Dateisysteme

Betriebssysteme I
WS 2005/2006

Jörg Kaiser
IVS – EOS

Otto-von-Guericke-Universität Magdeburg

Dateisysteme: Motivation

Wozu wird einen zusätzliche Art von Speicher benötigt?

- **Persistenz ?**
- **Gemeinsame Nutzung ?**
- **Zugriffsschutz ?**
- **Größe ?**

Dateisysteme: Die systemorientierte Sicht

Dateien als allgemeine Abstraktion für langlebige Einheiten:

- **Benutzerdokumente: reguläre Dateien**
- **Programme: ausführbare Dateien**
- **Strukturen zur Dateiorganisation: Verzeichnisse**
- **Abstraktionen von Speichergeräten: Block-Dateien**
- **Abstraktionen zur Modellierung v. Geräten: Spezial-Dateien**

Unterschiede werden im Dateityp festgehalten.

Dateinamen

Beispiele

name.extension	Bedeutung
name.txt	Textdatei
name.c	C Quelldatei
name.o	Objektdatei (Maschinencode nicht gelinked)
name.bak	Backup-Datei
name. jpg	Datei codiert im JPEG Standard
name.mp3	Datei codiert im MPEG 3 Standard
name.pdf	pdf Datei (portbale document format)

gif, tiff, as, ps, zip, tex, hlp, html, doc, exe, xls.....

Variationen: Characters: upper/lower case, unicode,. .

Erweiterungen: Konventionen vs. interpretiert durch das BS

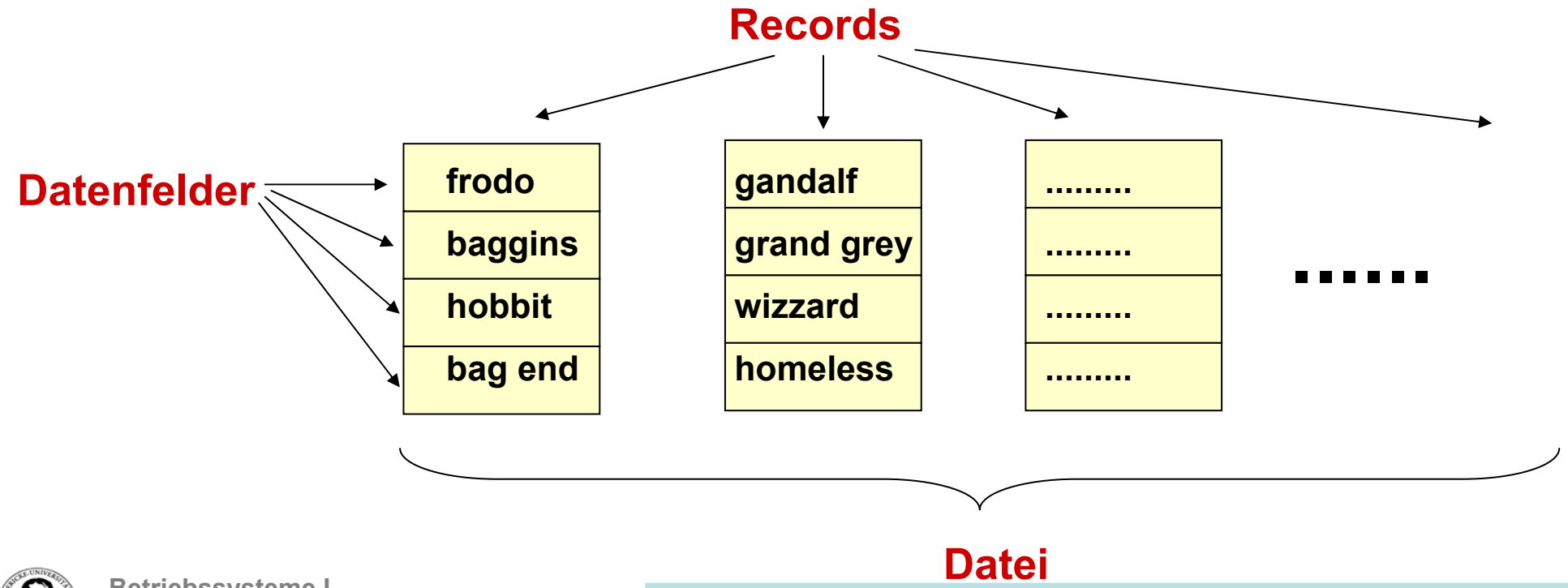
Informationsstruktur

Feld: grundlegendes Datenelement

Record: Menge zusammengehörender Felder

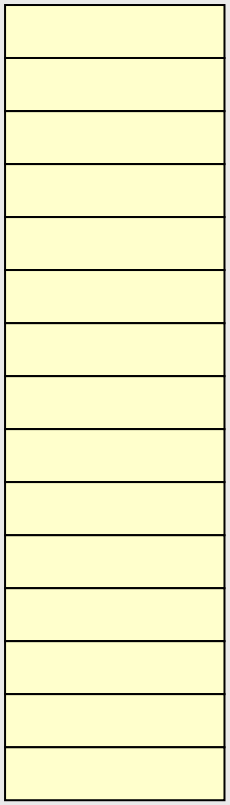
Datei: Menge zusammengehörender Records

Beispiel für die Informationsstruktur: <first name>, <family name>, <origin>, <home address>

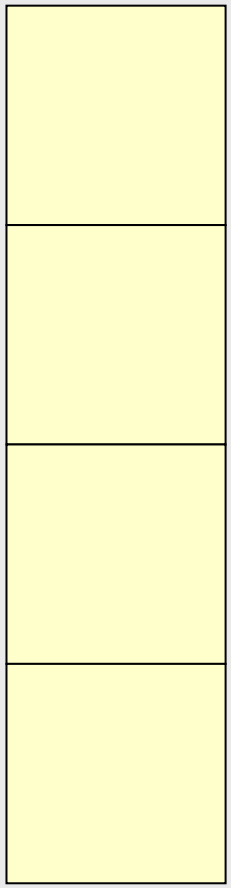


Dateiorganisation

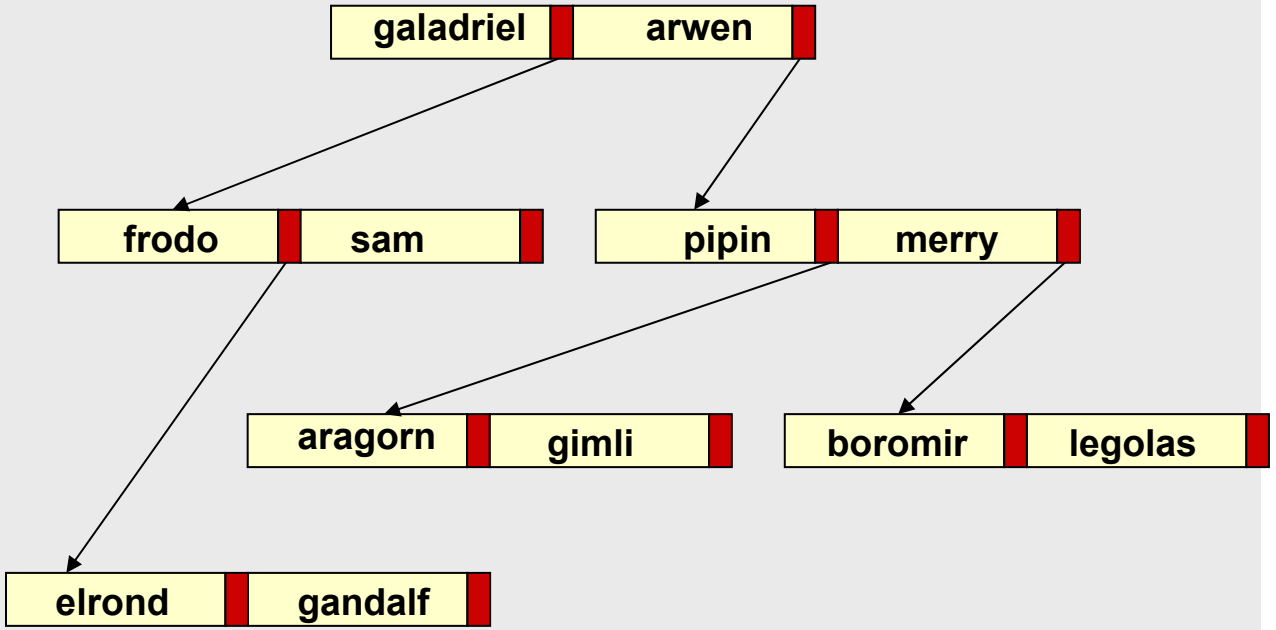
Folge von Bytes



Folge von Records



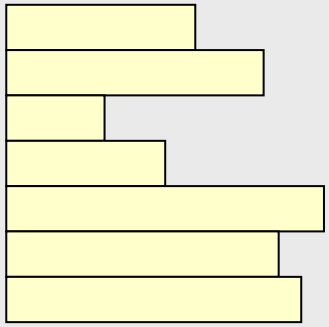
verkettete (Baum-) Struktur



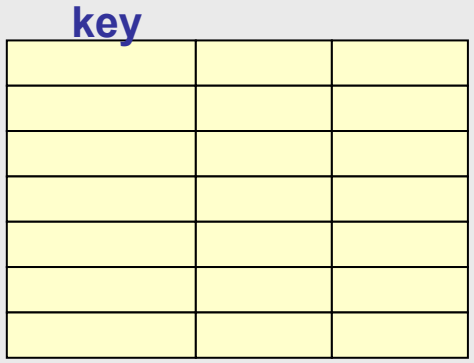
Dateiorganisation und Zugriff

Wie findet man einen Record? Alternativen in der Dateiorganisation:

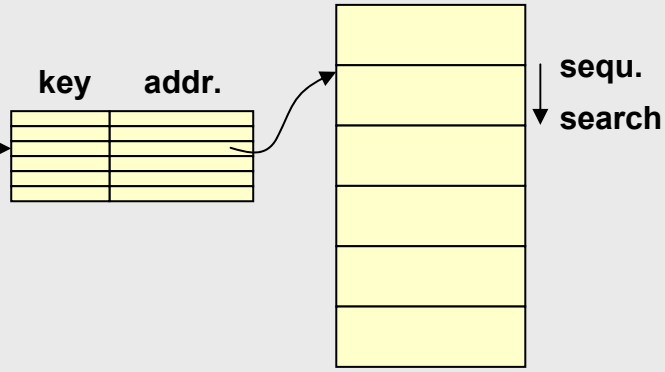
Sammlung (collection):
Records variabler Länge in chronologischer Reihenfolge.



Sequentiell:
Records fester Länge in sequentieller Ordnung gemäß eines Schlüsselfeldes



Index sequentiell:
Sequentielle Suche im entsprechenden Bereich



vollständig indiziert:
Index für mehrere Felder
Index für jeden Record
keine sequentielle Suche

Dateiattribute

Basisinformation: Dateiname, Dateityp, (Dateiorganisation)

Addressinformation: Gerät, phys. Startadresse, akt. Größe, max. Größe

Zugriffskontrollinfo.: Besitzer, Zugriffsautorisation, Zugriffsrechte

Dateiinformation: Erzeugungsdaten, Erzeuger ID, letzter Lesezugriff, ID des letzten Lesers, Datum der letzten Modifikation, ID des letzten Schreibers, Datum des Backup, aktuelle Benutzunginformation.

Dateiattribute

Beispiele von Dateiattributen

acces control	who is allowed to do what with the file
passwd	passwd for the file access
creator	ID of the file creator
owner	current owner
read-only-flag	0: R/W, 1:read only
hidden flag	0: default, 1: invisible
system flag	0: normal file, 1: system file
archive flag	0: changes saved, 1: not yet saved
ASCII/binary flag	0: ASCII file, 1: binary file
random access flag	0: sequential file, 1: random access
temporary flag	0: normal, 1: delete file on process termination
lock flags	0: not locked, ≠ 0: file locked
record length	number of bytes in a record
key position	offset to the key in the record
key length	number of bytes in the key
time of last access	date and time of last access to this file
time of last modification	data and time of last change
actual (max) size	number of (max) bytes in file

Zugriffsoperationen für Dateien

Vorbeitung zum Dateizugriff	create	delete
	open	close
normaler Dateizugriff	read	write
	append	
Suchen & Verwalten	seek	
	set attributes	get attributes
	rename	

"Memory Mapped" Dateien

Idee: Abbildung von Dateien in den virtuellen Speicher. Ausnutzung des Seiten-Mechanismus, um Dateien zwischen Platte und Hauptspeicher ein- und auszulagern.

Vorteil: Zugriff auf eine Datei kann über normale Lese- und Schreib-operationen auf den Speicher realisiert werden.

Systemaufrufe:

map (virtual address): bildet die Datein in den virtuellen Adressraum ab. Startadresse: virtual address

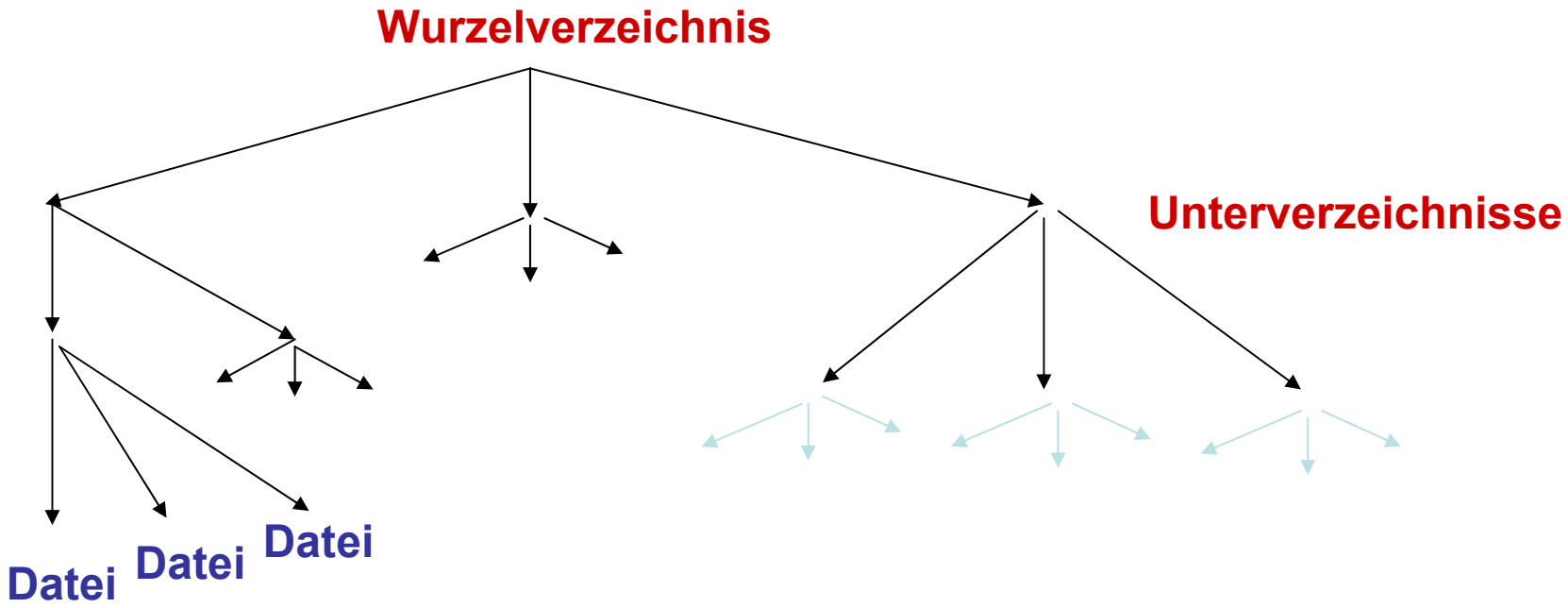
unmap: entferne Datei aus dem virtuellen Adressraum.

Probleme: exakte Größe der Ausgabedatei, gemeinsame Nutzung von "Memory Mapped" Dateien, Datei ist größer als der virtuelle Adressraum.

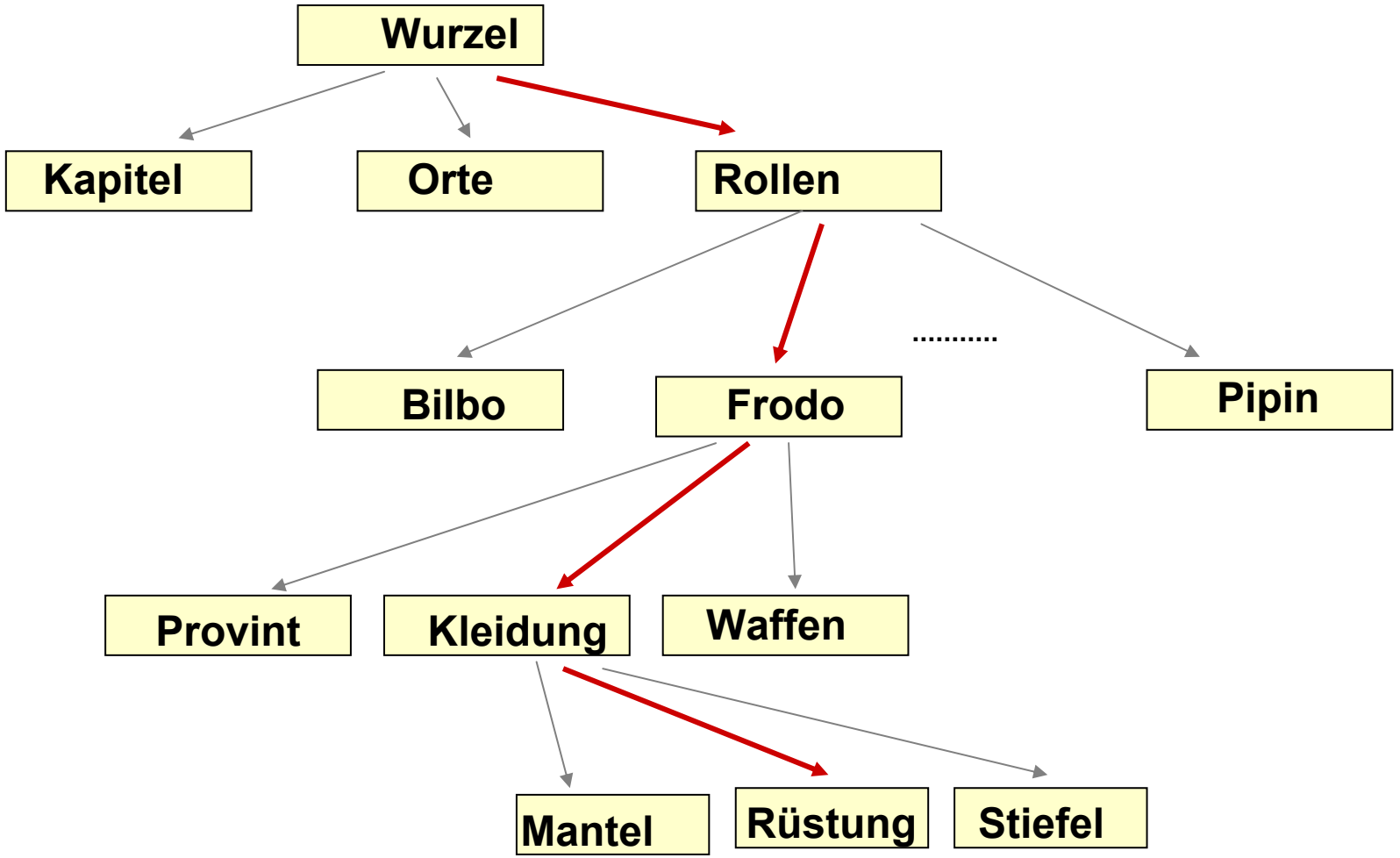
Verzeichnisse und Ordner

Flache Organisationsstruktur: Dateien werden als Sammlung einem Benutzer zugeordnet. Einfache Realisierung.

Hierarchische Verzeichnisstruktur:



Hierarchische Verzeichnisstruktur und Pfadnamen



Verzeichnisse und Pfadnamen

Beispieldialog in Unix: eingegebene Kommandos, Reaktion

```
cd /
pwd
/
ls
bin  boot  dev  etc  home  lib  lost+found  tmp  usr  var
cd
pwd
/usr/kaiser
ls -all
drwxr-xr-x  14 kaiser  root    4096 March 22 18:17    .
drwxr-xr-x   3 root    root    4096 Dec   11 2003     ..
-rw-------   1 kaiser usr   742068 Nov   13 2004    pubsub-12112003.tar.gz
....
....
cd ..
pwd
/usr
```

Operationen auf Verzeichnissen

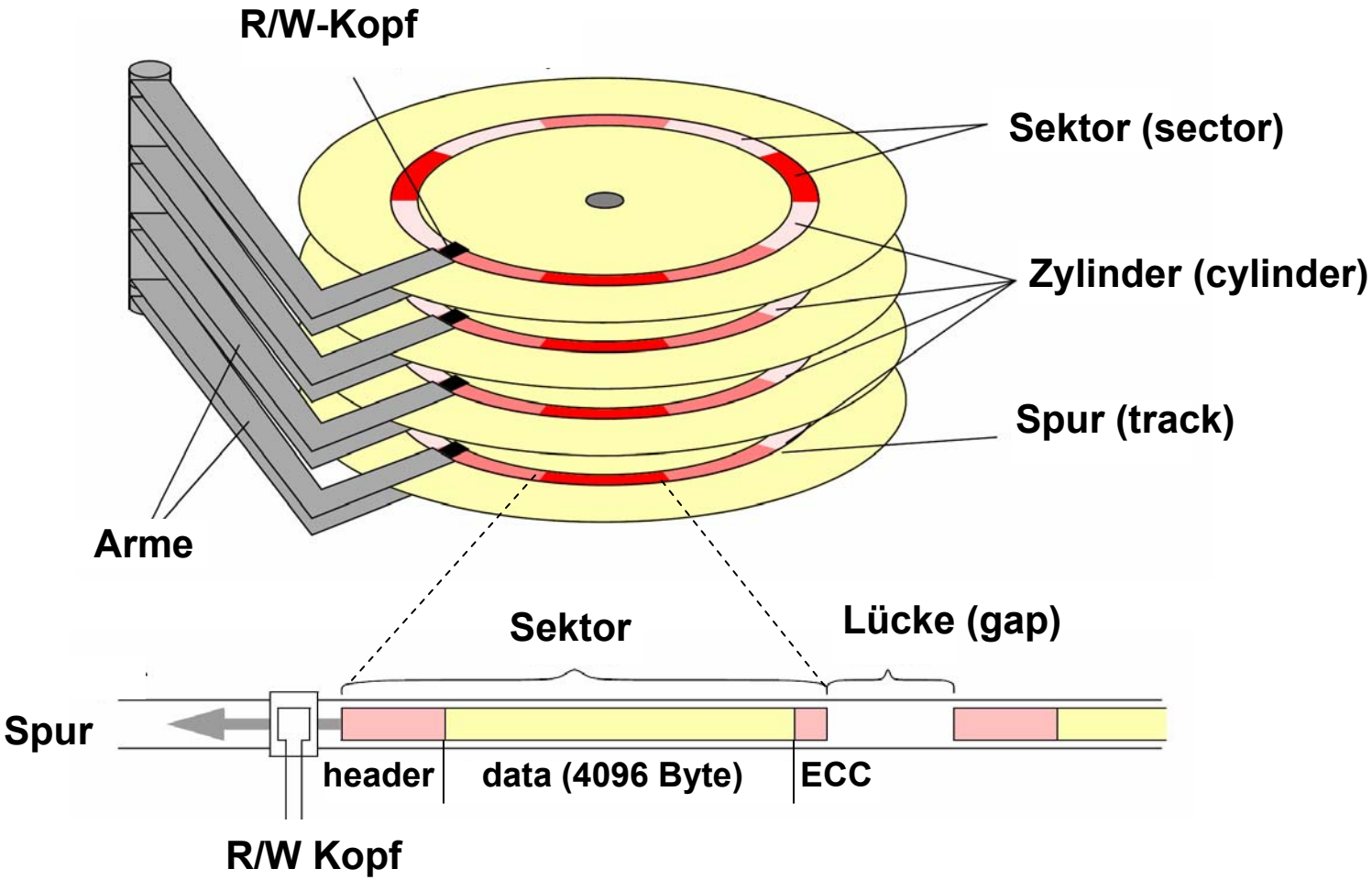
- creat(e)
- delete
- opendir
- closedir
- readdir
- rename
- link
- unlink

Implementierung von Dateisystemen

Punkte:

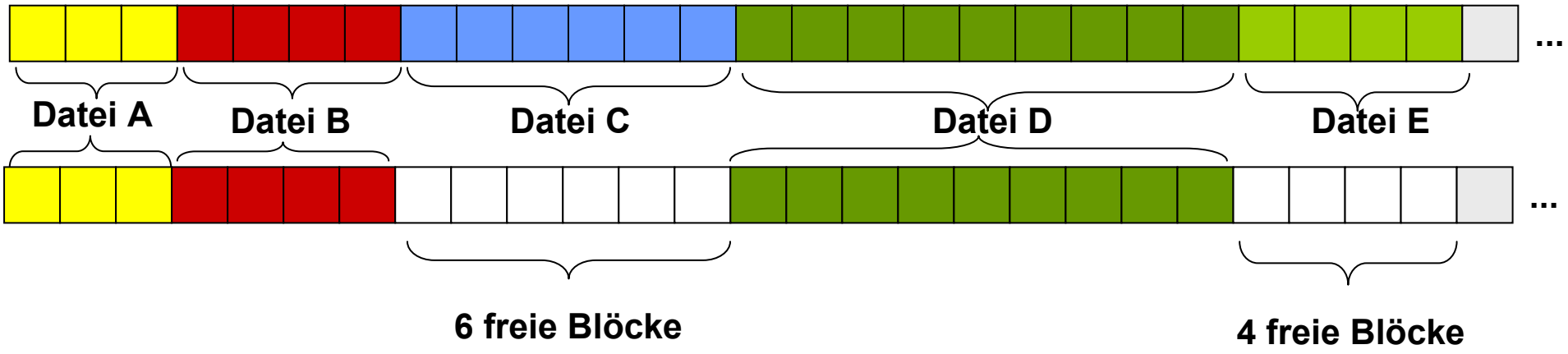
- **Wie werden Dateien auf Plattenblöcke abgebildet?**
- **Wie werden die entsprechenden Plattenblöcke gefunden?**
- **Wie werden Verzeichnisse realisiert?**
- **Wie werden Dateien gemeinsam genutzt?**

(Physische) Organisation einer Platte



Organisationsvariationen

Fortlaufende Belegung von Plattenblöcken



Pro: einfache Implementierung

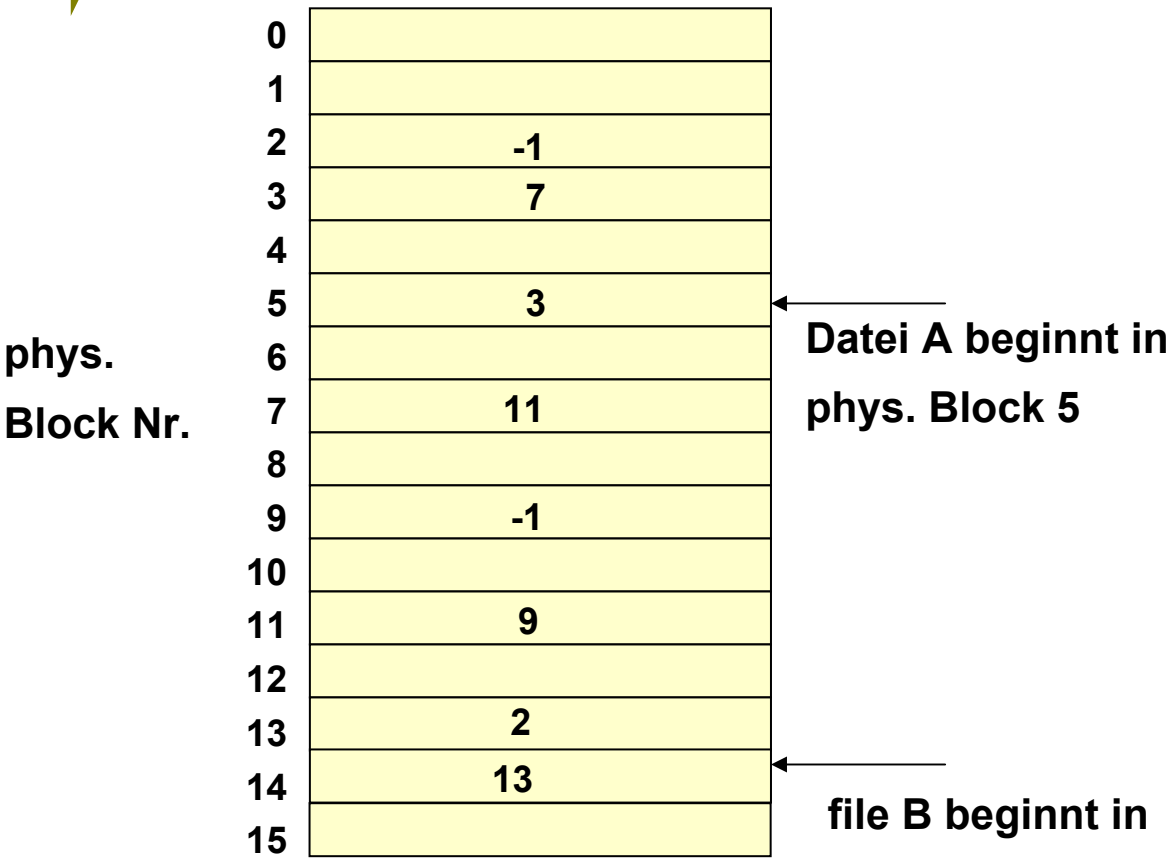
gute Lese-Performanz

Con: Dateigröße muss a priori bekannt sein

Auffinden und Wiederverwendung von freien Blöcken ist schwierig

Organisationsvariationen

verkettete Liste realisiert durch: File Allocation Table (FAT) im Speicher



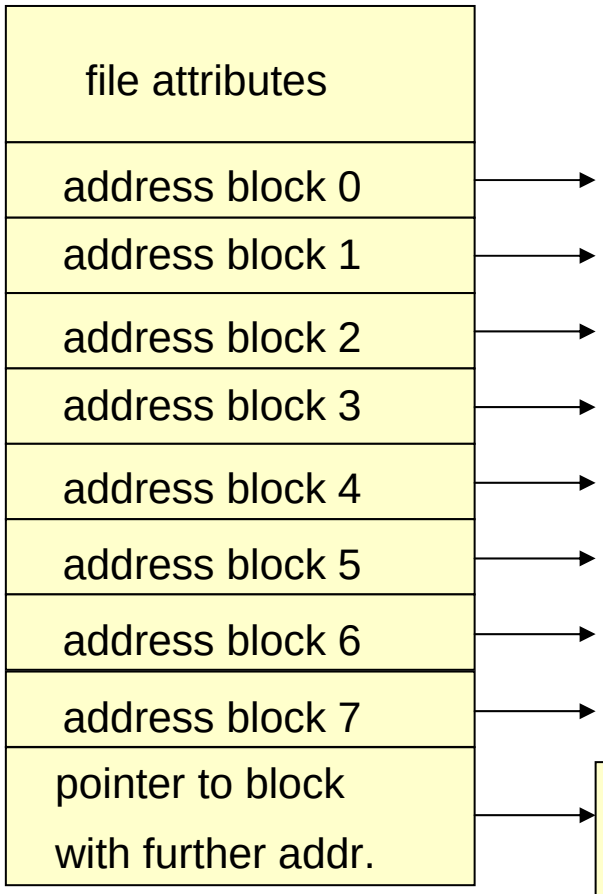
-1: Dateiende symbol

Pro: mildert die Probleme des wahlfreien Zugriffs auf verkettete Listen

Con: die Größe ist proportional zur Plattengröße.

Organisationsvariationen

i-node, inode, index node



Dateiattribute sind z.B.:

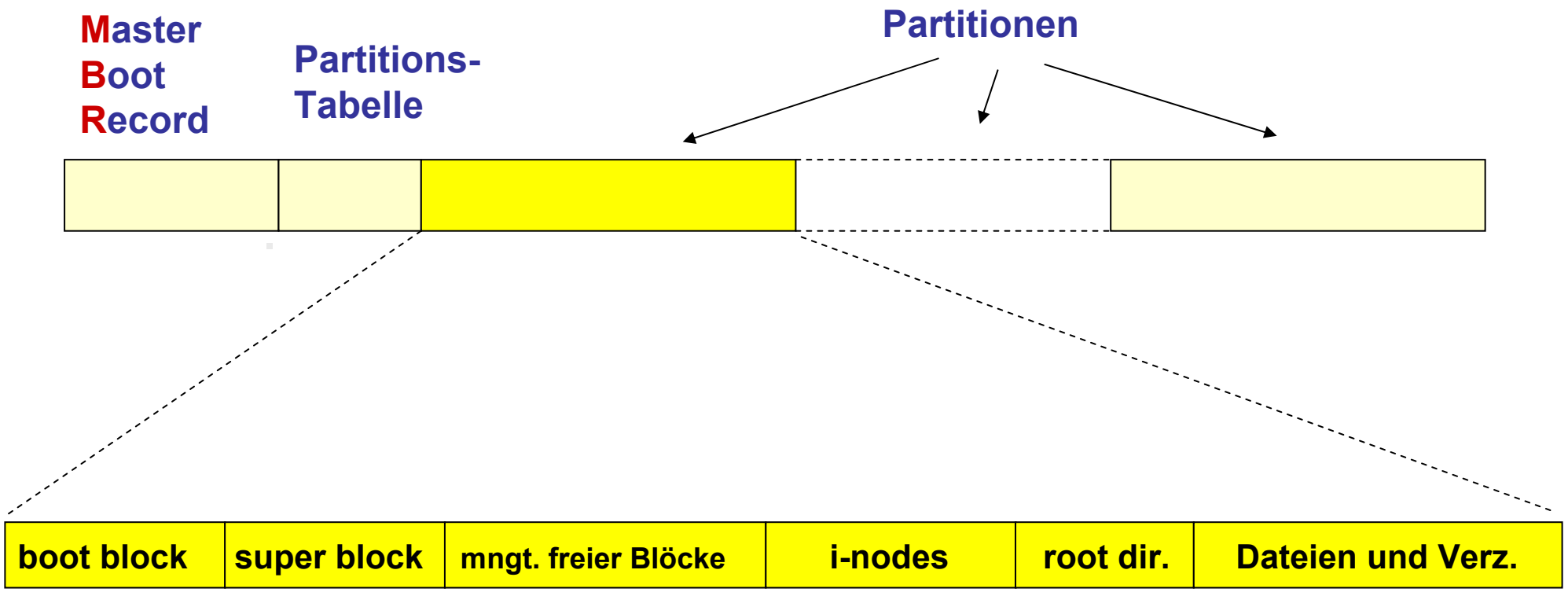
- **owner**
- **file type**
- **access permissions**
- **access time**
- **number of links to the file**
- **file size**

pro: Nur die i-nodes offener Dateien benötigt Platz im Hauptspeicher

nicht mit Plattengröße korelliert

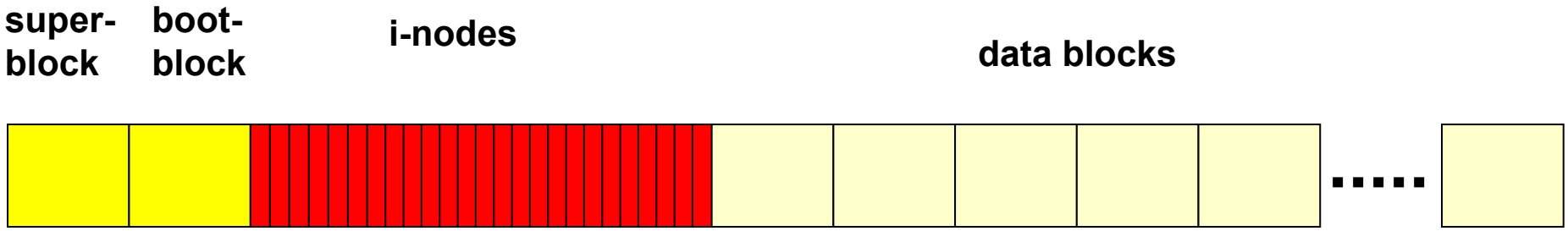
Dateiattribute und Dateiinhalt können getrennt gespeichert werden.

Layout eines Dateisystems



Unix Dateisystem Verwaltung

- Klassisches Unix System



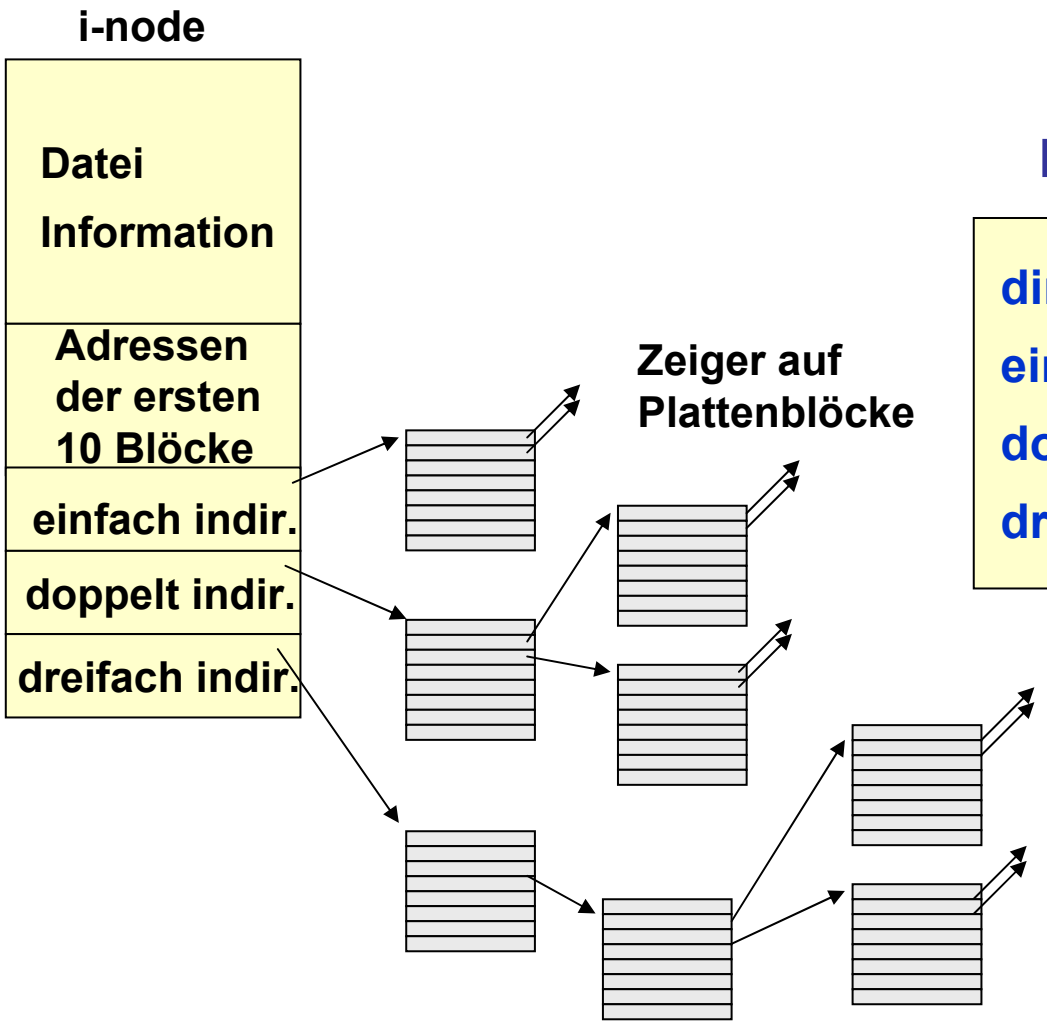
- Berkeley Fast File System:
  - lange Dateinamen (255 charakters)
  - Platte wird in Zylindergruppen mit eigenem Super Block, i-nodes und Datenblöcken strukturiert.
  - 2 Blockgrößen zur effiezienten Verwaltung

- Linux File System: sehr ähnlich zum Berkeley Fast File system.

i-nodes in UNIX

File Mode:	16-Bit Flag which stores access rights
	0 .. 2 rights for "all" users <read, write, exec>
	3 .. 5 rights for the "group" <read, write, exec>
	6 .. 8 rights for "owner" <read, write, exec>
	9 ..11 execution flag
	12..14 file type (regular, char./block-oriented, FIFO pipe)
Link Counter	number of directory references to this i-node
UID	Owner ID
GID	Group ID
Size	in Bytes
File address	39 byte file address information
Last access	date/time
Change of i-node	date/time
Address info for blocks	direct, single ind., double ind., triple ind.

Dateiallokation



Kapazität des UNIX Dateisystems

direkt	10 Blöcke	10 K
einfach indir.	256 Blöcke	256 K
doppelt ind.	64K Blöcke	64 M
dreifach ind.	256x64K Blöcke	16 G

Impelmentierung von Verzeichnissen

welche Information wird in einem Verzeichniseintrag benötigt?

Informationen über den Dateityp.
Wie wird die Datei auf der Platte gefunden?
Zusätzliche information.

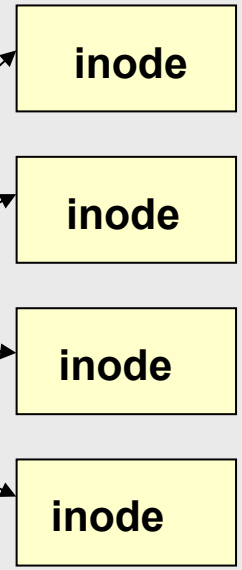
} Dateiattribute

Einfache Verzeichnisstruktur mit Einträgen fester Länge

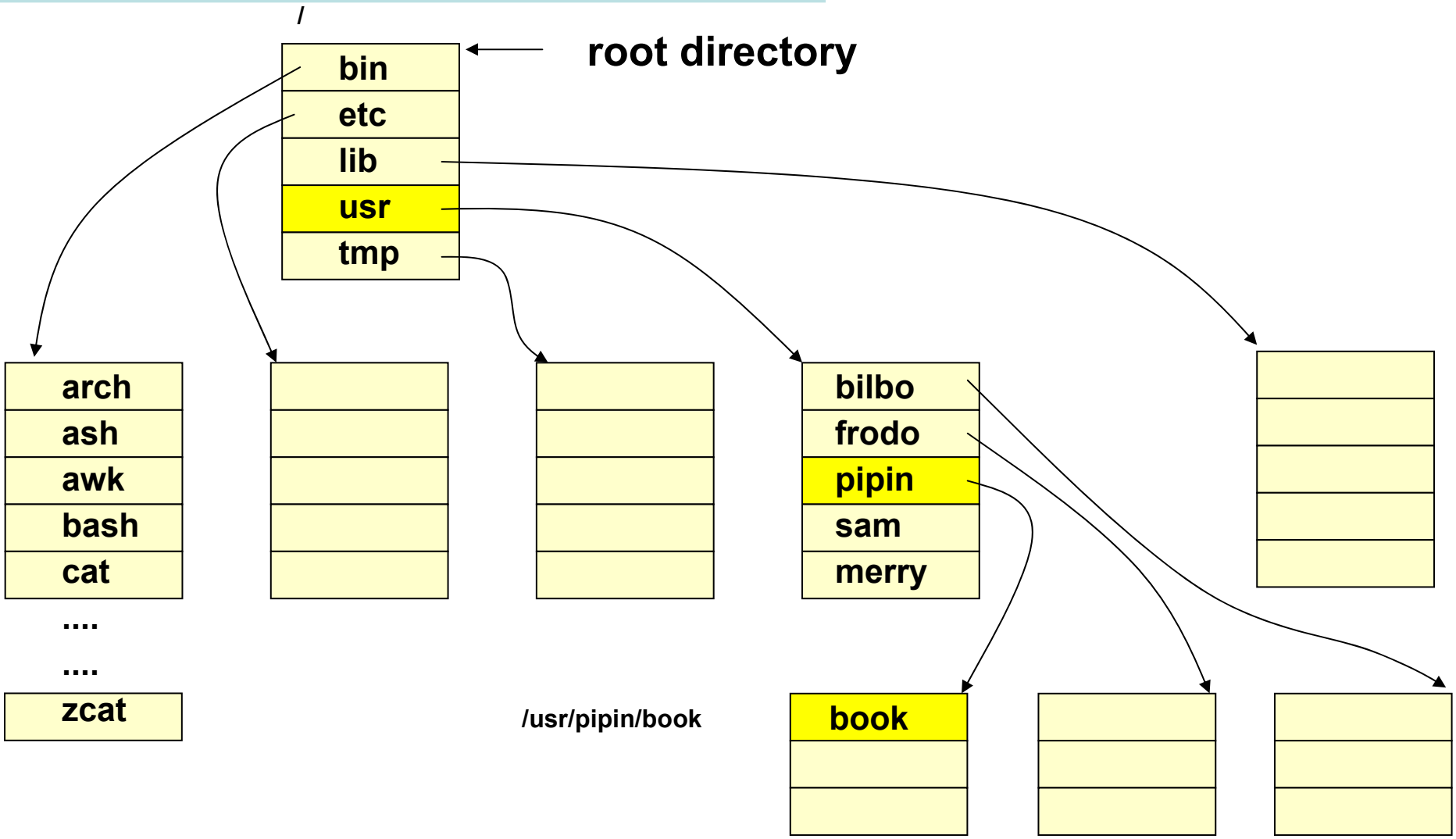
album	attributes
mail	attributes
papers	attributes
courses	attributes

Verzeichnisstruktur, die i-nodes ausnutzt.

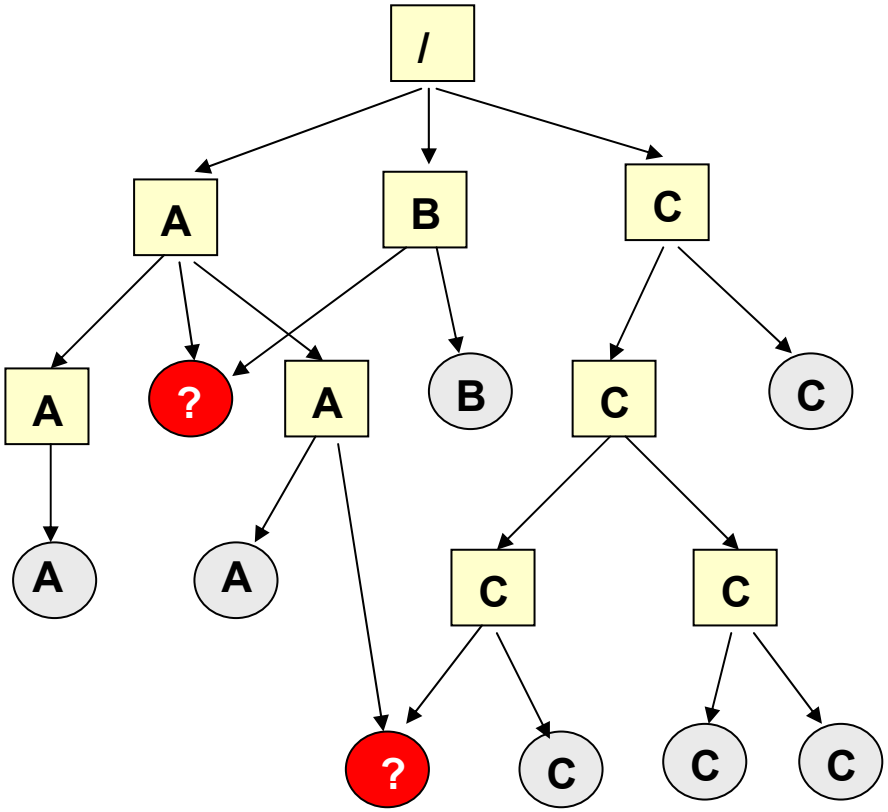
album	inode-no.
mail	inode-no.
papers	inode-no.
courses	inode-no.



UNIX Dateibaum



Gemeinsame Nutzung von Dateien



gerichteter azyklischer Graph

Verzeichnis

Datei

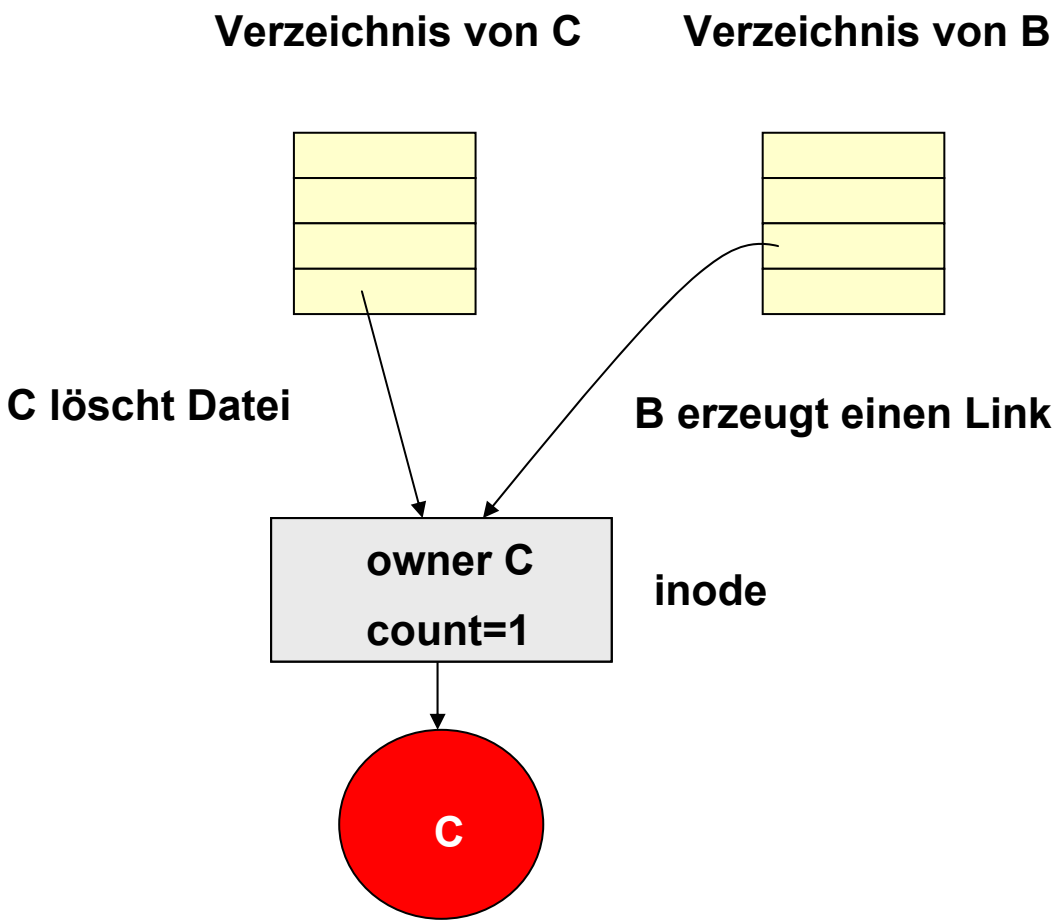
gemeinsam benutzte Datei (shared file)

A,B,C : owner

Probleme:

- Wer ist der Besitzer einer "shared" Datei?
- Wie wird die Sichtbarkeit von Änderungen durchgesetzt?

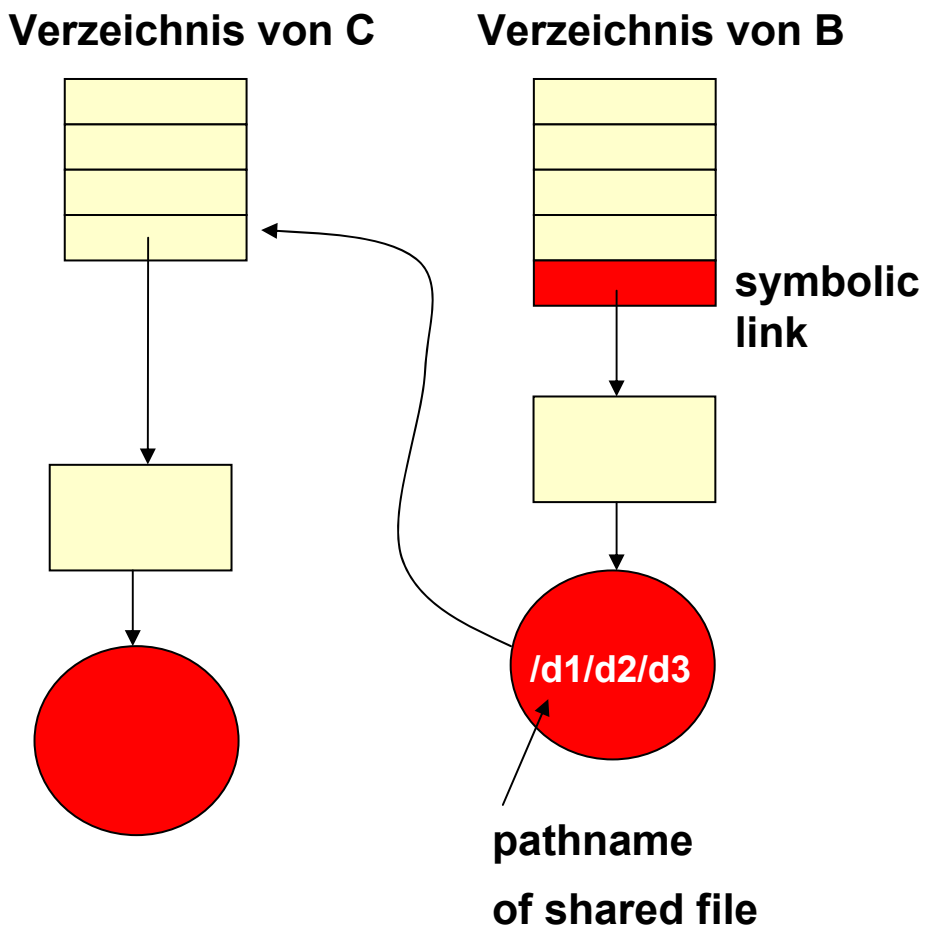
Gemeinsame Nutzung von Dateien



Problem:

B ist der einzige Nutzer einer Datei deren Besitzer C ist.

Datei-Sharing mit symbolischen Links



Besitzer hat volle Kontrolle über die Datei.

Problem: Aufwand

- Analyse und Verfolgung des Pfads benötigt zusätzlich Plattenzugriffe

- zusätzlicher i-node für jeden Link.

Weitere Punkte in Dateisystemen

- Verwaltung des physischen Plattenplatzes
  - Blockgröße
  - Allokation freier Blöcke
  - Plattenquotas
- Zuverlässigkeit
  - Backups und Wiederherstellung (Recovery)
  - Konsistenz
- Leistung eines Dateisystems
  - Caching
  - vorausschauendes Lesen
- Verteilte Dateisysteme

Betriebssysteme II