.
- **Datenblöcke** (*Blocks*): Enthalten Verzeichnisdaten, Indirektionsblöcke und Nutzdaten. Nutzdaten sind die von den Applikationsprogrammen persistent zu speichernden Daten.

FH Bielefeld
University of
Applied Sciences

UFS: Organisation Platte

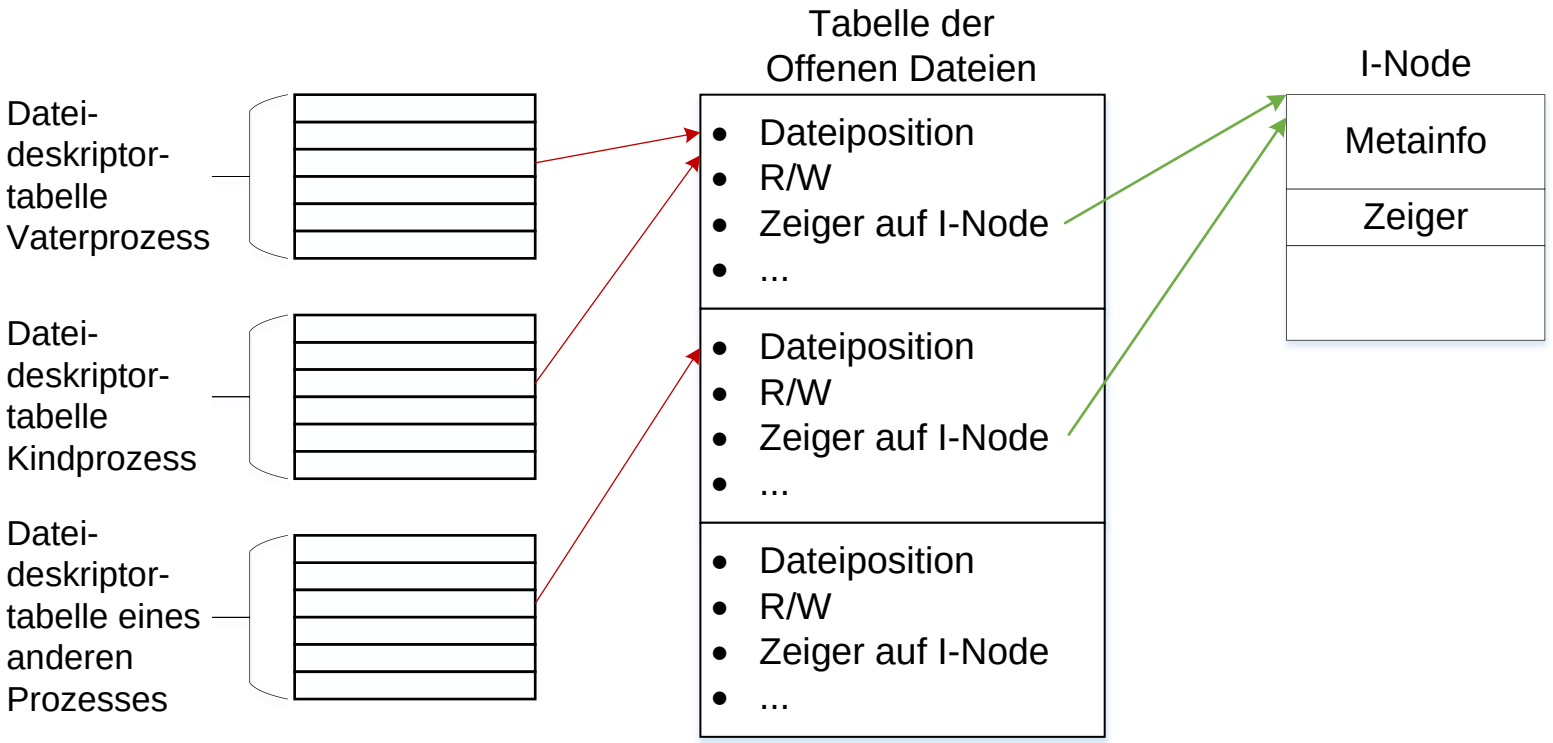
- **Master Boot Record** (MBR) lädt Inhalt des Boot-Block Codes der aktiven Partition
- **Tabelle** verwaltet Liste der Partitionen



FH Bielefeld
University of
Applied Sciences

UFS: Dateilokalisation

- I-Nodes von geöffneten Dateien werden im Hauptspeicher abgelegt



FH Bielefeld
University of
Applied Sciences

UFS: Verzeichnisorganisation

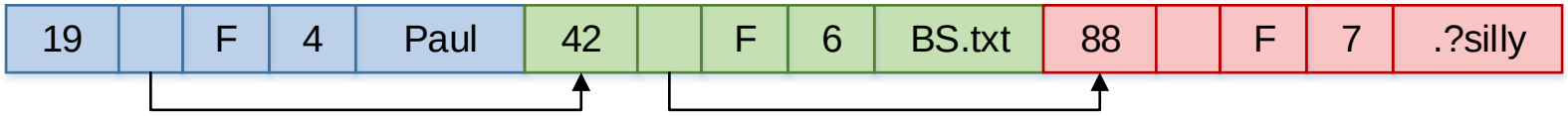
- Das Wurzelverzeichnis verwendet I-Node 2. (I-Node 0 und 1 sind reserviert. i.a. zeigt I-Node 1 auf die defekten Blöcke.)
  - Ursprünglicher Verzeichniseintrag in Unix, Dateinamen auf 14 Zeichen begrenzt:

Größe in Byte:	2	14
	I-Node-Nr.	Dateiname

Aktuelle Dateisysteme enthalten mehr Einträge, Dateinamen sind i.d.R. auf 255 Zeichen begrenzt:

Größe in Byte:	4	2	1	1	1..255
	I-Node-Nr.	Länge Eintrag	Typ	Länge Namen	Dateiname

Beispiel: Verzeichnis mit drei Dateien:

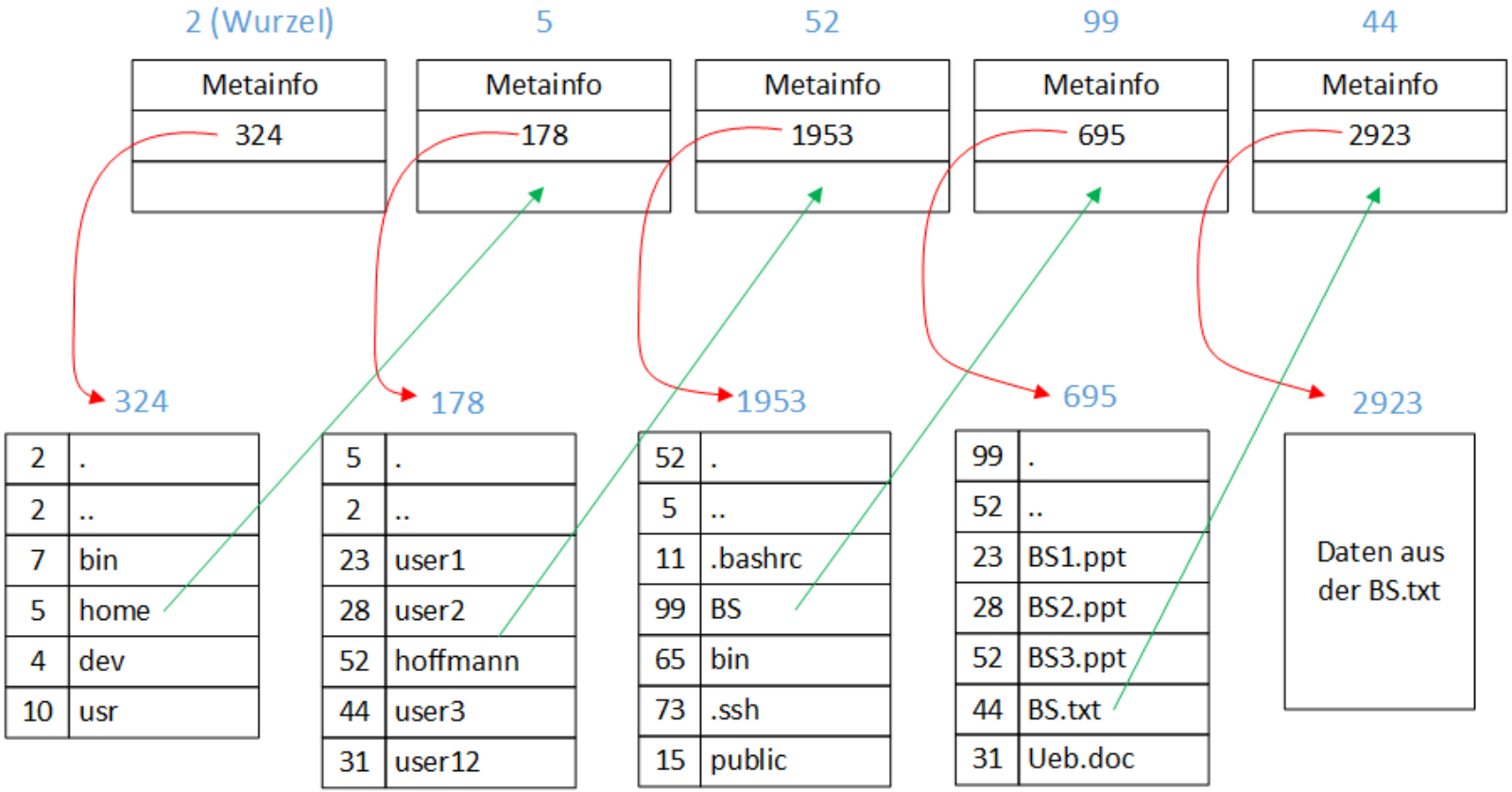


Löschen der Datei **BS.txt**:

FH Bielefeld
University of
Applied Sciences

UFS: Dateilokalisation

Beispiel: Lokalisierung einer Datei **/home/hoffmann/BS/BS.txt**



FH Bielefeld
University of
Applied Sciences

Vorlesung

Vielen Dank für Ihre Aufmerksamkeit

Dozent

Prof. Dr.-Ing.
Martin Hoffmann
martin.hoffmann@fh-bielefeld.de