

# 4 Dateisysteme

- 1 Einführung
- 2 Prozesse und Threads
- 3 Speicherverwaltung
- 4 Dateisysteme**
- 5 Eingabe und Ausgabe
- 6 Deadlocks
- 7 Virtualisierung und die Cloud
- 8 Multiprozessorsysteme
- 9 IT-Sicherheit
- 10 Fallstudie 1: Linux
- 11 Fallstudie 2: Windows
- 12 Entwurf von Betriebssystemen

# 4 Dateisysteme

4.1 Dateien

4.2 Verzeichnisse

4.3 Implementierung von Dateisystemen

4.4 Dateisystemverwaltung und -optimierung

4.5 Beispiele von Dateisystemen

# 4.1 Dateien

4.1.1 Benennung von Dateien

4.1.2 Dateistruktur

4.1.3 Dateitypen

4.1.4 Dateizugriff

4.1.5 Dateiattribute

4.1.6 Dateioperationen

4.1.7 Beispielprogramm

# Dateisysteme (1)

Grundlegende Anforderungen für die langfristige Informationsspeicherung:

1. Es muss möglich sein, eine sehr große Menge an Informationen zu speichern.
2. Informationen müssen die Beendigung des Prozesses überleben.
3. Mehrere Prozesse müssen gleichzeitig auf Informationen zugreifen können.

# Dateisysteme (2)

Stellen Sie sich eine Festplatte/SSD als eine lineare Folge von Blöcken mit fester Größe vor und unterstützen Sie zwei Operationen:

1. Lese Block k.
2. Schreibe Block k

# Dateisysteme (3)

Folgende Fragen stellen sich rasch:

1. Wie finden Sie Informationen?
2. Wie halten Sie einen Benutzer davon ab, die Daten eines anderen Benutzers zu lesen?
3. Woher weiß man, welche Blöcke frei sind?

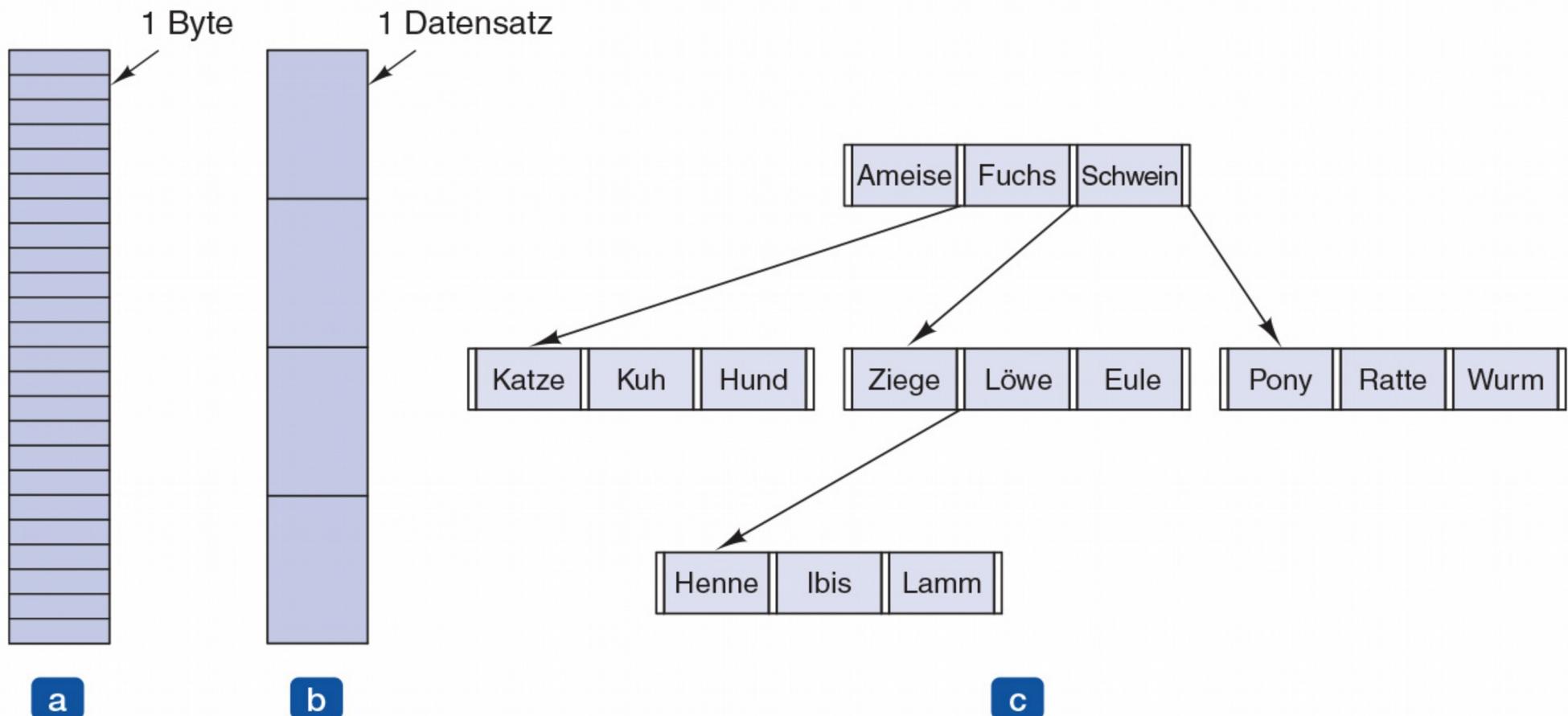
# Typische Dateiendungen

Erweiterung	Bedeutung
.bak	Sicherungsdatei
.c	C-Quelltextdatei
.gif	Bilddatei im Compuserve Graphical Interchange Format
.hlp	Hilfdatei
.html	Dokument in Hypertext Markup Language für das WWW
.eps	Bilddatei nach dem JPEG-Standard codiert
.mp3	Musikdatei im MPEG-Layer-3-Format
.mpg	Film nach dem MPEG-Standard codiert
.o	Objektdatei (übersetzt, noch nicht gebunden)
.pdf	Datei im Portable Document Format
.ps	PostScript-Datei
.tex	Eingabedatei für das TeX-Satzsystem
.txt	Allgemeine Textdatei
.zip	Komprimiertes Archiv

Tanenbaum, A. S.; Bos, H.: Moderne Betriebssysteme. Pearson Studium 2016

Abbildung 4.1: Einige typische Dateiendungen.

# Dateistruktur



**Abbildung 4.2:** Drei Dateiarten: (a) Bytefolge, (b) Folge von Datensätzen, (c) Baum.

# Dateitypen

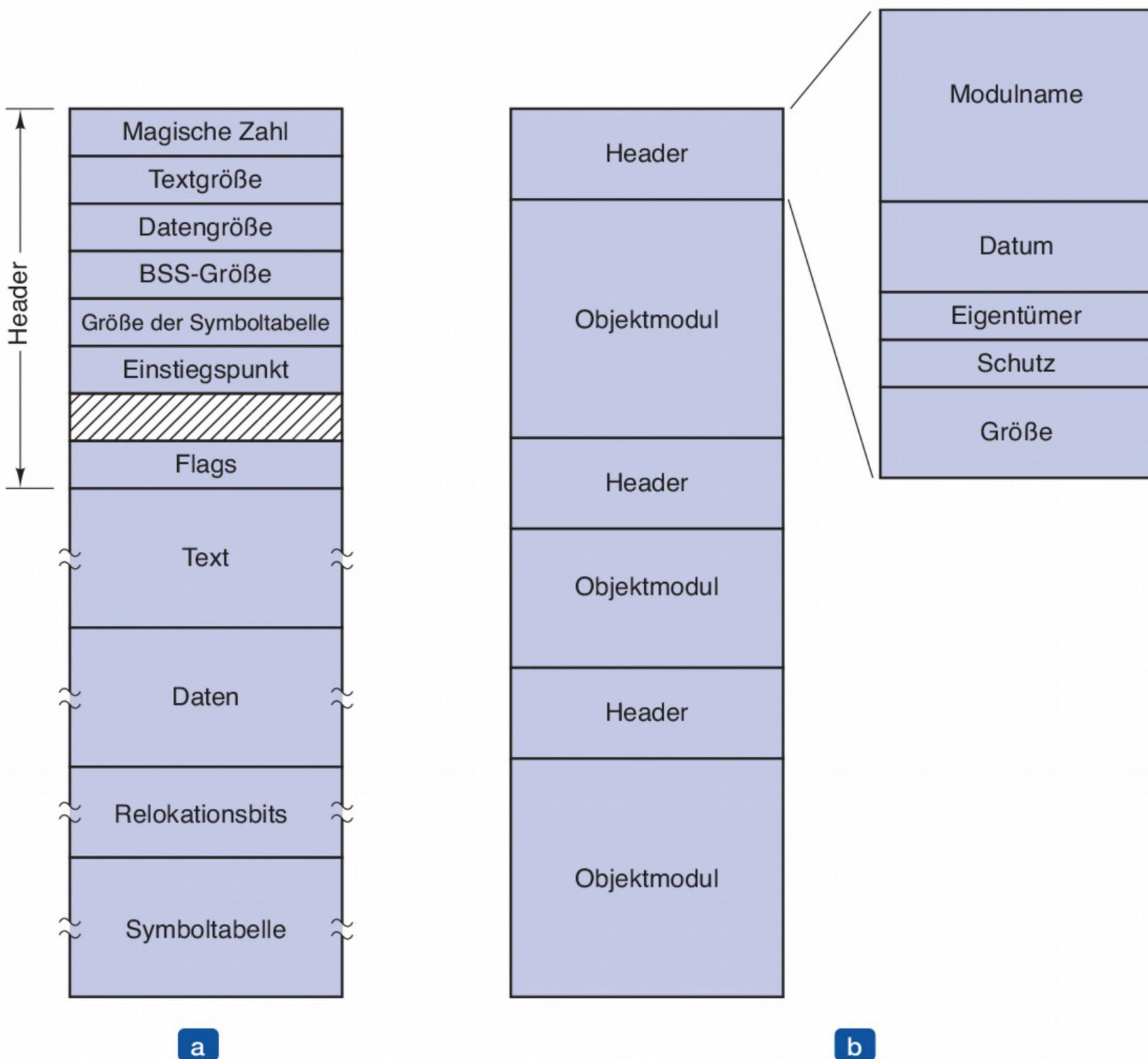


Abbildung 4.3: (a) Ausführbare Datei. (b) Archiv.

# Dateitattribute (1)

Attribute	Bedeutung
Schutz	Wer kann wie auf die Datei zugreifen?
Passwort	Passwort für den Zugriff auf die Datei
Urheber	ID der Person, die die Datei erzeugt hat
Eigentümer	Aktueller Eigentümer
Read-only-Flag	0: Lesen/Schreiben; 1: nur Lesen
Hidden-Flag	0: normal; 1: in Listen nicht sichtbar
System-Flag	0: normale Datei; 1: Systemdatei
Archiv-Flag	0: wurde gesichert; 1: muss noch gesichert werden
ASCII/Binär-Flag	0: ASCII-Datei; 1: Binärdatei
Random-Access-Flag	0: nur sequenzieller Zugriff; 1: wahlfreier Zugriff
Temporary-Flag	0: normal; 1: Datei bei Prozessende löschen
Sperr-Flags	0: nicht gesperrt; nicht null: gesperrt
Datensatzlänge	Anzahl der Bytes in einem Datensatz

## Dateiattribute (2)

Schlüsselposition	Offset des Schlüssels innerhalb des Datensatzes
Schlüssellänge	Anzahl der Bytes im Schlüsselfeld
Erstellungszeit	Datum und Zeitpunkt der Dateierstellung
Zeitpunkt des letzten Zugriffs	Datum und Zeitpunkt des letzten Zugriffs
Zeitpunkt der letzten Änderung	Datum und Zeitpunkt der letzten Dateiänderung
Aktuelle Größe	Anzahl der Bytes in der Datei
Maximale Größe	Anzahl der Bytes für maximale Größe der Datei

**Abbildung 4.4:** Einige mögliche Dateiattribute.

# Dateioperationen

- 1. Create
- 2. Delete
- 3. Open
- 4. Close
- 5. Read
- 6. Write
- 7. Append
- 8. Seek
- 9. Get attributes
- 10. Set attributes
- 11. Rename

# Beispielprogramm zum Kopieren einer Datei (1)

Tanenbaum, A. S.; Bos, H.: Moderne  
Betriebssysteme. Pearson Studium 2016

```
/* Dateikopierprogramm. Fehlerbehandlung und -bericht sind minimal. */

#include <sys/types.h>           /* benötigte Header-Dateien */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char *argv[]); /* ANSI-Prototyp */

#define BUF_SIZE 4096             /* benutzt Puffergröße von */
                                /* 4096 Byte */
#define OUTPUT_MODE 0700          /* Schutzbits für Ausgabedatei */

int main (int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit (1);      /* Syntaxfehler, wenn „argc“ */
                                /* nicht 3 ist */

    /* Eingabedatei öffnen und Ausgabedatei erzeugen */
    in_fd = open (argv[1], O_RDONLY); /* Quelldatei öffnen */
    if (in_fd < 0) exit (2);       /* Beenden, wenn nicht */
                                /* geöffnet werden kann */

    out_fd = creat (argv[2], OUTPUT_MODE); /* Zielfile erzeugen */
    if (out_fd < 0) exit (3);     /* Beenden, wenn Er-*/
                                /* zeugung nicht möglich */
```

# Beispielprogramm zum Kopieren einer Datei (2)

```
/* Kopierschleife */
while (TRUE) {
    rd_count = read (in_fd, buffer, BUF_SIZE);      /* Datenblock */
    /* lesen */
    /* Ende der Datei */
    /* oder Fehler */
    wt_count = write (out_fd, buffer, rd_count);    /* Daten */
    /* schreiben */
    /* wt_count <= 0 */
    /* bedeutet Fehler */
    if (rd_count <= 0) break;
    if (wt_count <= 0) exit (4);
}

/* Dateien schließen */
close (in_fd);
close (out_fd);
if (rd_count == 0)          /* kein Fehler beim letzten Lesen */
    exit (0);
else
    exit (5);                /* Fehler beim letzten Lesen */
}
```

**Abbildung 4.5:** Ein einfaches Programm zum Kopieren einer Datei.

## 4.2 Verzeichnisse

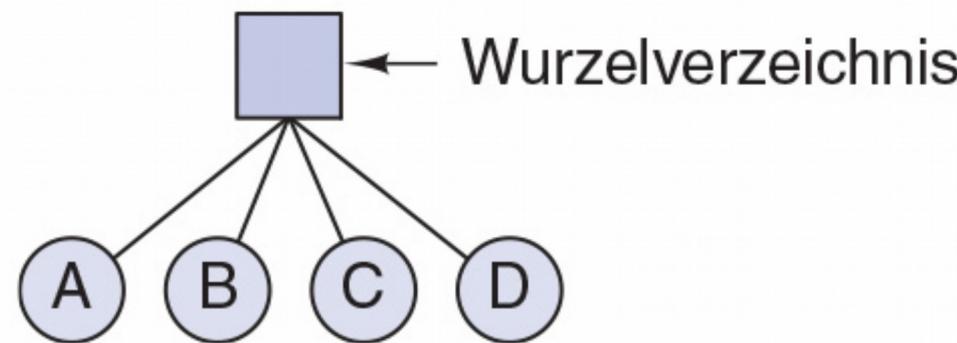
4.2.1 Verzeichnissysteme mit einer Ebene

4.2.2 Hierarchische Verzeichnissysteme

4.2.3 Pfadnamen

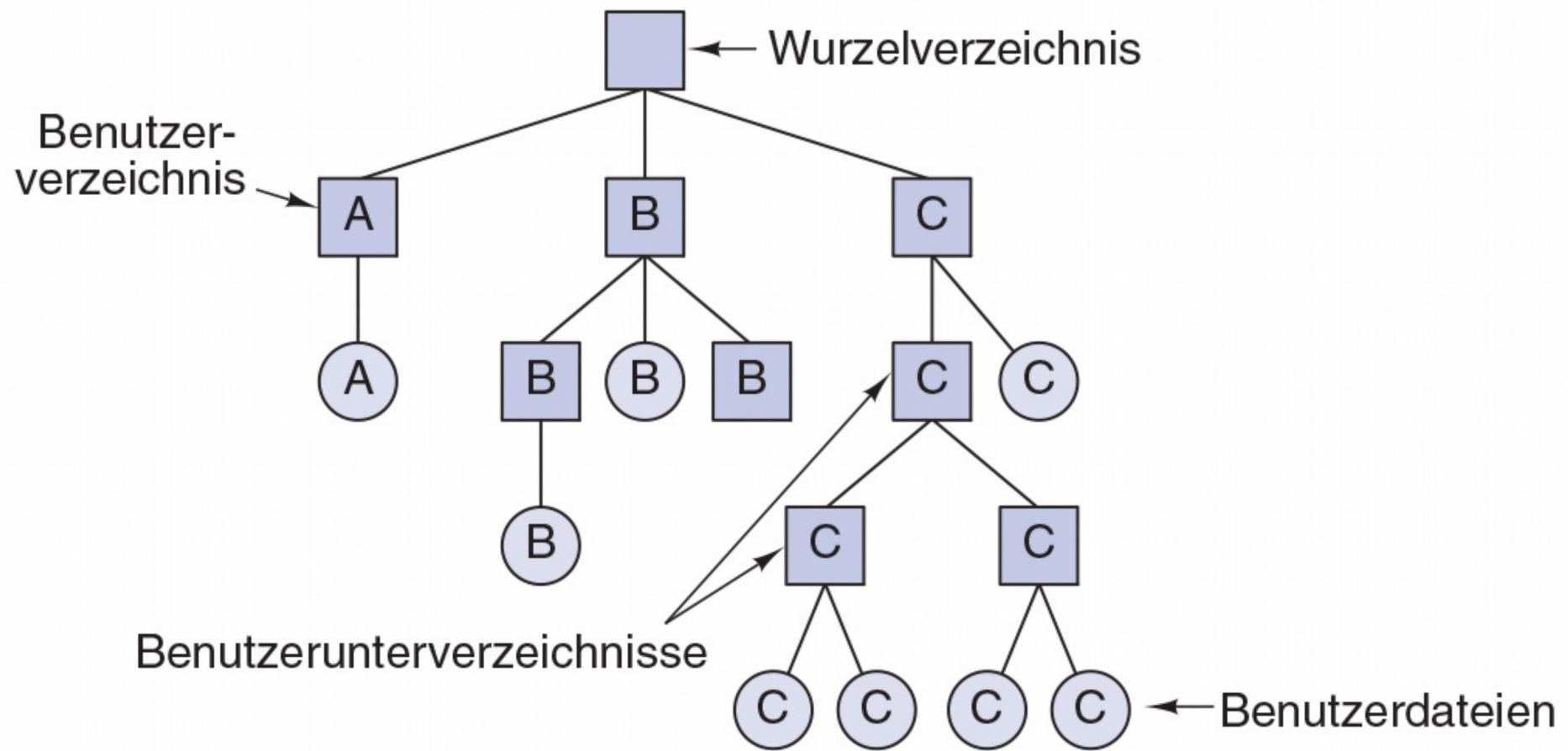
4.2.4 Operationen auf Verzeichnissen

# Verzeichnissysteme mit einer Ebene



**Abbildung 4.6:** Ein Verzeichnissystem mit einer Ebene und vier Dateien.

# Hierarchische Verzeichnissysteme



**Abbildung 4.7:** Ein hierarchisches Verzeichnissystem.

# Pfadnamen

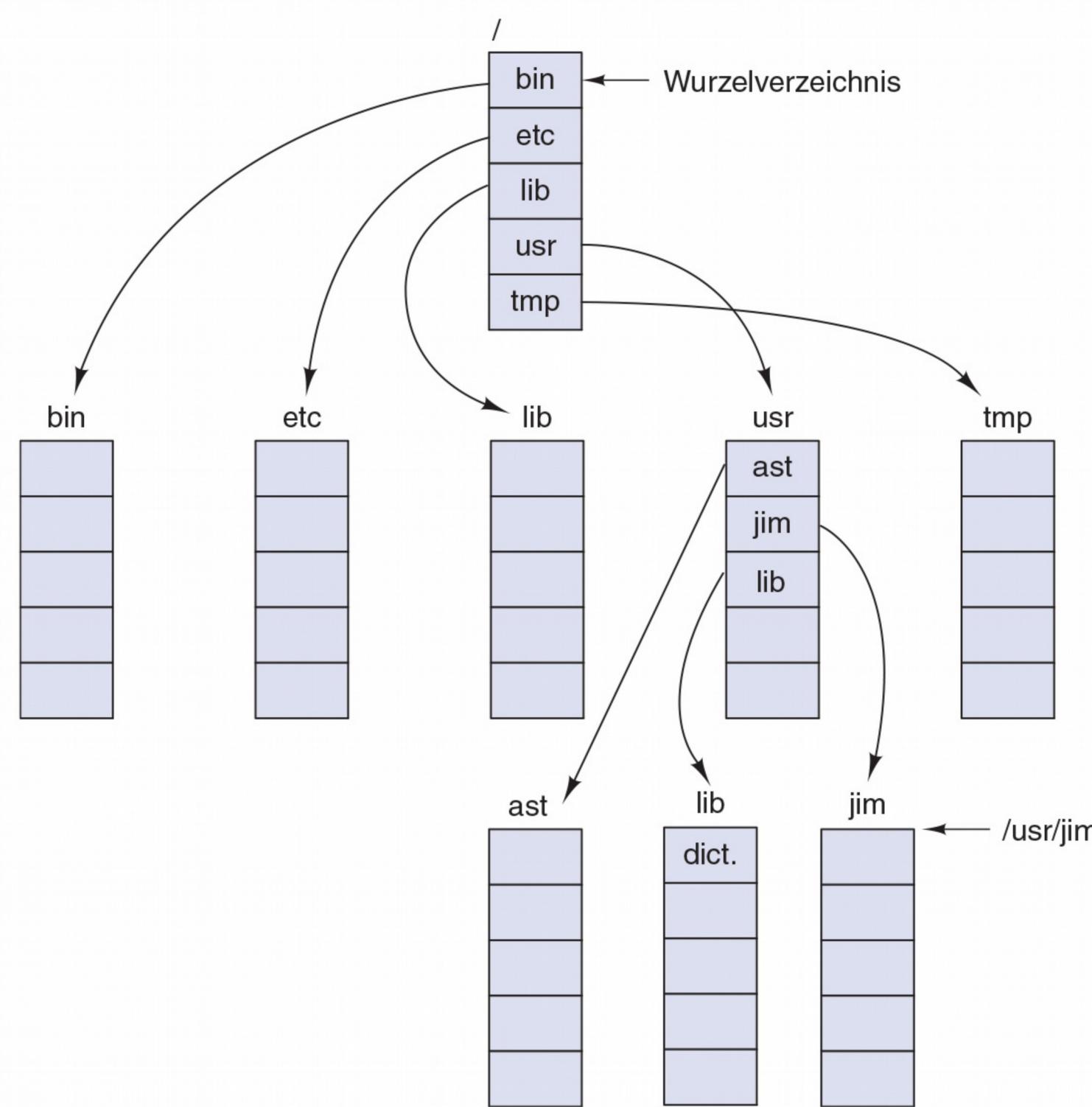


Abbildung 4.8: UNIX-Verzeichnisbaum.

Tanenbaum, A. S.; Bos, H.: Moderne Betriebssysteme. Pearson Studium 2016

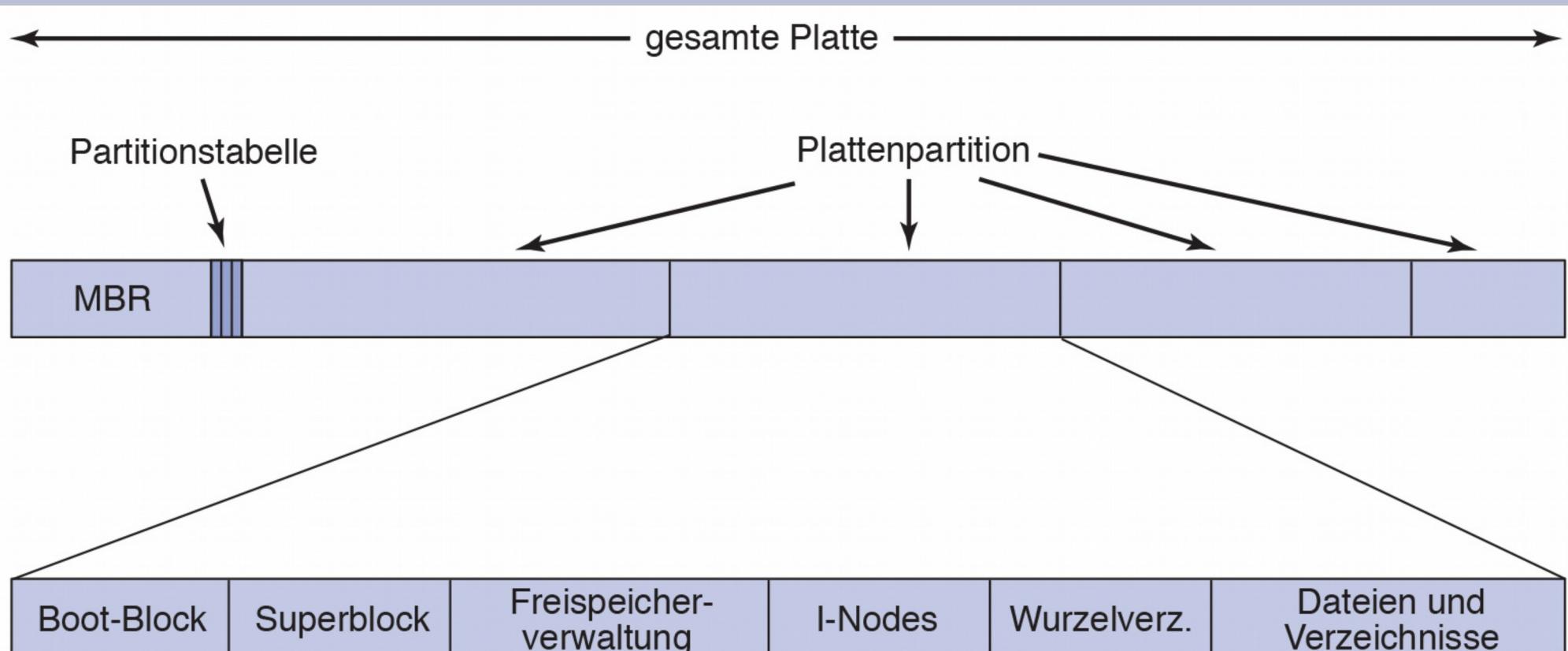
# Operationen auf Verzeichnissen

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

## 4.3 Implementierung von Dateisystemen

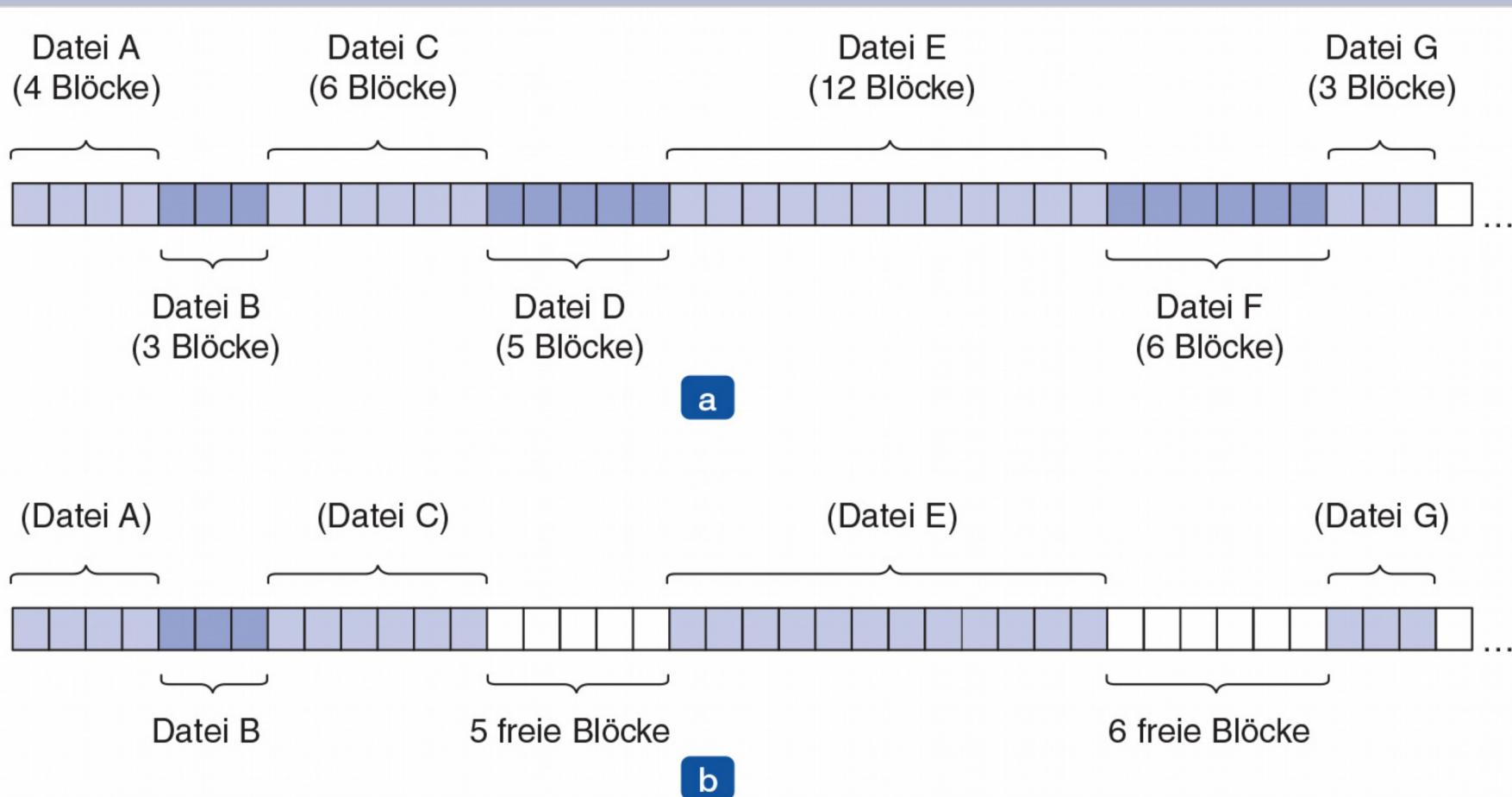
- 4.3.1 Layout eines Dateisystems
- 4.3.2 Implementierung von Dateien
- 4.3.3 Implementierung von Verzeichnissen
- 4.3.4 Gemeinsam benutzte Dateien
- 4.3.5 Log-basierte Dateisysteme
- 4.3.6 Journaling-Dateisysteme
- 4.3.7 Virtuelle Dateisysteme

# Layout eines Dateisystems



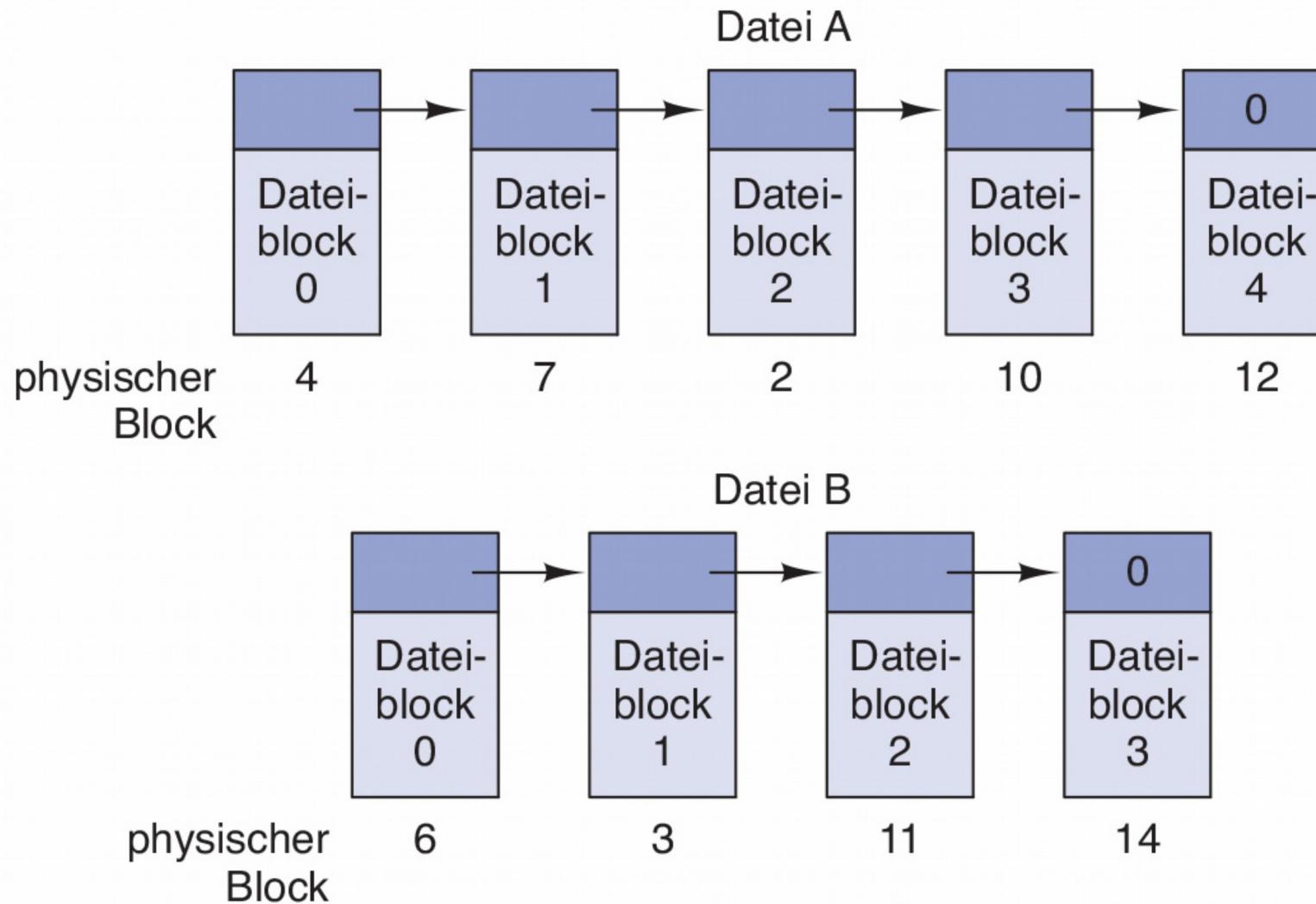
**Abbildung 4.9:** Ein mögliches Layout eines Dateisystems.

# Implementierung von Dateien – zusammenhängendes Layout



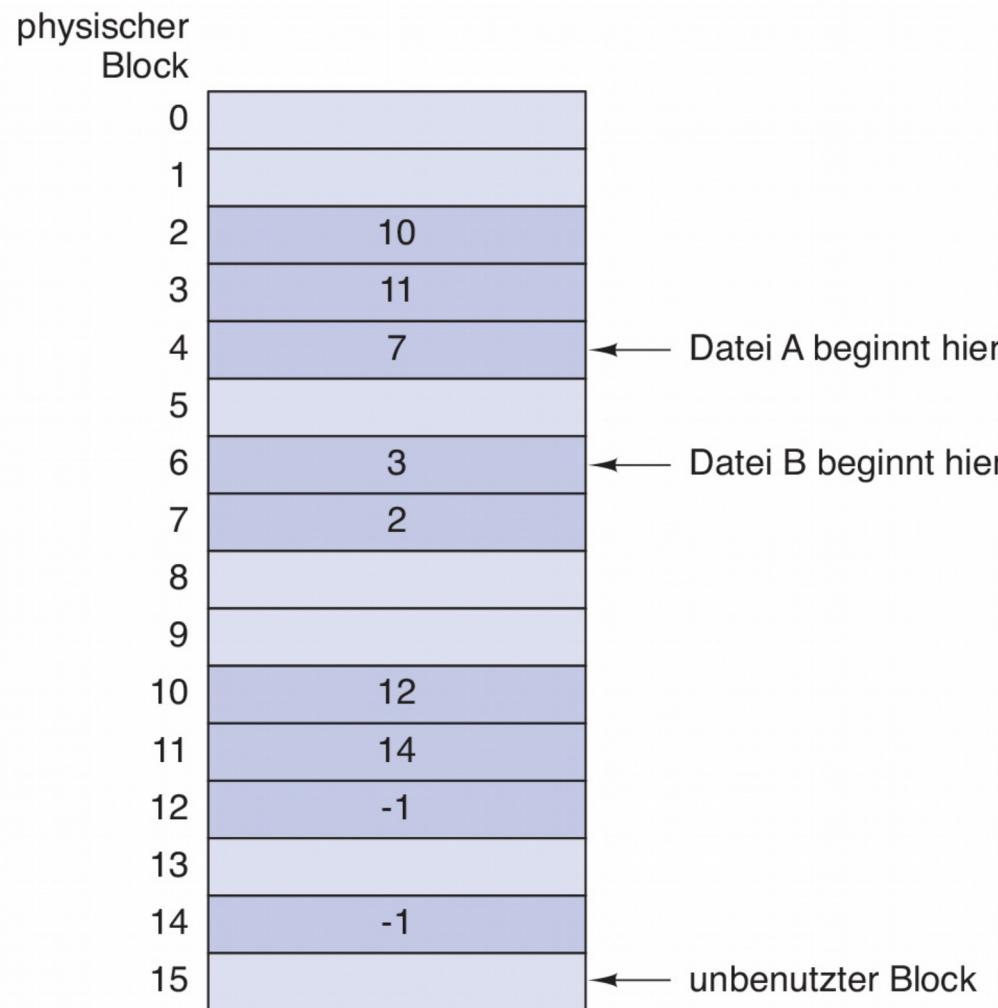
**Abbildung 4.10:** (a) Zusammenhängende Belegung des Plattenplatzes für sieben Dateien. (b) Zustand der Platte, nachdem die Dateien *D* und *F* entfernt worden sind.

# Implementierung von Dateien – verkettete Liste von Plattenblöcken



**Abbildung 4.11:** Die Speicherung einer Datei als verkettete Liste von Plattenblöcken.

# Implementierung von Dateien – verkettete Liste von Plattenblöcken



**Abbildung 4.12:** Die Belegung mit verketteten Listen unter Verwendung einer Datei-Allokationstabelle (FAT) im Arbeitsspeicher.

# Implementierung von Dateien – I-nodes

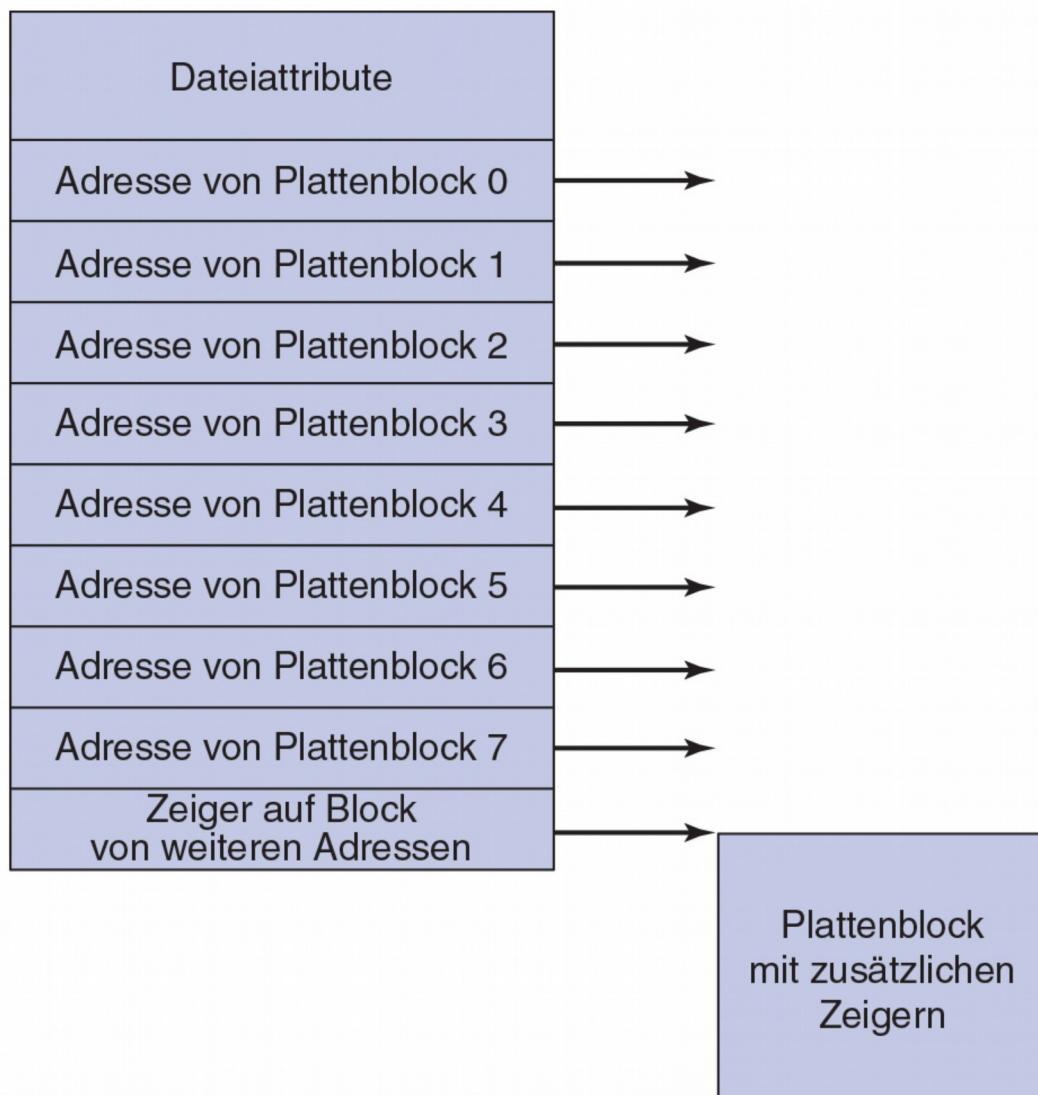
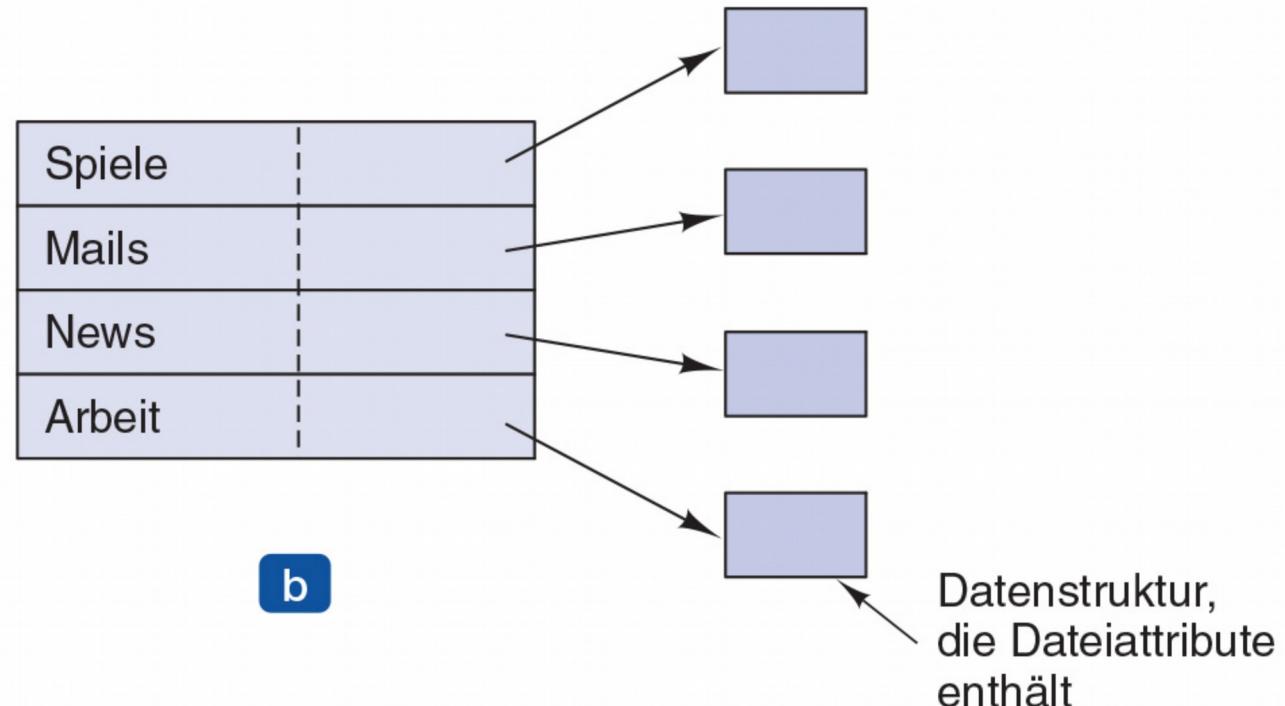


Abbildung 4.13: Beispiel für einen I-Node.

# Implementierung von Verzeichnissen (1)

Spiele	Attribute
Mails	Attribute
News	Attribute
Arbeit	Attribute

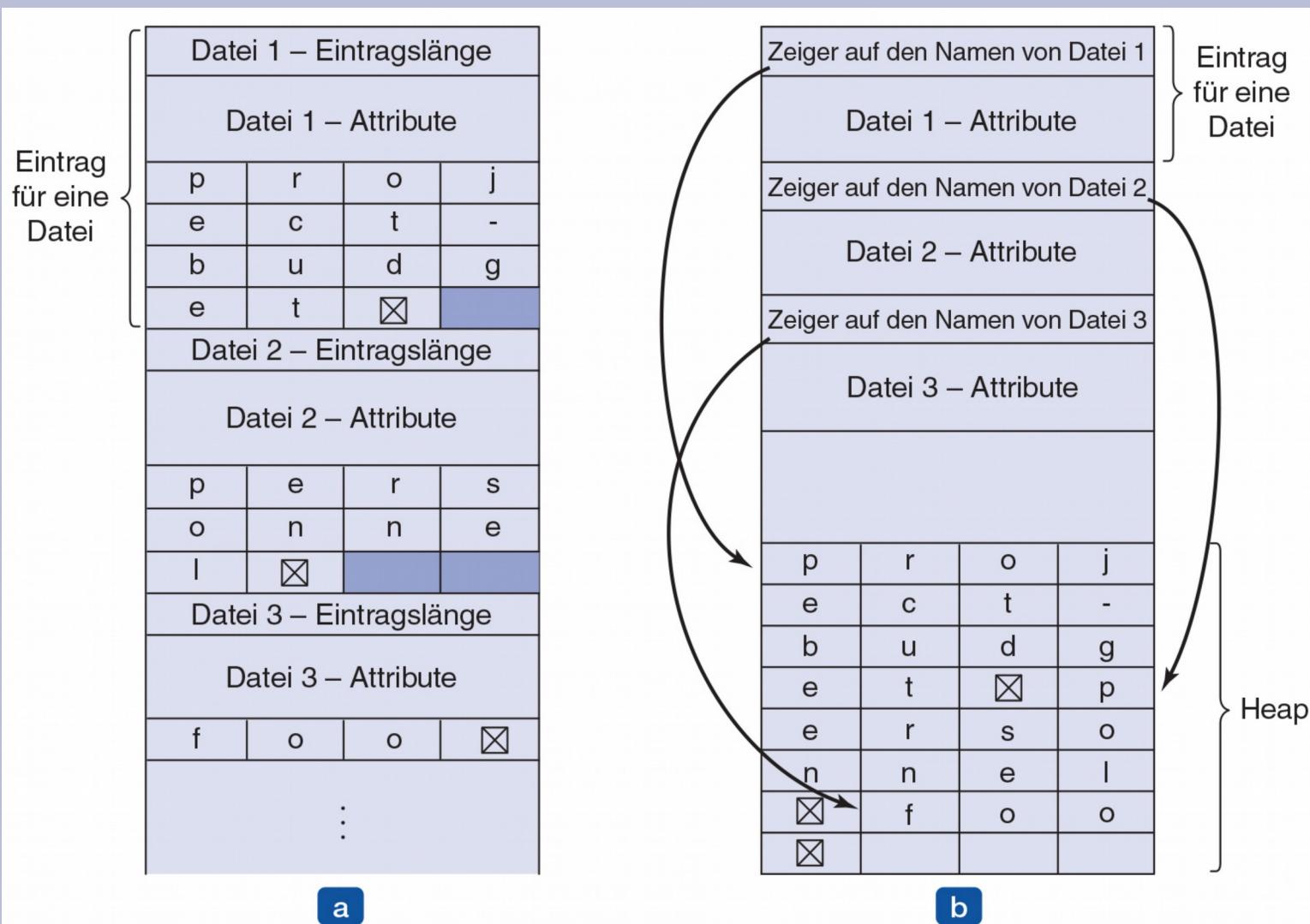
a



b

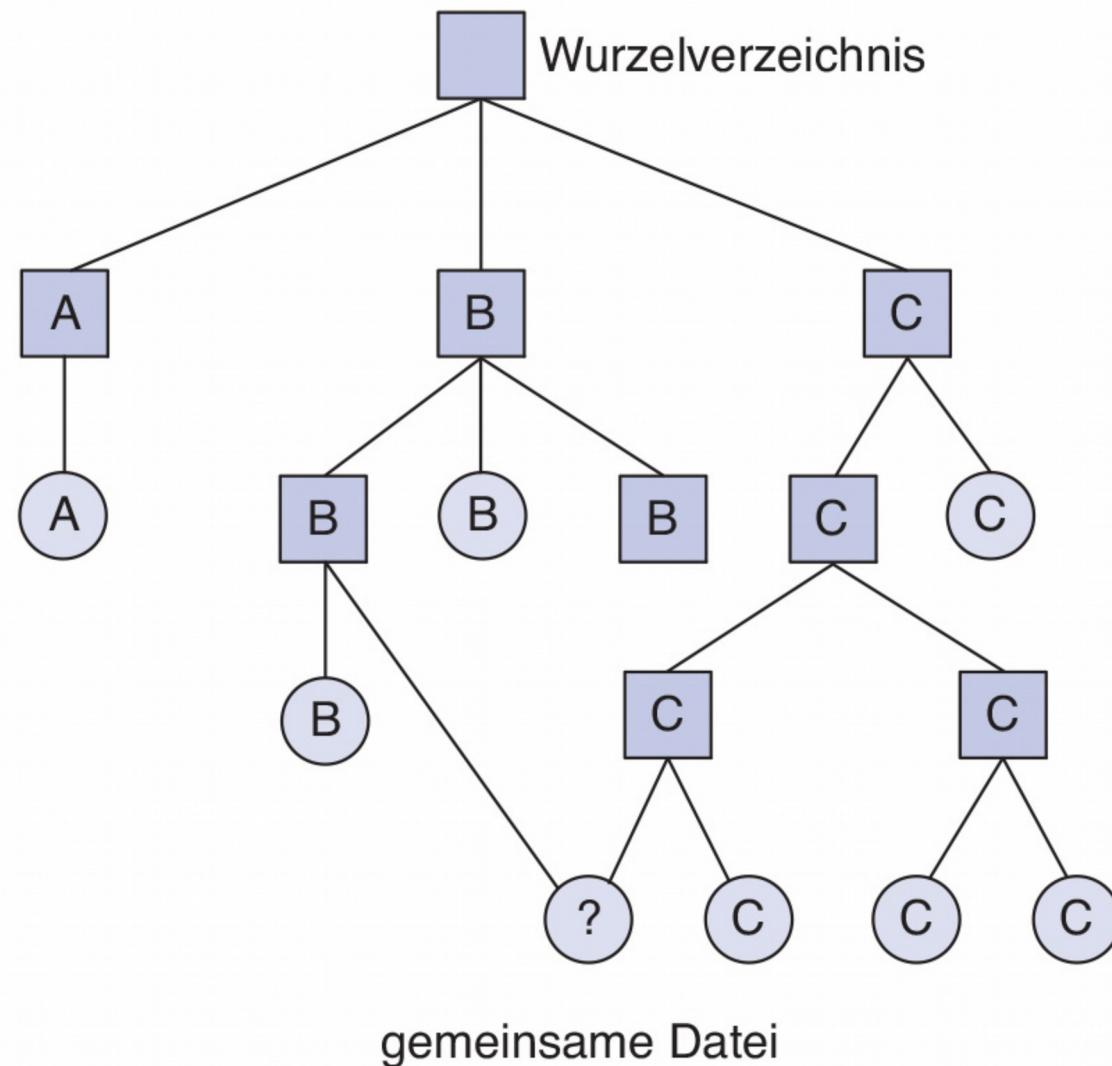
**Abbildung 4.14:** (a) Ein einfaches Verzeichnis mit Einträgen fester Länge der Plattenadressen und Attribute im Verzeichniseintrag. (b) Ein Verzeichnis, bei dem sich jeder Eintrag nur auf einen I-Node bezieht.

# Implementierung von Verzeichnissen (2)



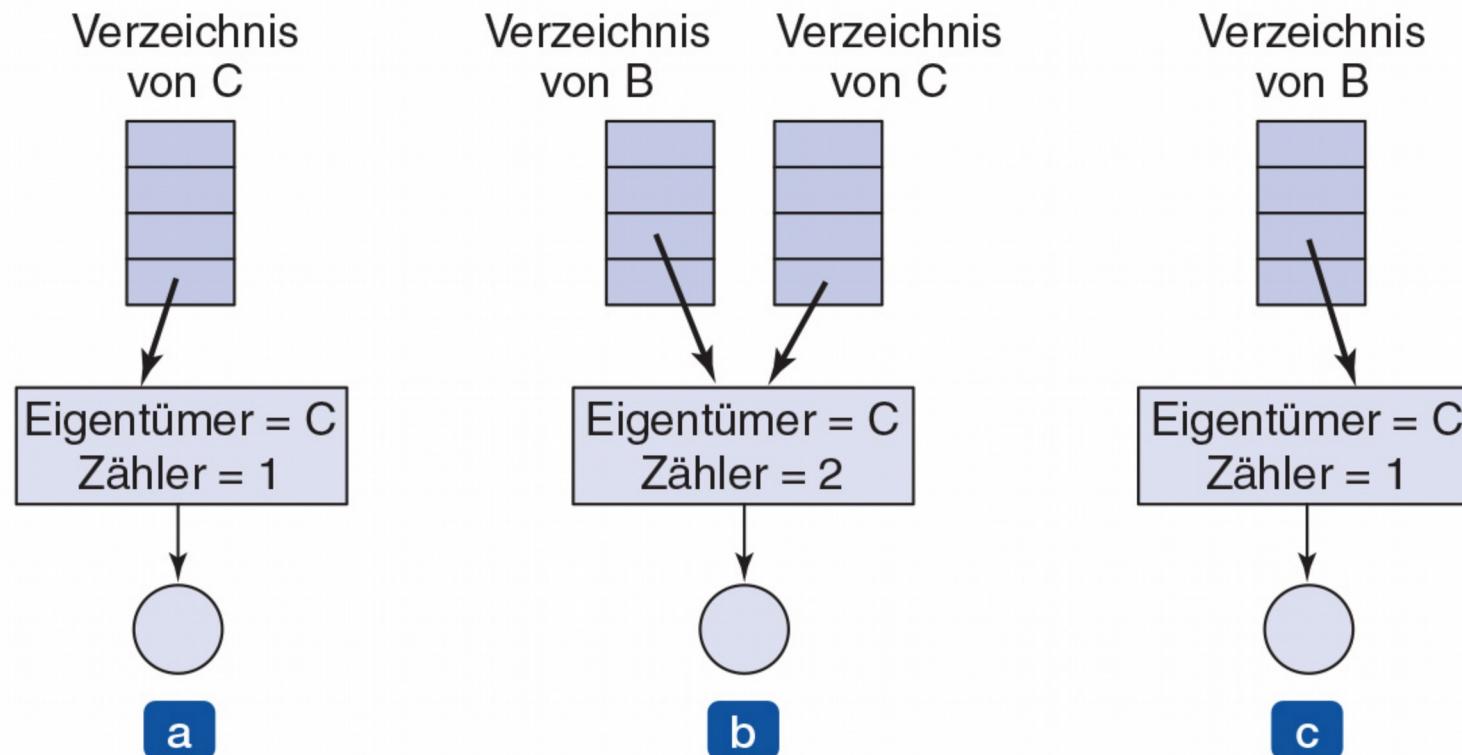
**Abbildung 4.15:** Zwei Möglichkeiten, mit langen Dateinamen in Verzeichnissen umzugehen: (a) linear, (b) mit einem Heap.

# Gemeinsam benutzte Dateien (1)



**Abbildung 4.16:** Ein Dateisystem mit einer gemeinsam benutzten Datei.

# Gemeinsam benutzte Dateien (2)



**Abbildung 4.17:** (a) Situation vor der Verlinkung. (b) Situation nach der Erzeugung des Links. (c) Situation, nachdem der Eigentümer die Datei entfernt hat.

# Log-basierte Dateisysteme

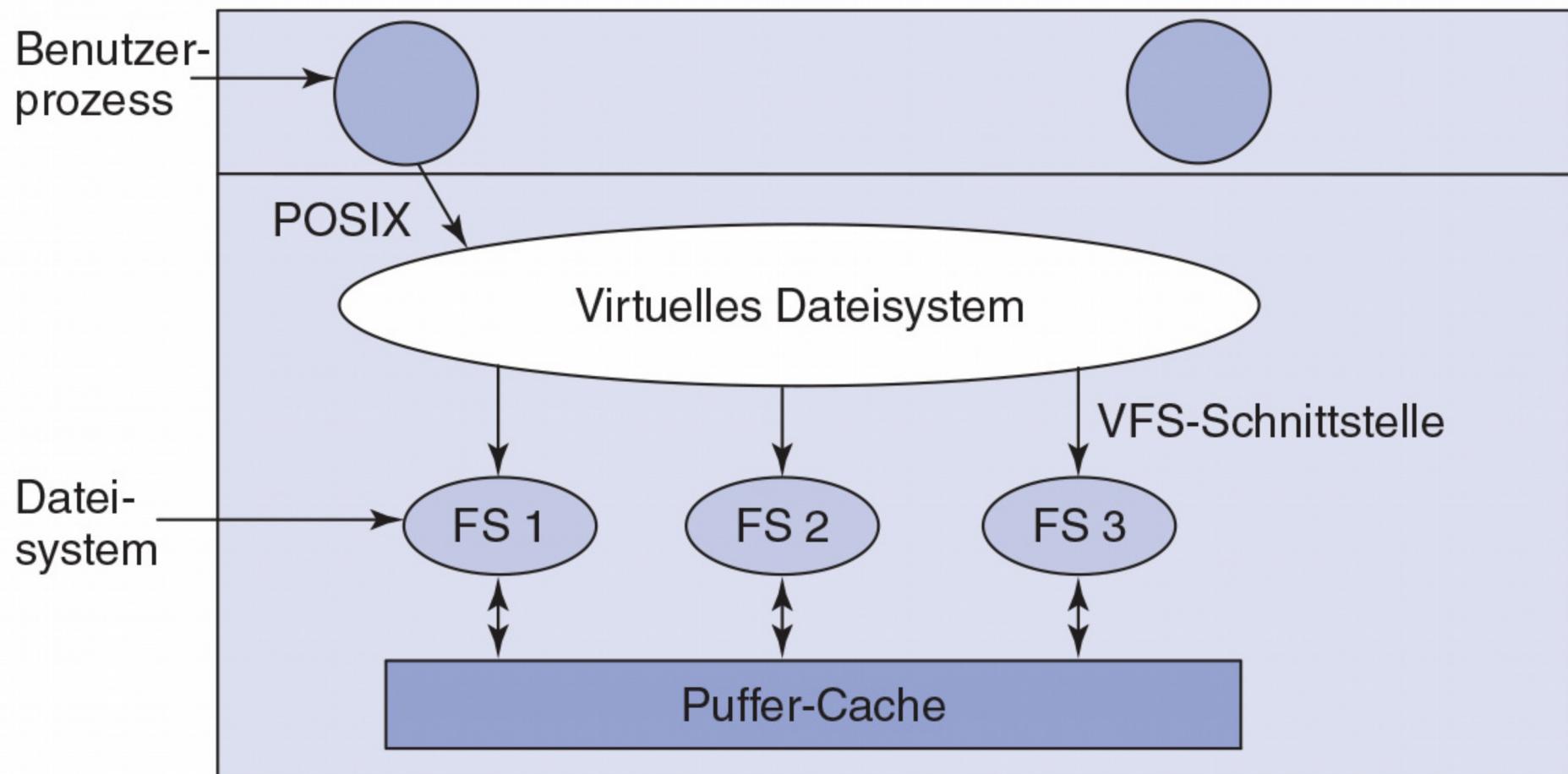
- I-nodes existieren weiter, stehen aber nicht auf fester Position
- I-nodes werden über das Log verstreut
- Auffinden I-nodes schwieriger:
  - I-node Map auf Festplatte (ggf. Kopie im Speicher)
- Schreibvorgänge:
  - werden im Speicher gesammelt
  - Regelmäßig als ein Segment an Ende des Logs geschrieben
- Problem: Log belegt irgendwann die ganze Platte

# Journaling-Dateisysteme

Schritte zum Entfernen einer Datei in UNIX:

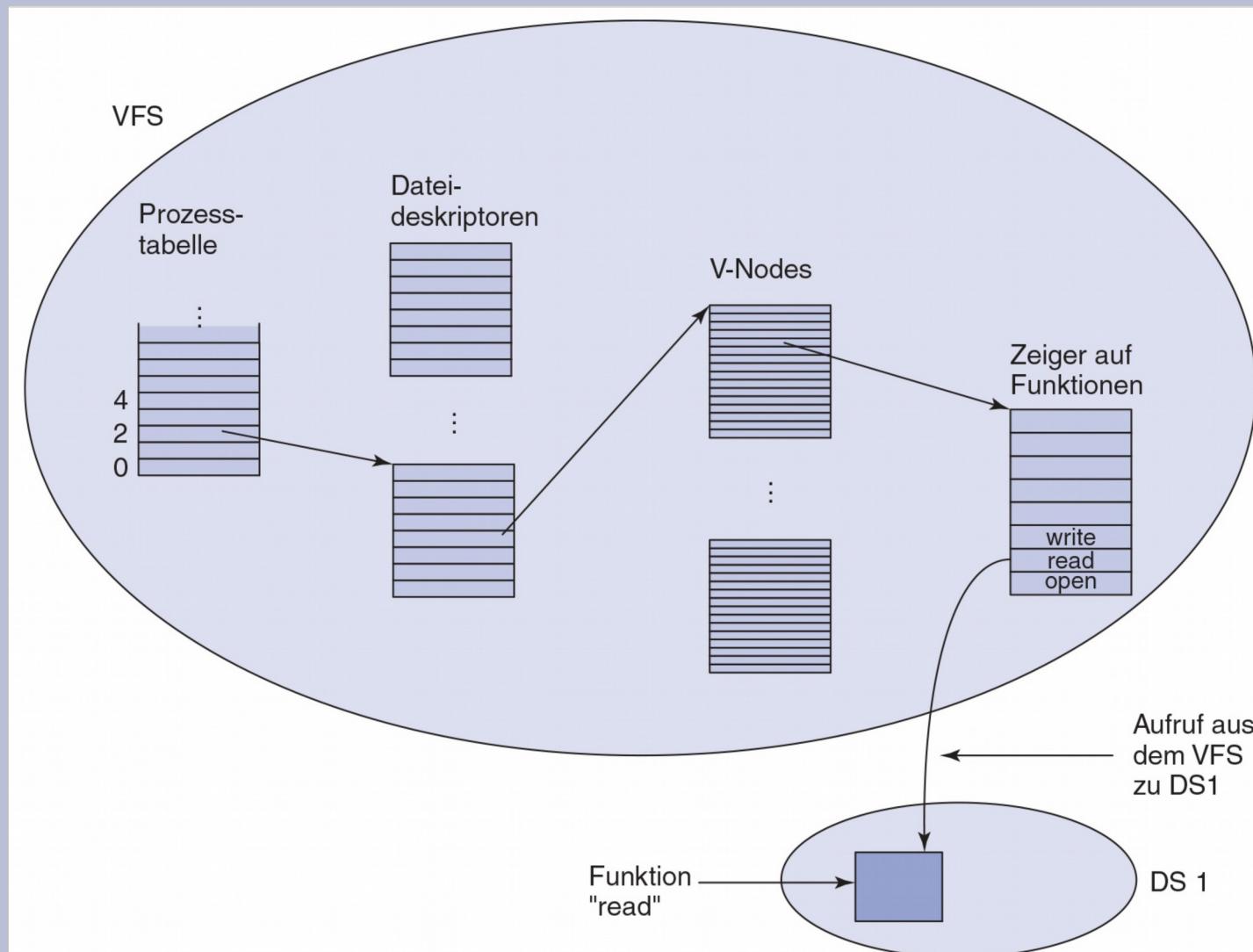
1. Datei aus ihrem Verzeichnis entfernen.
2. I-node an den Pool der freien I-nodes freigeben.
3. Alle Plattenblöcke zum Pool der freien Plattenblöcke zurückgeben.

# Virtuelle Dateisysteme (1)



**Abbildung 4.18:** Einordnung des virtuellen Dateisystems.

# Virtuelle Dateisysteme (2)



**Abbildung 4.19:** Ein vereinfachter Blick auf die Datenstrukturen und den Code, die vom VFS und dem konkreten Dateisystem benutzt werden, um einen read-Aufruf durchzuführen.

# 4.4 Dateisystemverwaltung und -optimierung

4.4.1 Plattenspeicherverwaltung

4.4.2 Sicherung von Dateisystemen

4.4.3 Konsistenz eines Dateisystems

4.4.4 Leistung eines Dateisystems

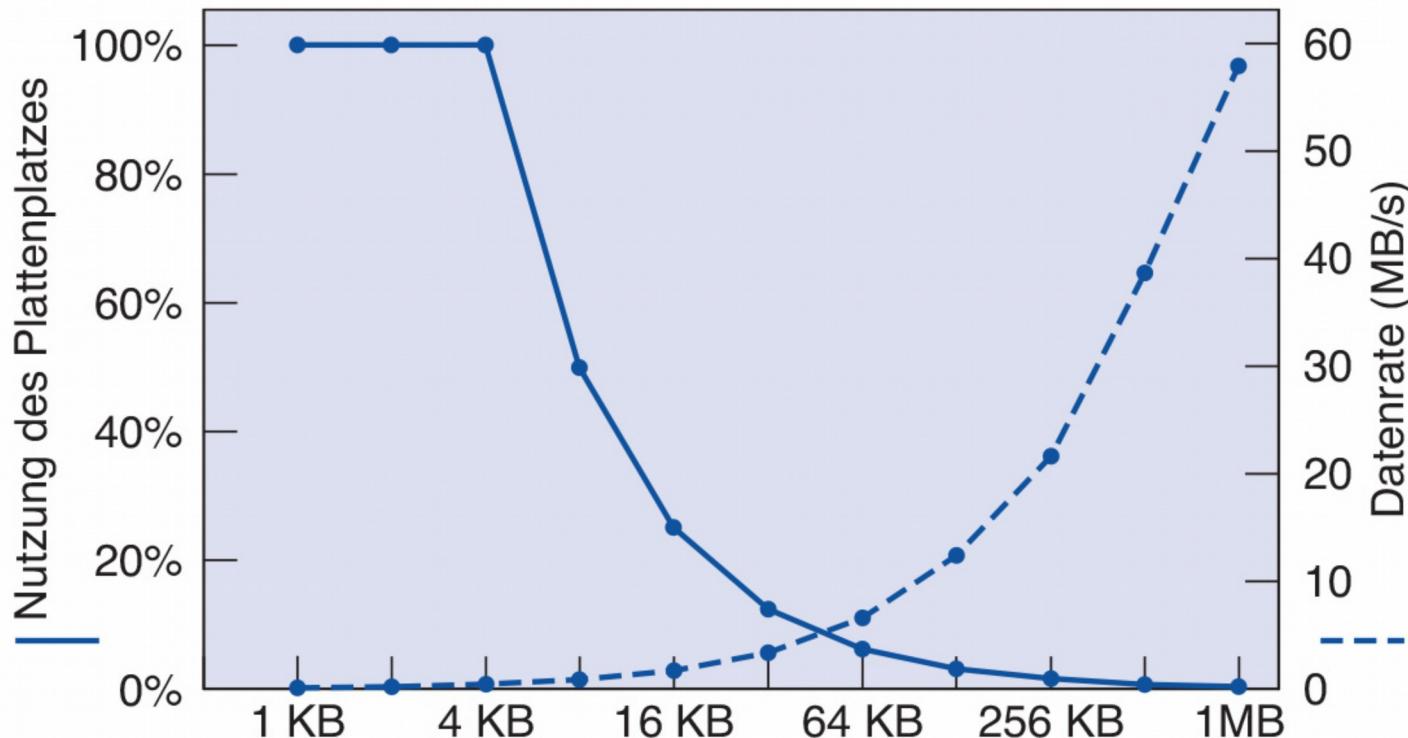
4.4.5 Defragmentierung von Plattspeicher

# Plattenspeicherverwaltung (1)

Länge	VU 1984	VU 2005	Web	Länge	VU 1984	VU 2005	Web
1	1,79	1,38	6,67	16 KB	92,53	78,92	86,79
2	1,88	1,53	7,67	32 KB	97,21	85,87	91,65
4	2,01	1,65	8,33	64 KB	99,18	90,84	94,80
8	2,31	1,80	11,30	128 KB	99,84	93,73	96,93
16	3,32	2,15	11,46	256 KB	99,96	96,12	98,48
32	5,13	3,15	12,33	512 KB	100,00	97,73	98,99
64	8,71	4,98	26,10	1 MB	100,00	98,87	99,62
128	14,73	8,03	28,49	2 MB	100,00	99,44	99,80
256	23,09	13,29	32,10	4 MB	100,00	99,71	99,87
512	34,44	20,62	39,94	8 MB	100,00	99,86	99,94
1 KB	48,05	30,91	47,82	16 MB	100,00	99,94	99,97
2 KB	60,87	46,09	59,44	32 MB	100,00	99,97	99,99
4 KB	75,31	59,13	70,64	64 MB	100,00	99,99	99,99
8 KB	84,97	69,96	79,69	128 MB	100,00	99,99	100,00

Abbildung 4.20: Prozentsätze von Dateien, die kleiner als eine gegebene Größe sind (in Byte).

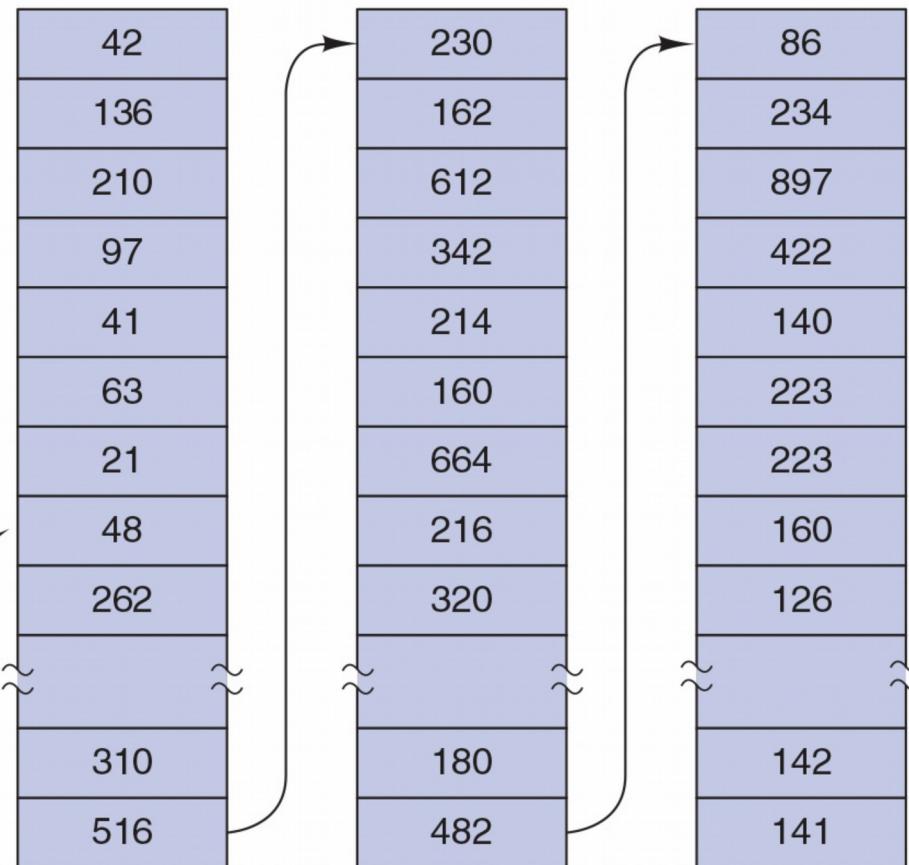
# Plattenspeicherverwaltung (2)



**Abbildung 4.21:** Die gestrichelte Linie (linke Ordinate) stellt die Datenrate einer Platte dar. Die durchgezogene Linie (rechte Ordinate) zeigt die Platzeffizienz der Platte. Alle Dateien sind 4 KB groß.

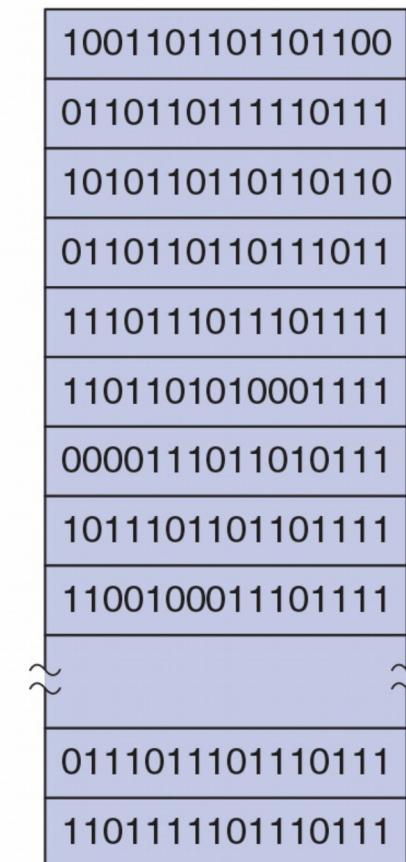
# Speicherung der Liste der freien Bereiche (1)

Freie Plattenblöcke: 16, 17, 18



Ein 1-KB-Plattenblock kann 256  
32-Bit-Plattenblocknummern speichern

a

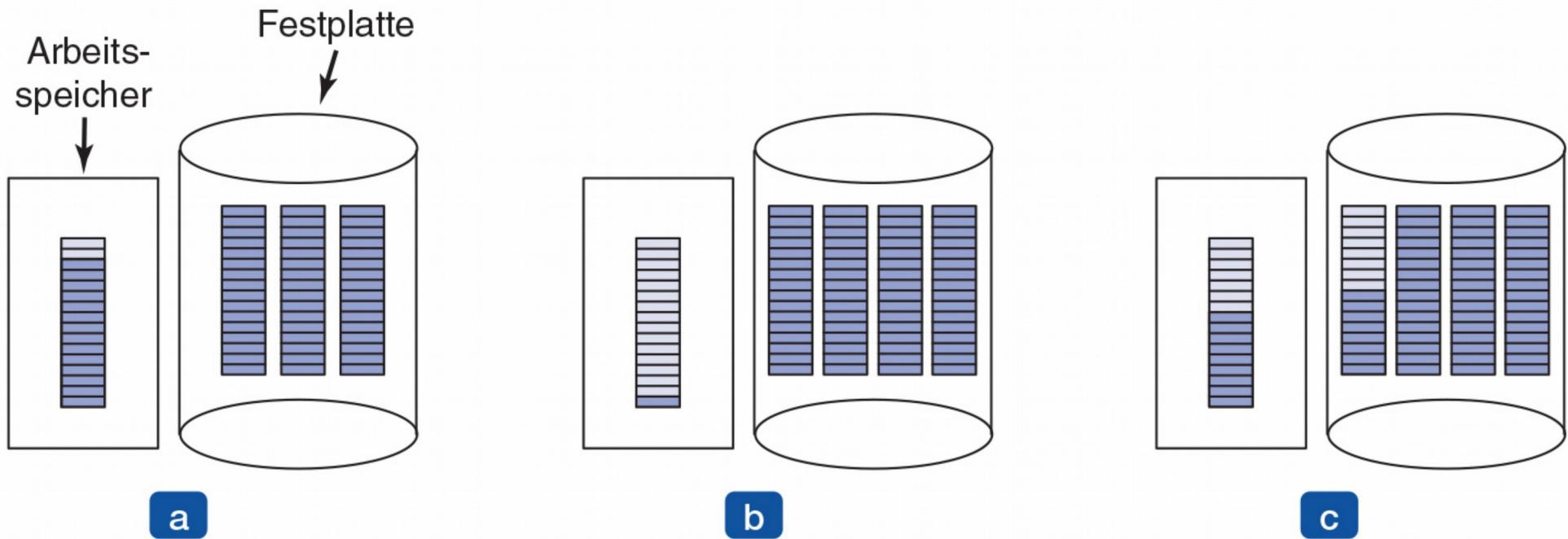


Eine Bitmap

b

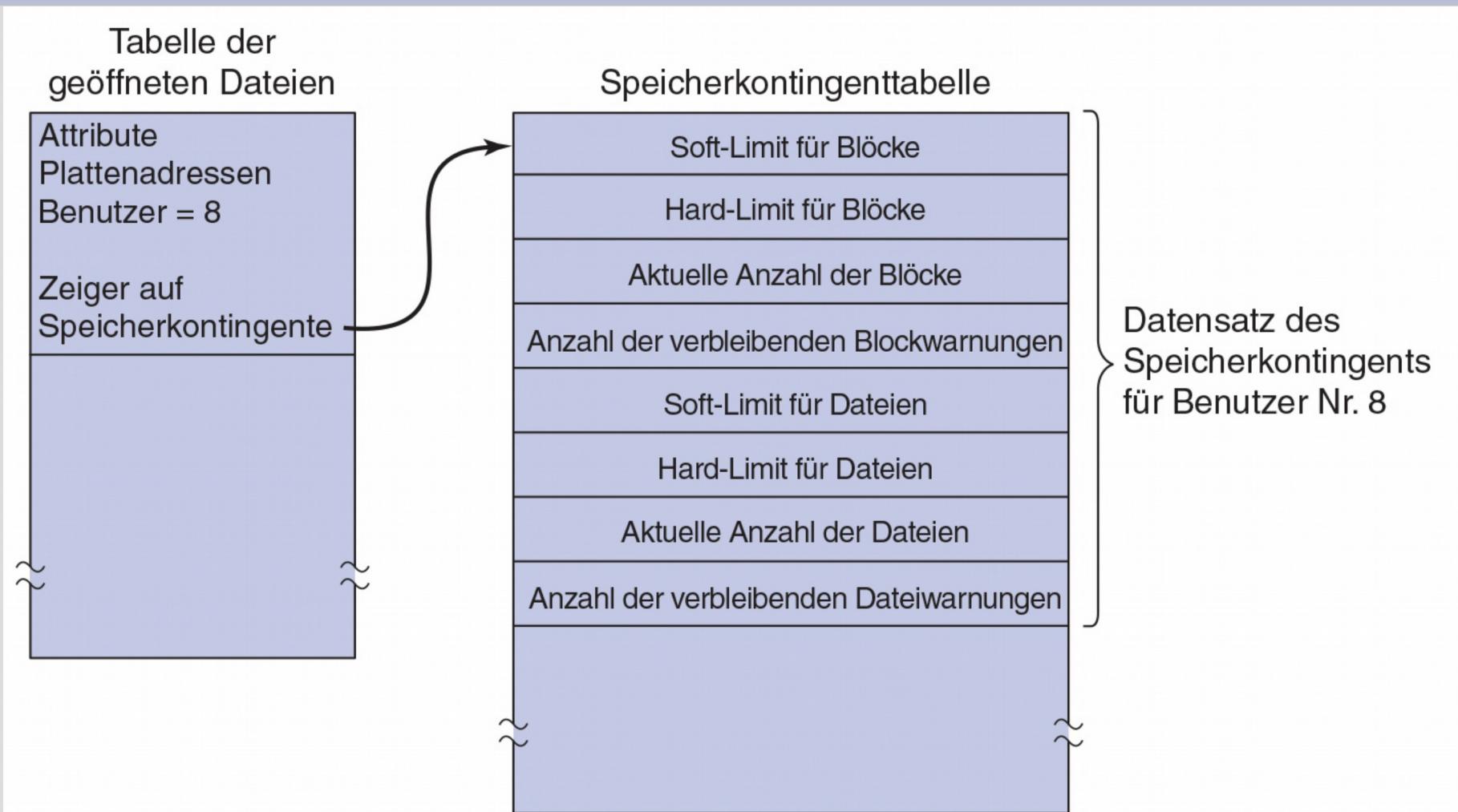
Abbildung 4.22: (a) Speicherung der Freibereichsliste als verkettete Liste, (b) Bitmap.

# Speicherung der Liste der freien Bereiche (2)



**Abbildung 4.23:** (a) Ein fast voller Zeigerblock im Arbeitsspeicher und drei Zeigerblöcke auf der Platte. (b) Situation nach der Freigabe einer 3-Block-Datei. (c) Eine alternative Strategie für die Verwaltung der drei freien Blöcke. Die dunkleren Einträge repräsentieren die Zeiger auf die freien Plattenblöcke.

# Disk Quotas



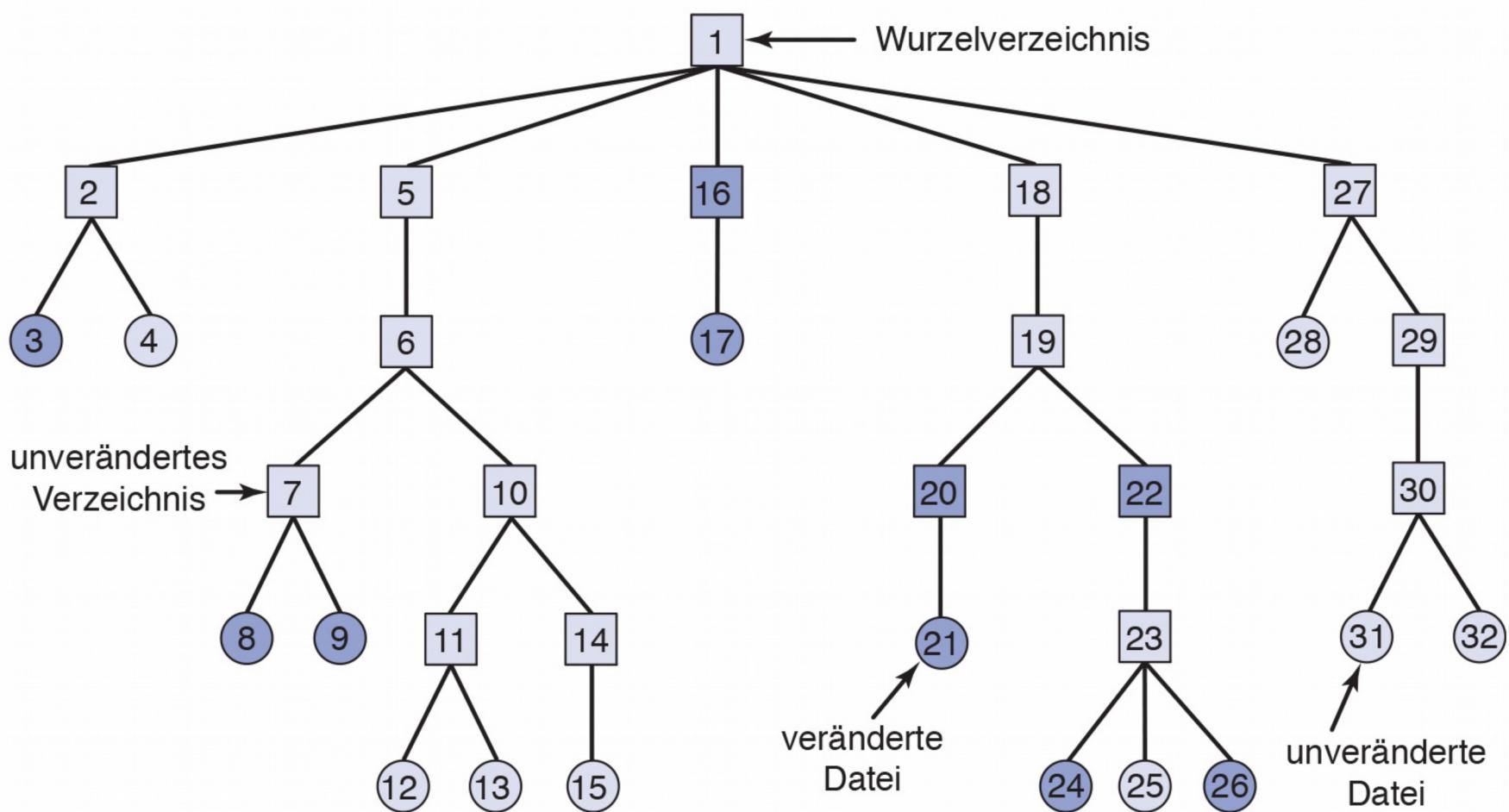
**Abbildung 4.24:** In einer Kontingenttabelle werden die Kontingente pro Benutzer festgehalten.

# Sicherung von Dateisystemen (1)

Sicherungen auf Band werden in der Regel gemacht, um eines von zwei möglichen Problemen zu behandeln:

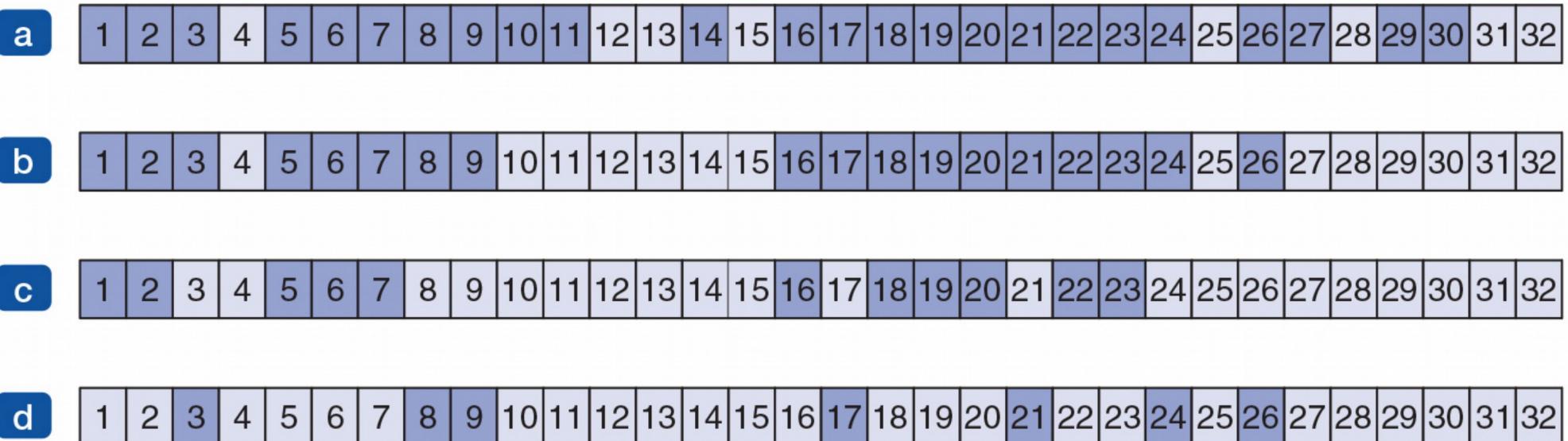
1. Wiederherstellung von einer Katastrophe.
2. Wiederherstellung von der eigenen Dummheit.

# Sicherung von Dateisystemen (2)



**Abbildung 4.25:** Ein Dateisystem, von dem eine Sicherungskopie angelegt werden muss. Die Rechtecke stellen die Verzeichnisse dar und die Kreise symbolisieren die Dateien. Die dunkler dargestellten Objekte wurden seit der letzten Sicherung modifiziert. Jedes Verzeichnis und jede Datei ist mit der zugehörigen I-Node-Nummer versehen.

# Sicherung von Dateisystemen (3)



**Abbildung 4.26:** Bitmaps, die vom Algorithmus für logische Sicherung verwendet werden.

Tanenbaum, A. S.; Bos, H.: Moderne  
Betriebssysteme. Pearson Studium 2016

# Konsistenz eines Dateisystems

Blocknummer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	0	0	1	1	1	0	0	0
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

a

Blocknummer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	0	0	1	1	1	0	0	0
0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1

c

Blocknummer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	1	0
0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1

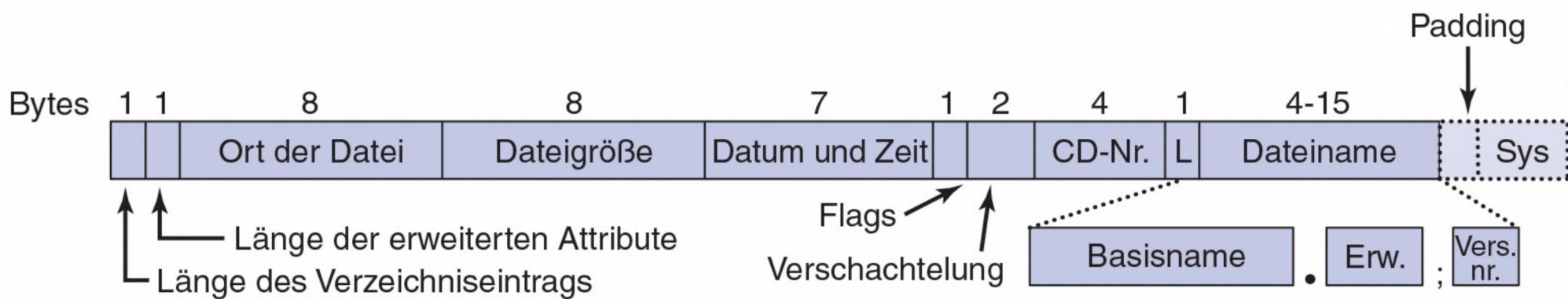
b

Blocknummer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	2	1	1	1	0	0	1	1	1	1	0
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

d

**Abbildung 4.27:** Zustände von Dateisystemen: (a) konsistent, (b) fehlender Block, (c) doppelt vorhandener Block in der Freibereichsliste, (d) doppelt vorhandener Datenblock.

# Leistung eines Dateisystems (1)



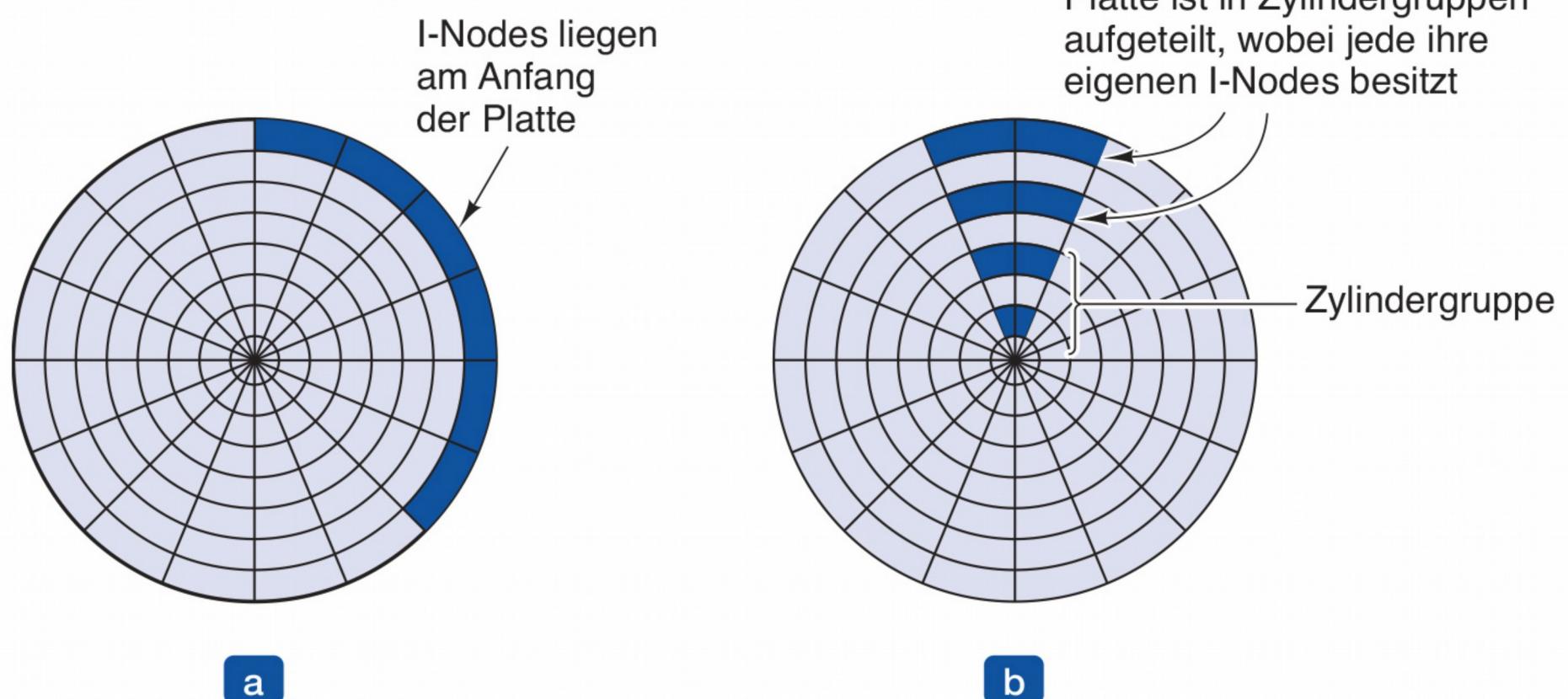
**Abbildung 4.28:** Die Datenstrukturen des Puffer-Cache.

# Leistung eines Dateisystems (2)

- Einige Blöcke werden innerhalb eines kurzen Intervalls selten zweimal referenziert.
- Dies führt zu einem modifizierten LRU-Schema\*, das zwei Faktoren berücksichtigt:
  - Wird der Block wahrscheinlich bald wieder benötigt?
  - Ist der Block für die Konsistenz des Dateisystems wesentlich?

\* Least Recently Used

# Reduktion der Bewegungen der Schreib-Lese-Köpfe



**Abbildung 4.29:** (a) Die Platzierung der I-Nodes am Anfang der Platte. (b) Die Platte aufgeteilt in Zylindergruppen mit jeweils eigenen Blöcken und I-Nodes.

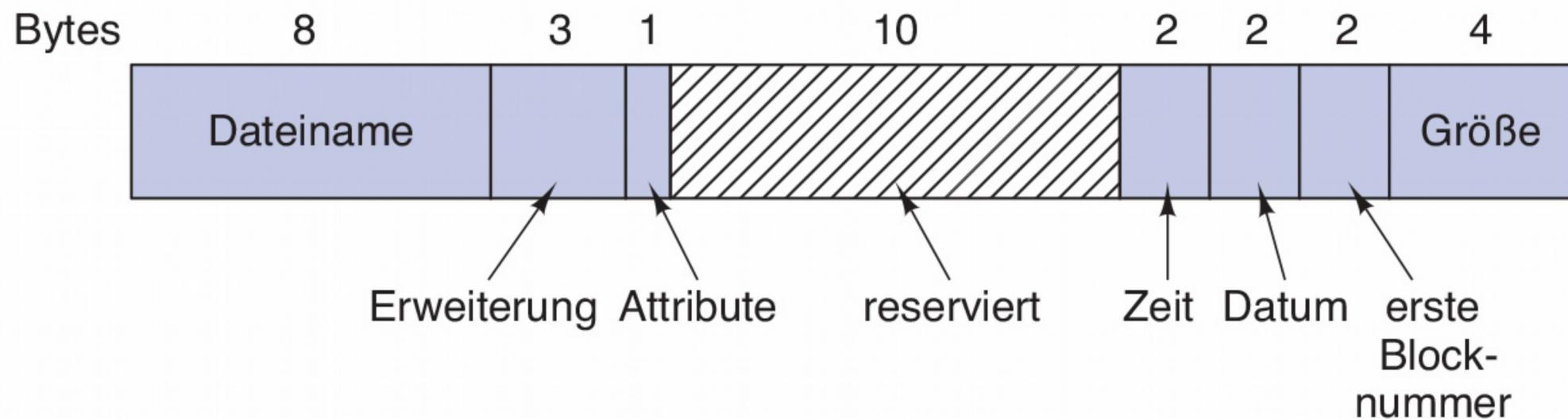
# 4.5 Beispiele von Dateisystemen

4.5.1 Das MS-DOS Dateisystem

4.5.2 Das UNIX-V7-Dateisystem

4.5.3 CD-ROM-Dateisysteme

# Das MS-DOS Dateisystem (1)



**Abbildung 4.30:** Der MS-DOS-Verzeichniseintrag.

# Das MS-DOS Dateisystem (2)

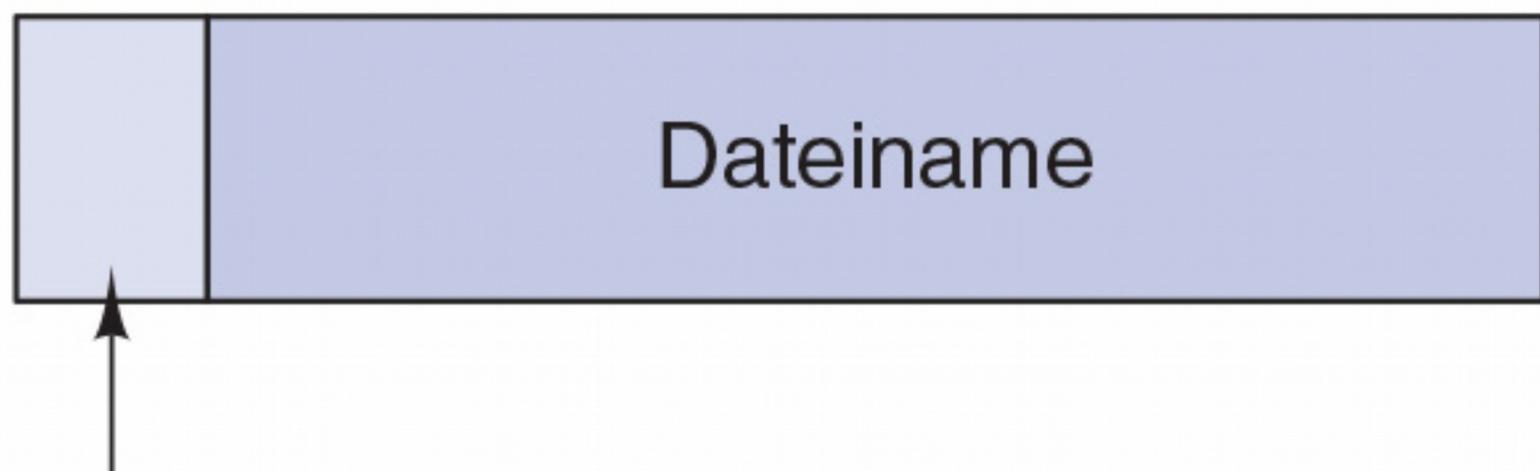
Blockgröße	FAT-12	FAT-16	FAT-32
0,5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

**Abbildung 4.31:** Die maximalen Partitionsgrößen für verschiedene Blockgrößen, die leeren Felder zeigen verbotene Kombinationen an.

# Das UNIX-V7-Dateisystem (1)

Bytes    2

14



I-Node-  
Nummer

**Abbildung 4.32:** UNIX-V7-Dateieintrag.

# Das UNIX-V7-Dateisystem (2)

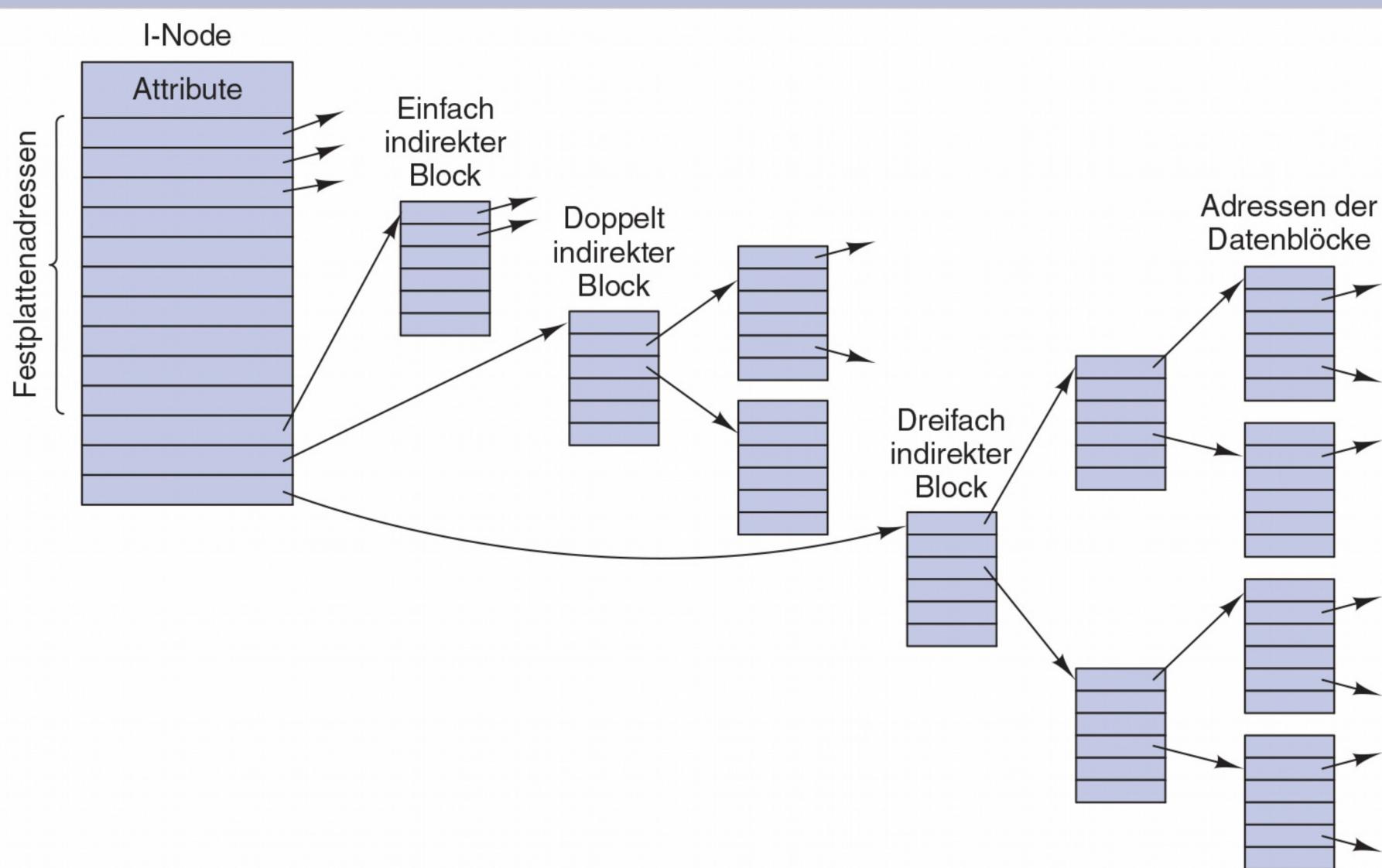
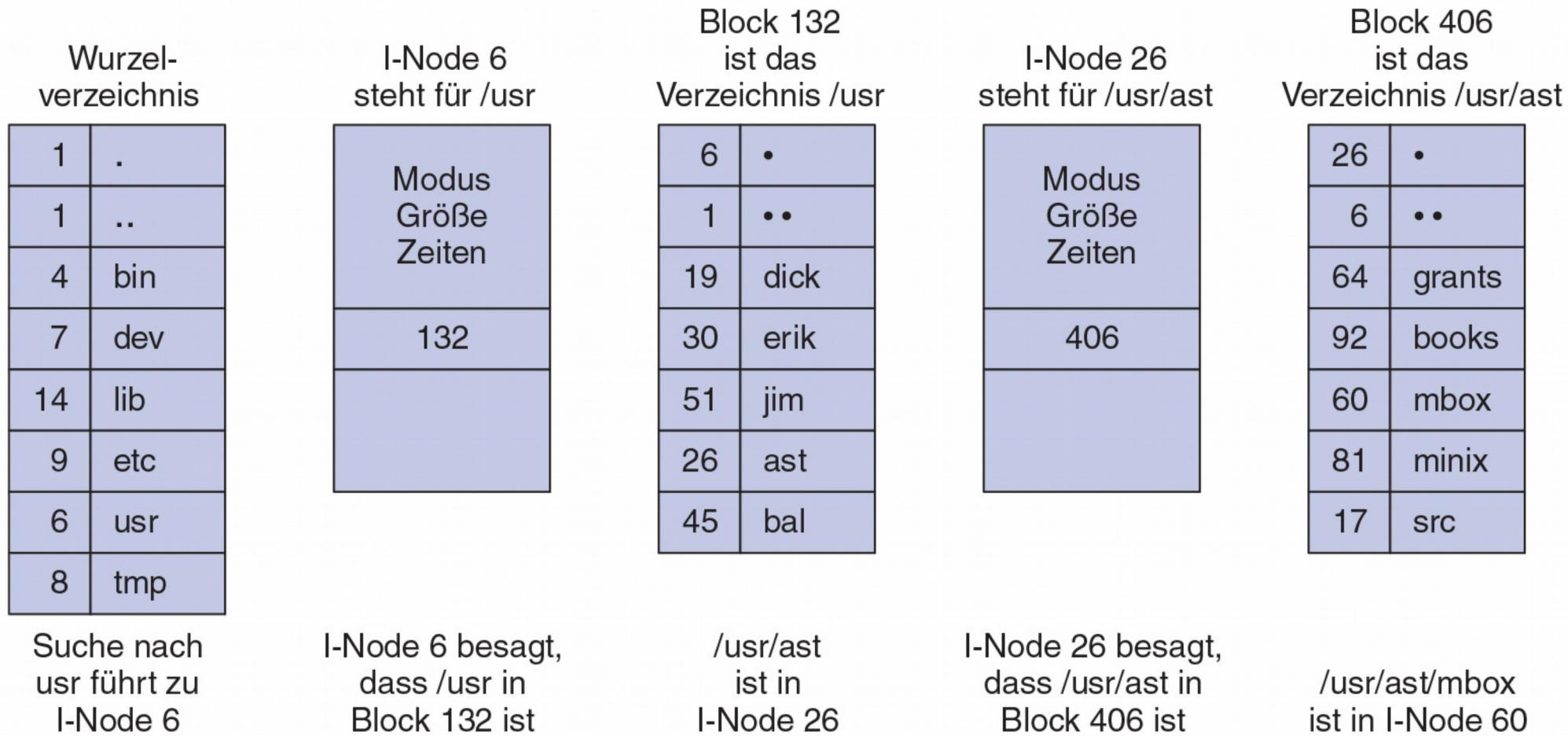


Abbildung 4.33: I-Node unter UNIX.

# Das UNIX-V7-Dateisystem (3)



**Abbildung 4.34:** Schritte, um /usr/ast/mbox zu finden.

# CD-ROM-Dateisysteme

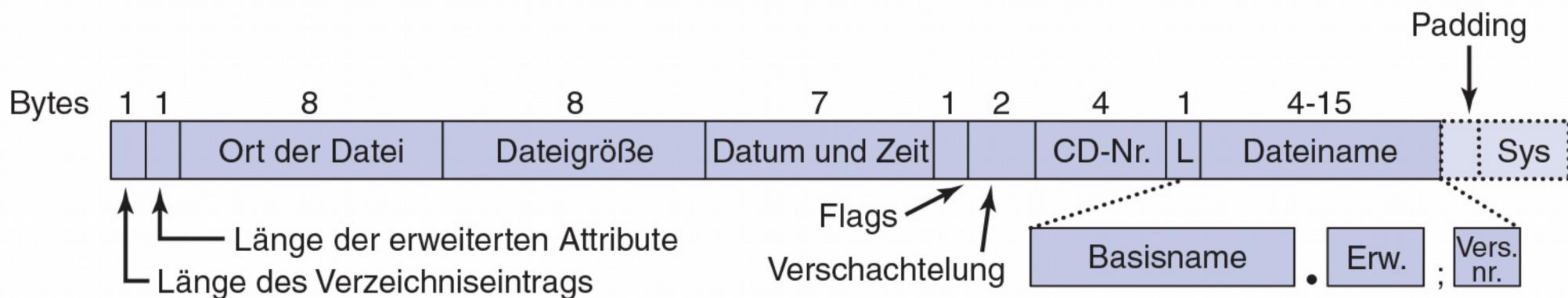


Abbildung 4.35: Der ISO-9660-Verzeichniseintrag.

# Rock Ridge Extensions

1. PX - POSIX attributes.
2. PN - Major and minor device numbers.
3. SL - Symbolic link.
4. NM - Alternative name.
5. CL - Child location.
6. PL - Parent location.
7. RE - Relocation.
8. TF - Time stamps.

# Joliet Extensions

1. Long file names.
2. Unicode character set.
3. Directory nesting deeper than eight levels.
4. Directory names with extensions