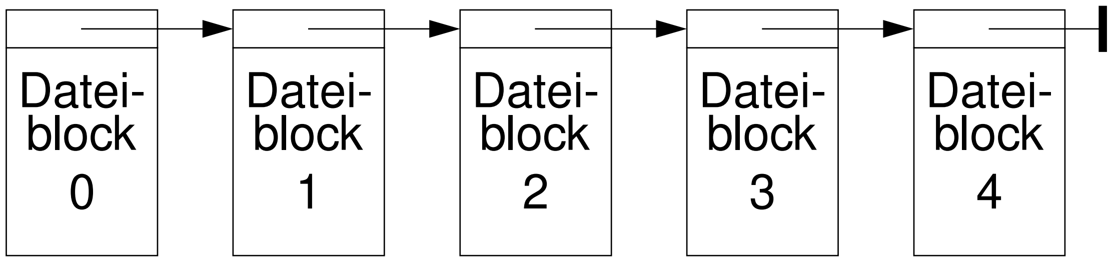
6 Implementationen von Dateien

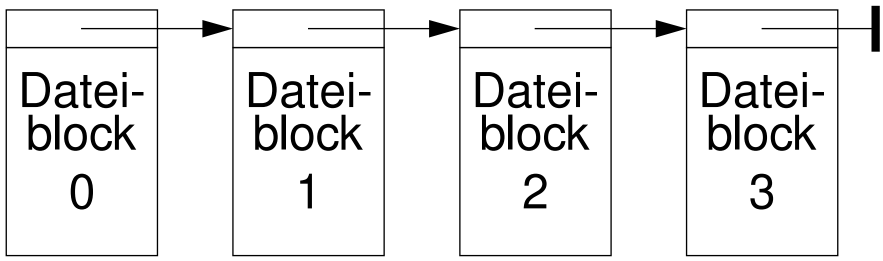
- ❏ Speicherung
  - zusammenhängend
  - verkettet
  - indiziert
  - baumsequentiell
- ❏ Dateideskriptoren (I-nodes)
- ❏ Implementierung von Katalogen:
  - CP/M, MS-DOS
  - UNIX
- ❏ Shared files
- ❏ Verwaltung des Externspeichers:
  - Blockgröße
  - Freispeicherverwaltung
  - Quotas
- ❏ Zuverlässigkeit
  - Bad blocks
  - Backup
- ❏ Konsistenz

Datei A



phys. Block 4 7 2 10 12

Datei B

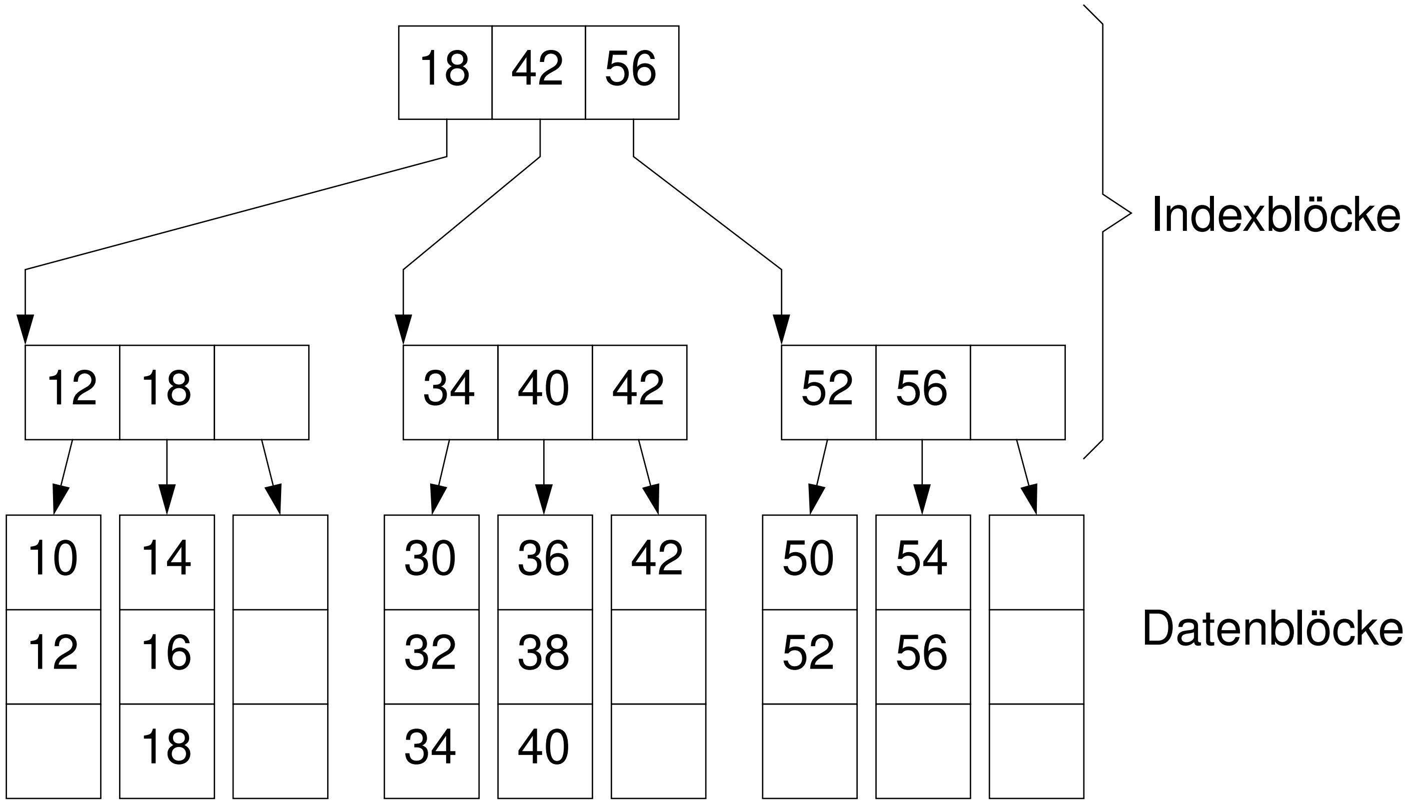


phys. Block 6 3 11 14

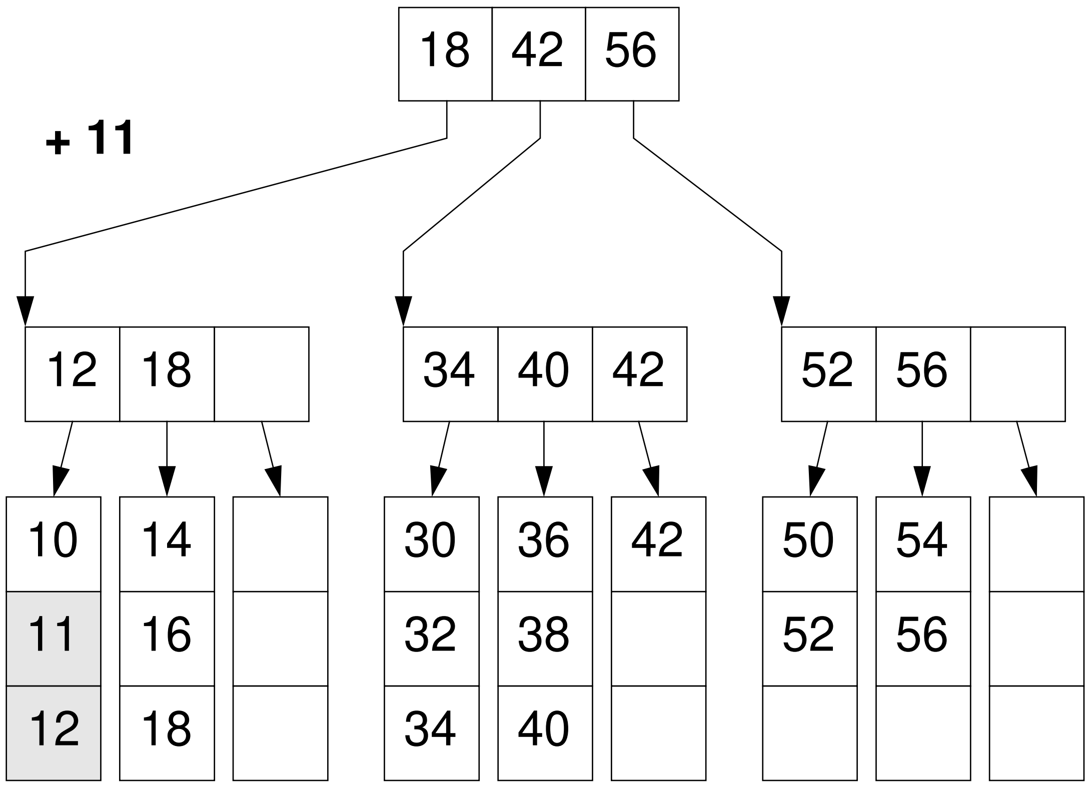
Externspeicher

			Dateiblock A, 2	Dateiblock B, 1	Dateiblock A, 0		Dateiblock B, 0	Dateiblock A, 1
phys. Block	0	1	2	3	4	5	6	7

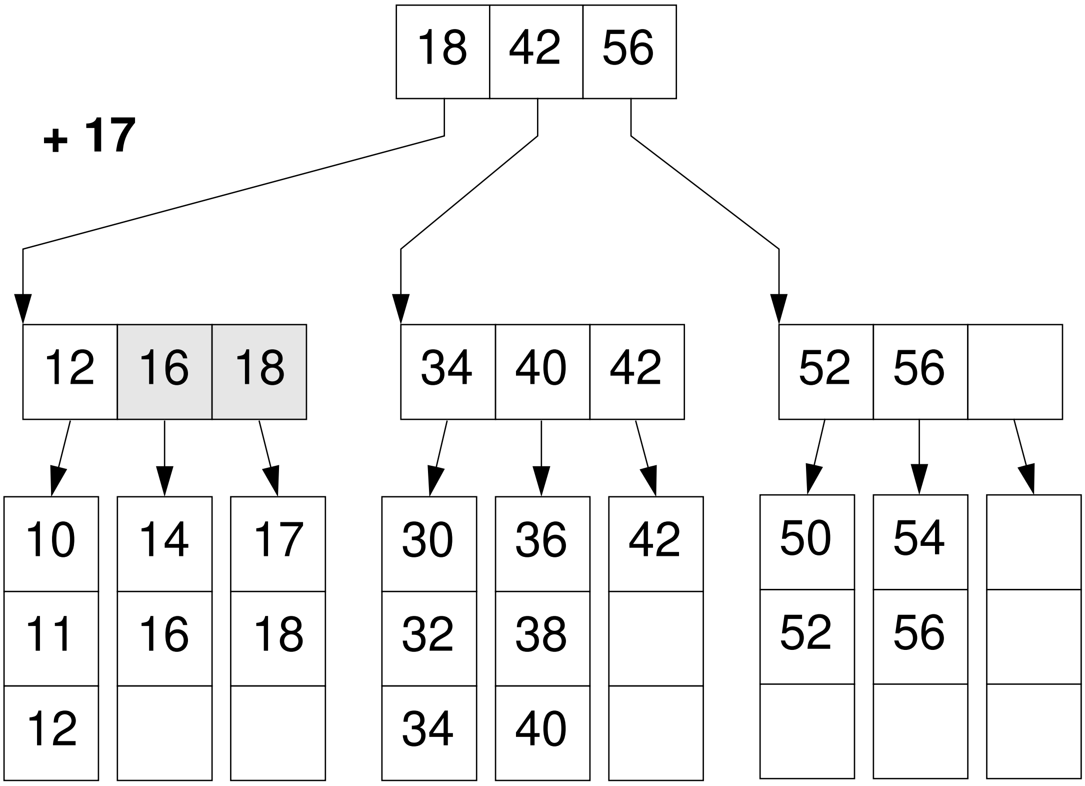
			Dateiblock A, 3	Dateiblock B, 2	Dateiblock A, 4		Dateiblock B, 3	
phys. Block	8	9	10	11	12	13	14	15


18 42 56
Indexblöcke
12 18
34 40 42
52 56
10 12
14 16 18
30 32 34
36 38 40
42
50 52
54 56
Datenblöcke

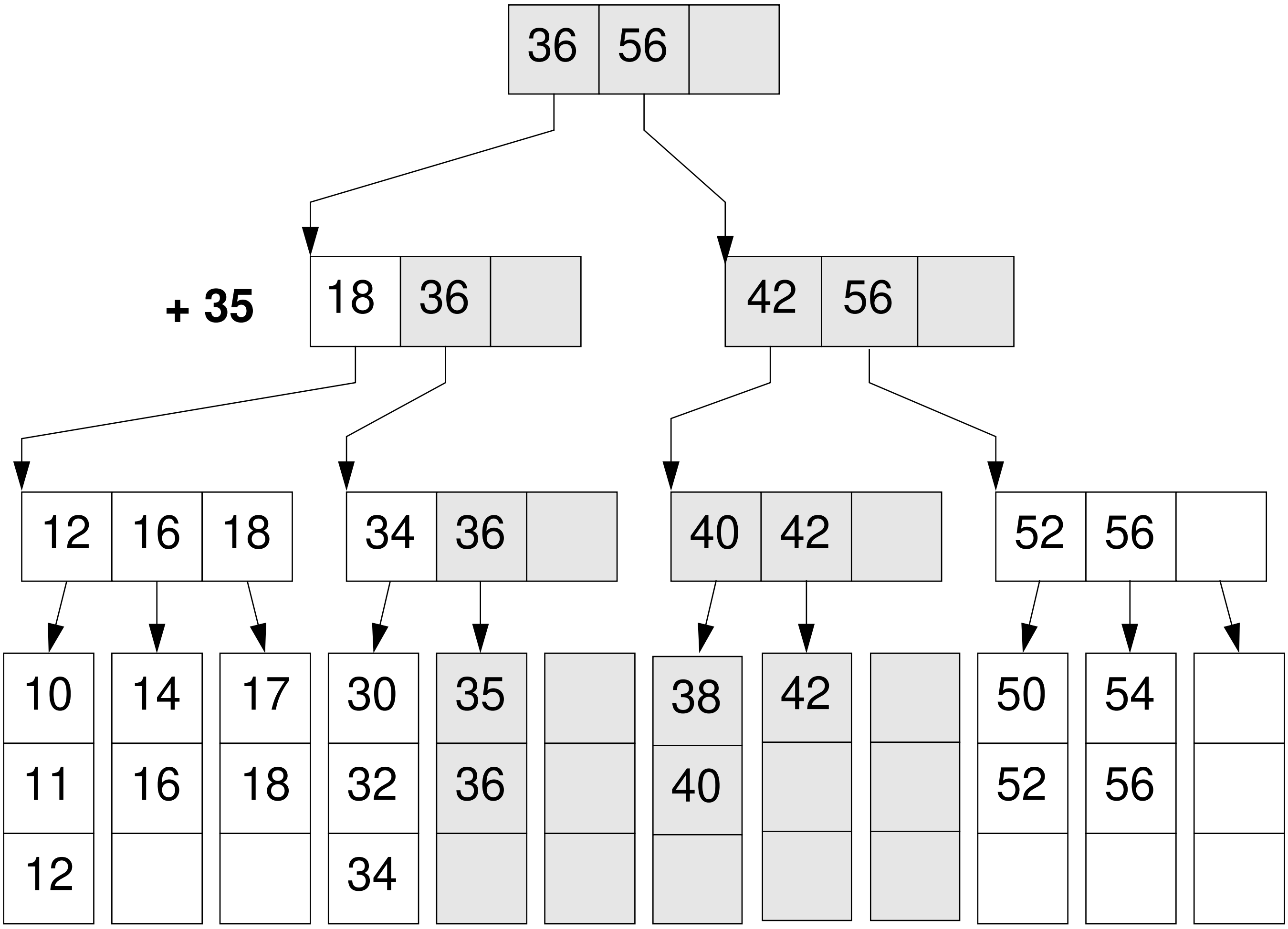
+ 11


18 42 56
12 18
34 40 42
52 56
10 11 12
14 16 18
30 32 34
36 38 40
42
50 52
54 56

+ 17


18 42 56
12 16 18
34 40 42
52 56
10 11 12
14 16
17 18
30 32 34
36 38 40
42
50 52
54 56

+ 35


36 56
18 36
42 56
12 16 18
34 36
40 42
52 56
10 11 12
14 16
17 18
30 32 34
35 36
38 40
42
50 52
54 56

Unix-Dateisystem

Funktionalität

- Abstraktionen für Benutzersicht:

  1. Pfade, Dateinamen
  2. Dateibaum verdeckt mehrere Platten (bzw. Partitionen)
  3. Dateien = unstrukturierte Byteströme
  4. *special files* fast wie reguläre Dateien zugreifbar

- diese Abstraktionen müssen vom Dateisystem auf konkrete Objekte abgebildet werden

- den obigen Abstraktionen entsprechen etwa folgende Objekte:

  1. Plattenpositionen, Adressen von Gerätecontrollern, etc.
  2. wie 1.
  3. Platten verwalten Sektoren, Spuren und Zylinder
  4. unterschiedliche Gerätetreiber werden für E/A-Operationen auf verschiedenen Peripheriegeräten benötigt

Directories (Kataloge)

- wichtigster Bestandteil zum Aufbau eines hierarchischen Dateisystems

- Funktion: Strukturierung des Namensraums

- *Directories* enthalten Verweise auf Inodes (von Dateien oder weiteren Directories)

Aufbau eines Directory

- unterschiedlich in Unix System V (bis V.3) und Berkeley-Unix

- SystemV-Dateisystem: Einträge sind feste Strukturen
  - 2 Byte *inode-number*
  - 14 Zeichen Dateiname

- Berkeley-Dateisystem: Einträge sind variabel
  - Größe des Eintrags
  - Länge des Dateinamens (max. 255)
  - *inode-number*
  - Dateiname

Inodes in Unix

- ein Inode enthält alle Informationen, die zur Verwaltung einer Datei und zum Auffinden ihres Inhalts benötigt werden

- beim Öffnen einer Datei werden die Inode-Daten von der Platte ***(disk inode)*** in die Inode-Tabelle des Systemkerns **(*in-core inode)*** gelesen

- ein Inode (*disk inode)* enthält folgende Einträge

  - Dateityp

  - Eigentümer und Gruppenzugehörigkeit

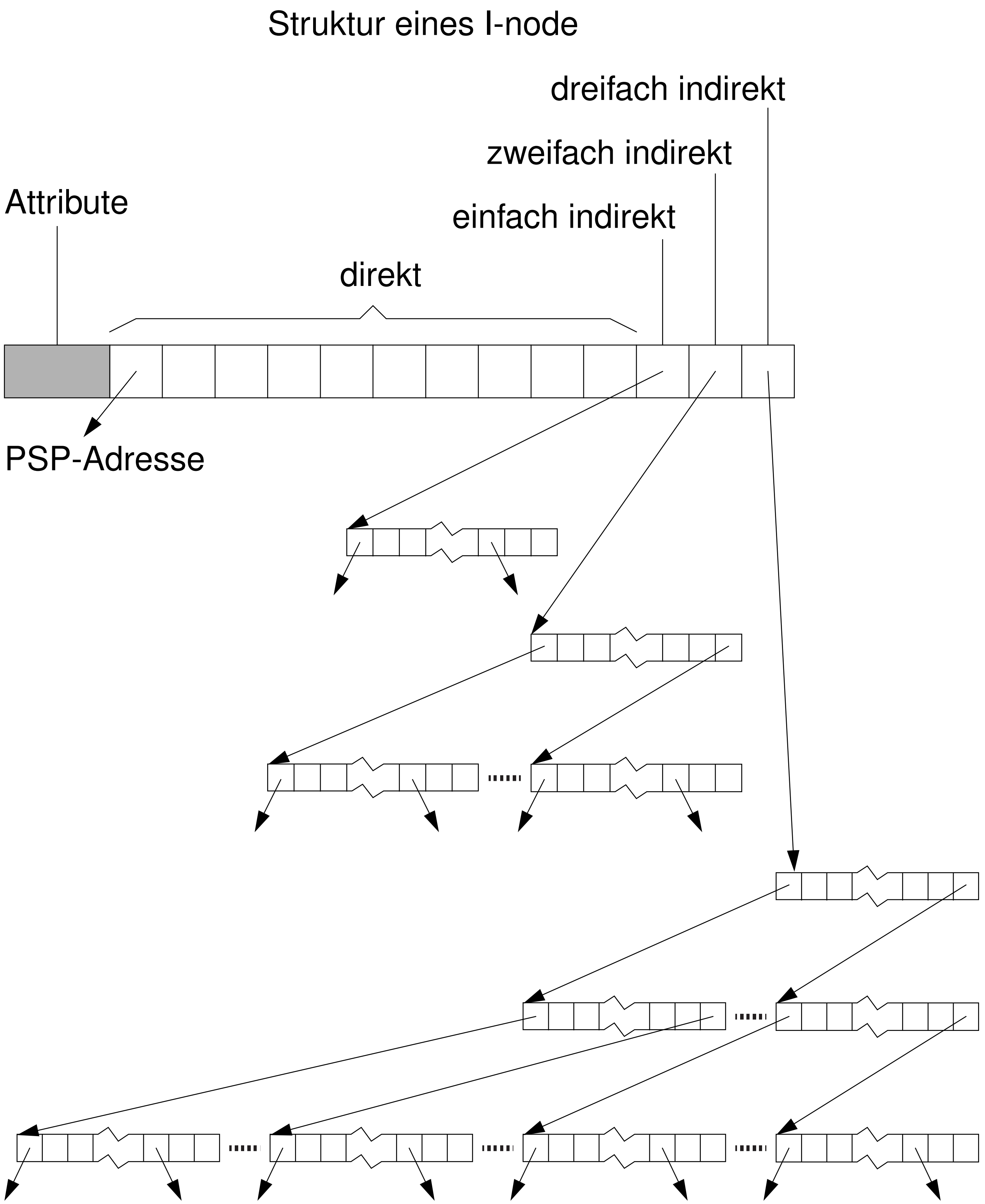
  - Zugriffsrechte

  - Zugriffszeiten
    – letzte Änderung (*mtime*) (ls -l)
    – letzter Zugriff (*atime*) (ls -lu)
    – letzte Änderung des Inodes (*ctime*) (ls -lc)

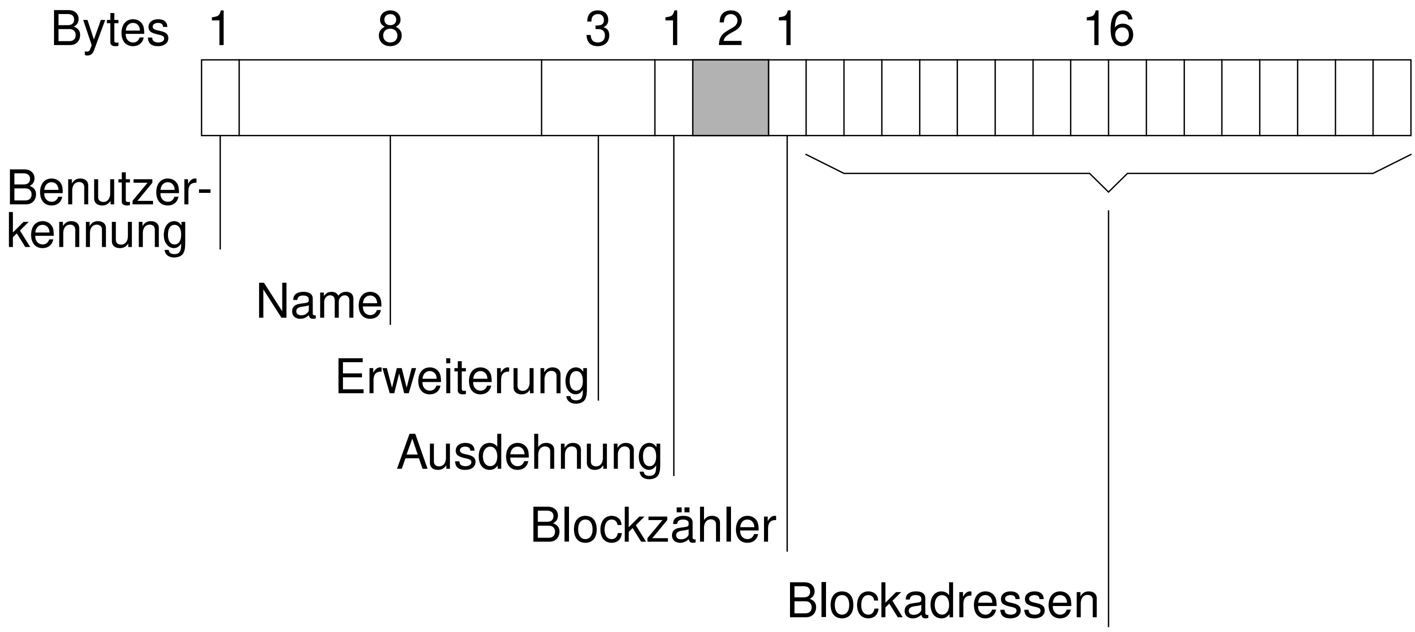
  - Anzahl der (hard) Links auf die Datei

  - Dateigröße (in Bytes)

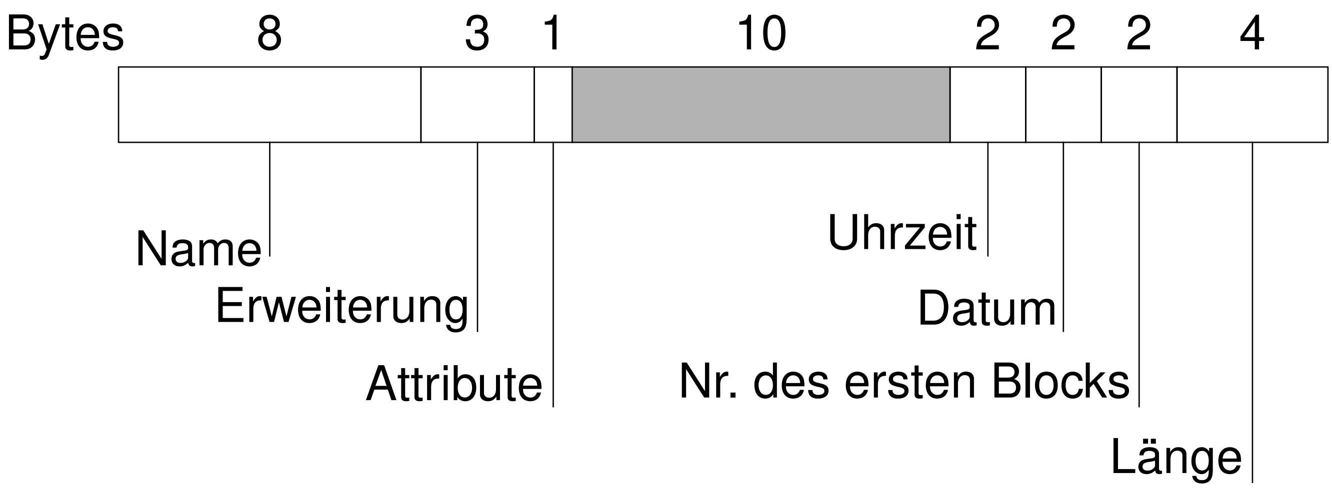
  - Adressen für Datenblöcke auf der Platte:
    (10 direkte Adressen + 3 Adressen für indirekte Adressierung)


Struktur eines I-node
dreifach indirekt
zweifach indirekt
Attribute
einfach indirekt
direkt
PSP-Adresse

CP/M


Bytes
1
8
3
1
2
1
16
Benutzer-
kennung
Name
Erweiterung
Ausdehnung
Blockzähler
Blockadressen

MS-DOS


Bytes
8
3
1
10
2
2
2
4
Name
Erweiterung
Attribute
Uhrzeit
Datum
Nr. des ersten Blocks
Länge

UNIX

Katalogwurzel (root directory)

2	*
2	**
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

I-node 6 beschreibt /usr

Modus Größe Zeitan- gaben
132

Block 132 ist Katalog /usr

6	*
2	**
19	dick
30	erik
151	jim
26	ast
45	bal

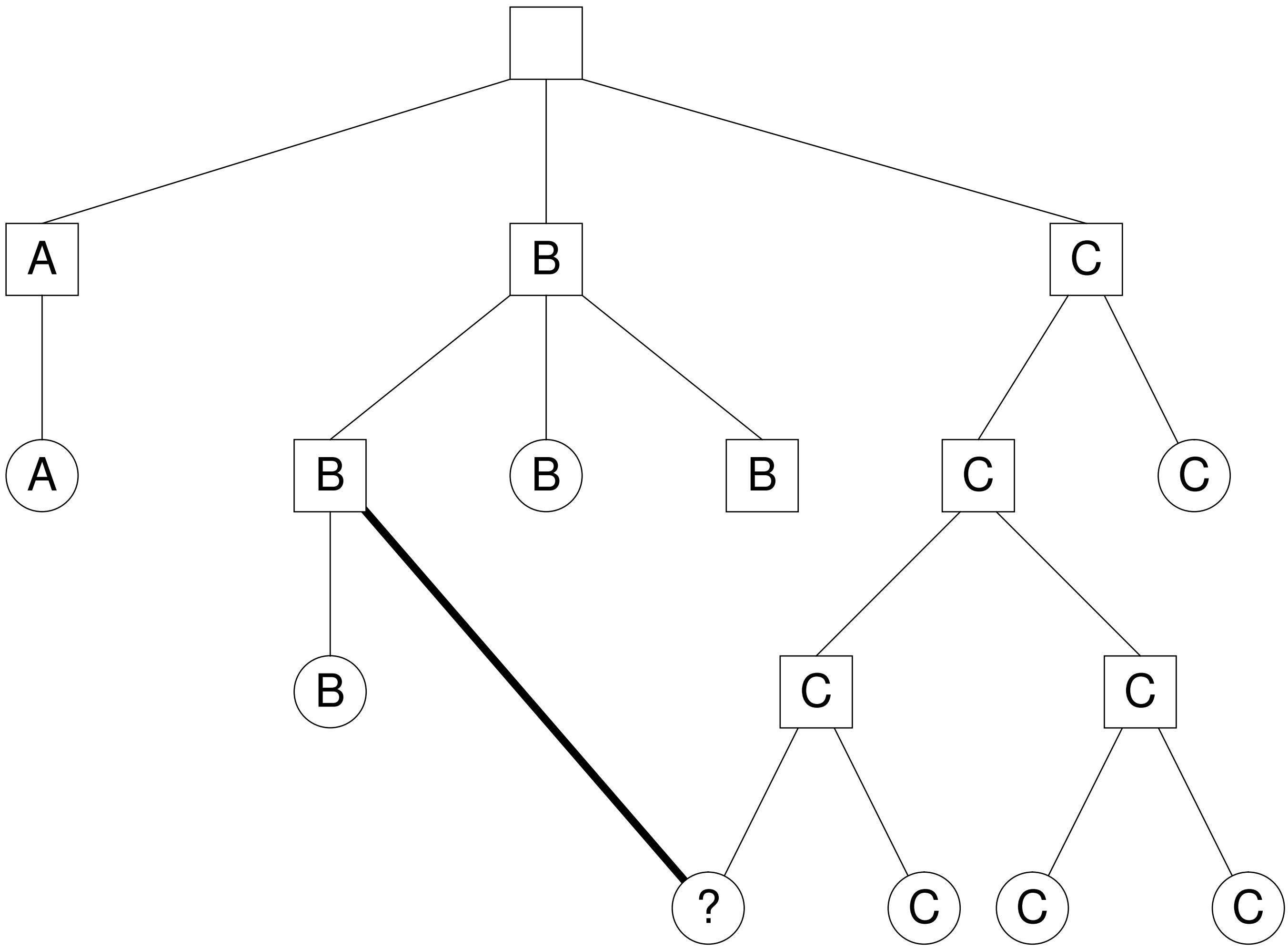
I-node 26 beschreibt /usr/ast

Modus Größe Zeitan- gaben
406

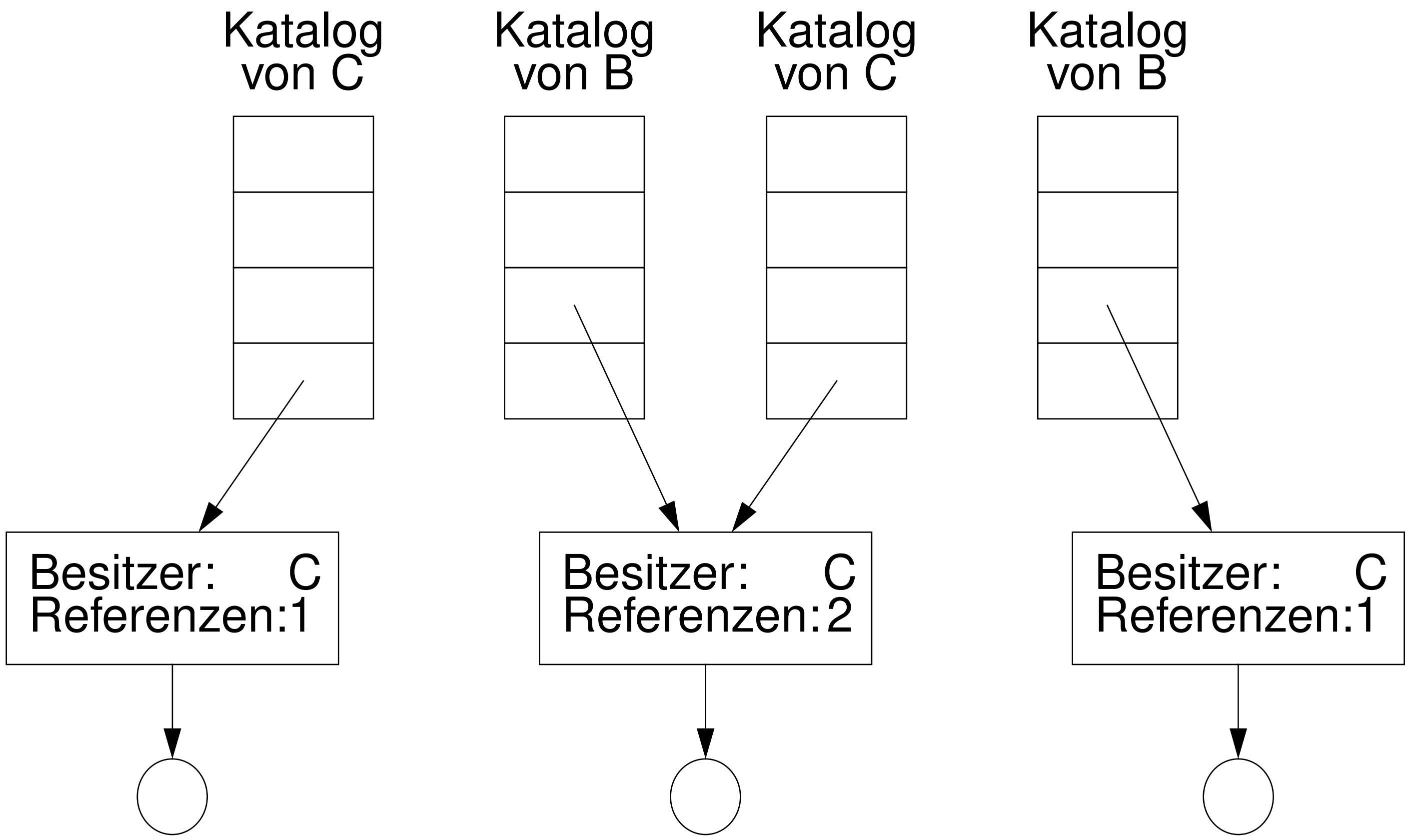
Block 406 ist Katalog /usr/ast

26	*
6	**
64	grants
92	books
60	mbox
81	linux
17	src

Mehrfach benannte Dateien



Mehrfaches Binden



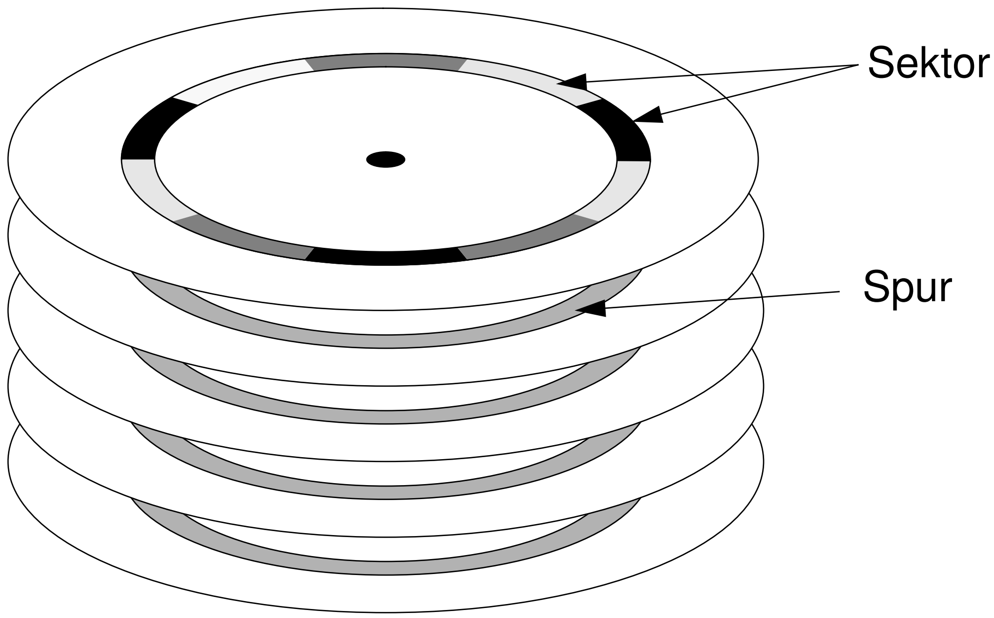
Plattenspeicher

Organisation einer Disk - Laufwerk

- typische Winchester-Festplatten bestehen aus mehreren, übereinander angeordneten Magnetplatten und einem Schreib-/ Lese-Kopf pro Plattenoberfläche

- Begriffe:

Spur *(Track)*	Umlaufbahn auf einer Diskoberfläche
Sektor	Teil einer Spur



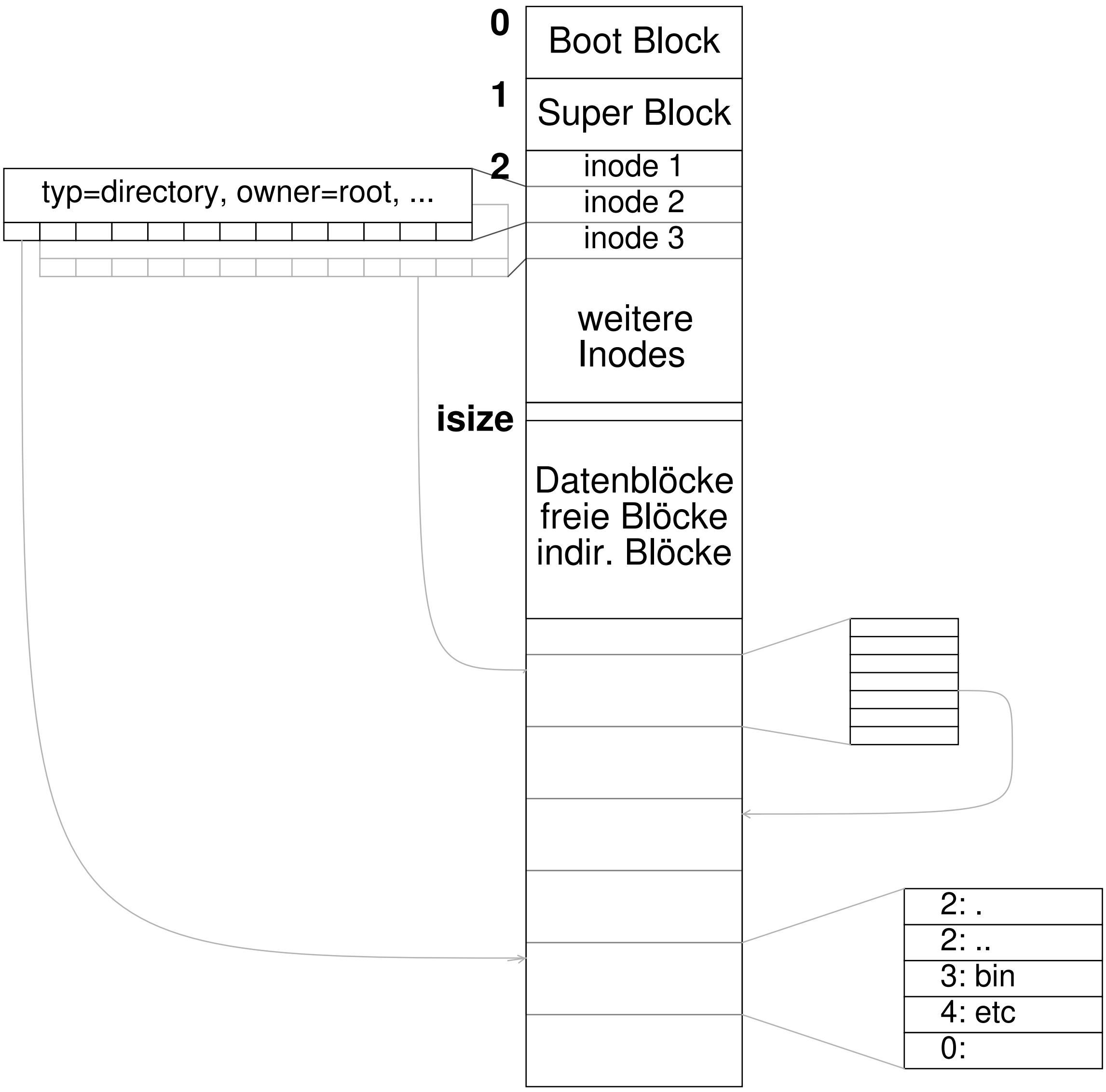
Zylinder	Menge der Spuren mit gleichem Radius über alle Oberflächen

Aufbau eines Dateisystems auf Platte

- Unterschiede zwischen Berkeley fast-file-system und herkömmlichen UNIX SystemV-Dateisystemen!

SystemV-Dateisystem

- (Blockgröße 512 oder 1024 Byte)



Super-Block

- Der *Super-Block* enthält die Verwaltungsinformationen für ein Dateisystem
  - Größe
  - Anzahl der freien Blöcke
  - Liste freier Blöcke *(free block list)*
  - Zeiger auf nächsten freien Block in der *free block list*
  - Größe der Inode-Liste
  - Anzahl freier Inodes
  - Liste freier Inodes *(free inode list)*
  - Zeiger auf ersten freien Inode
  - Lock-Felder für *free block* und *free inode list*
  - *modified flag* für Super-Block

Berkeley fast file system

- für UNIX 4.2bsd entwickelt - effizienter als SystemV-Dateisysteme

- Blockgröße minimal 4k, meist 8k

  ⇒ über zweifach indirekte Blöcke können bis zu 64 GB adressiert werden

- Directories anders aufgebaut - Dateinamen bis zu 255 Zeichen lang

- Dateisystem in ***cylinder groups*** aufgeteilt (*cylinder group* = Gruppe von aufeinanderfolgenden Zylindern auf Platte, häufig 16)

- pro *cylinder group* werden folgende Daten gehalten:

  - Backup-Kopie des Superblocks (da Verlust besonders schlimm)

  - Inodes

  - Bitmap für freie Blöcke (statt *free list*)

- Um Verschnitt gering zu halten, können Blöcke in 2, 4 oder 8 ***Fragmente*** aufgeteilt werden, die einzeln adressierbar sind

- Eine Datei wird - wenn möglich - in einer *cylinder group* gehalten

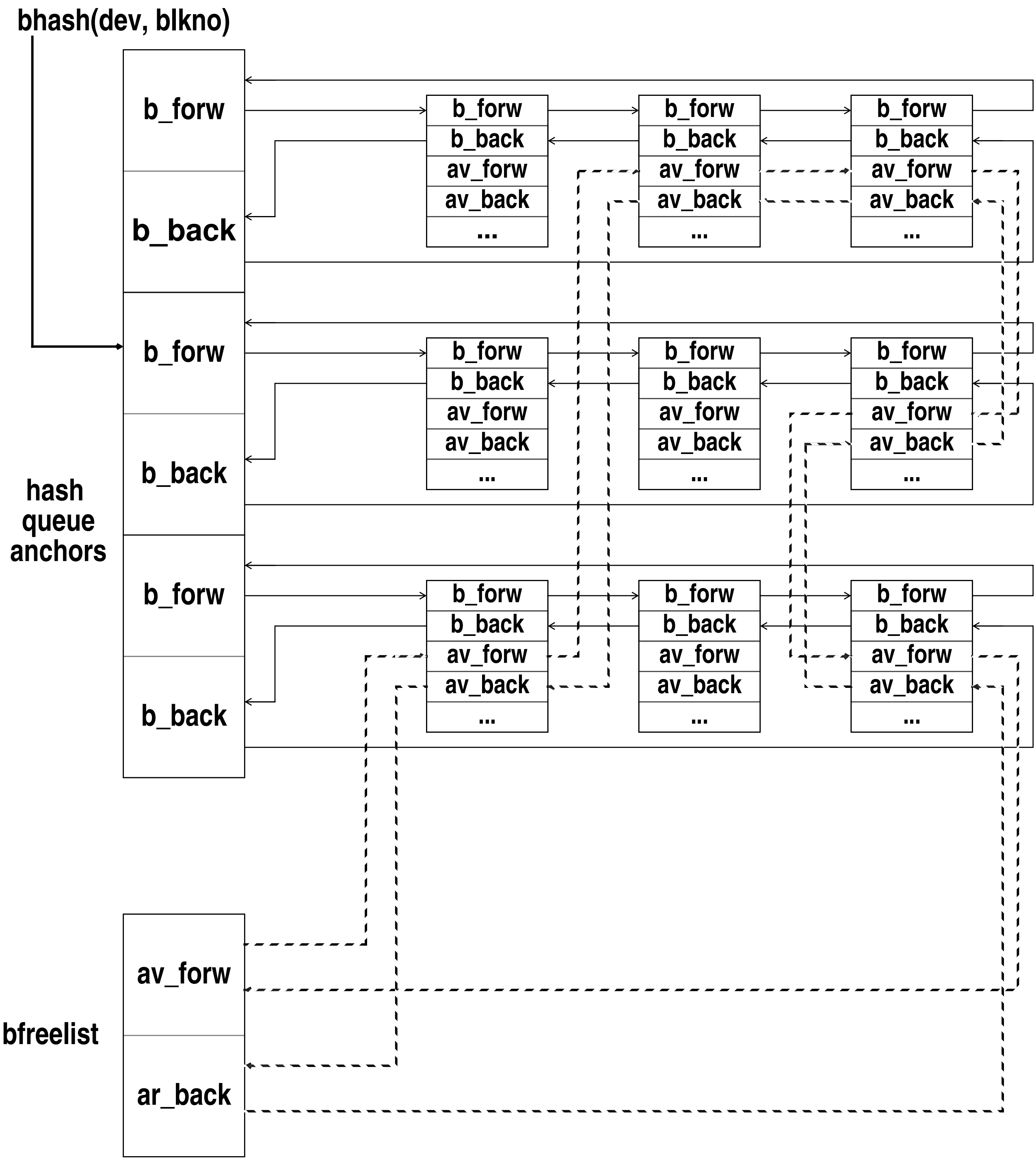
Block-Puffer Cache

Verwaltungsstrategien

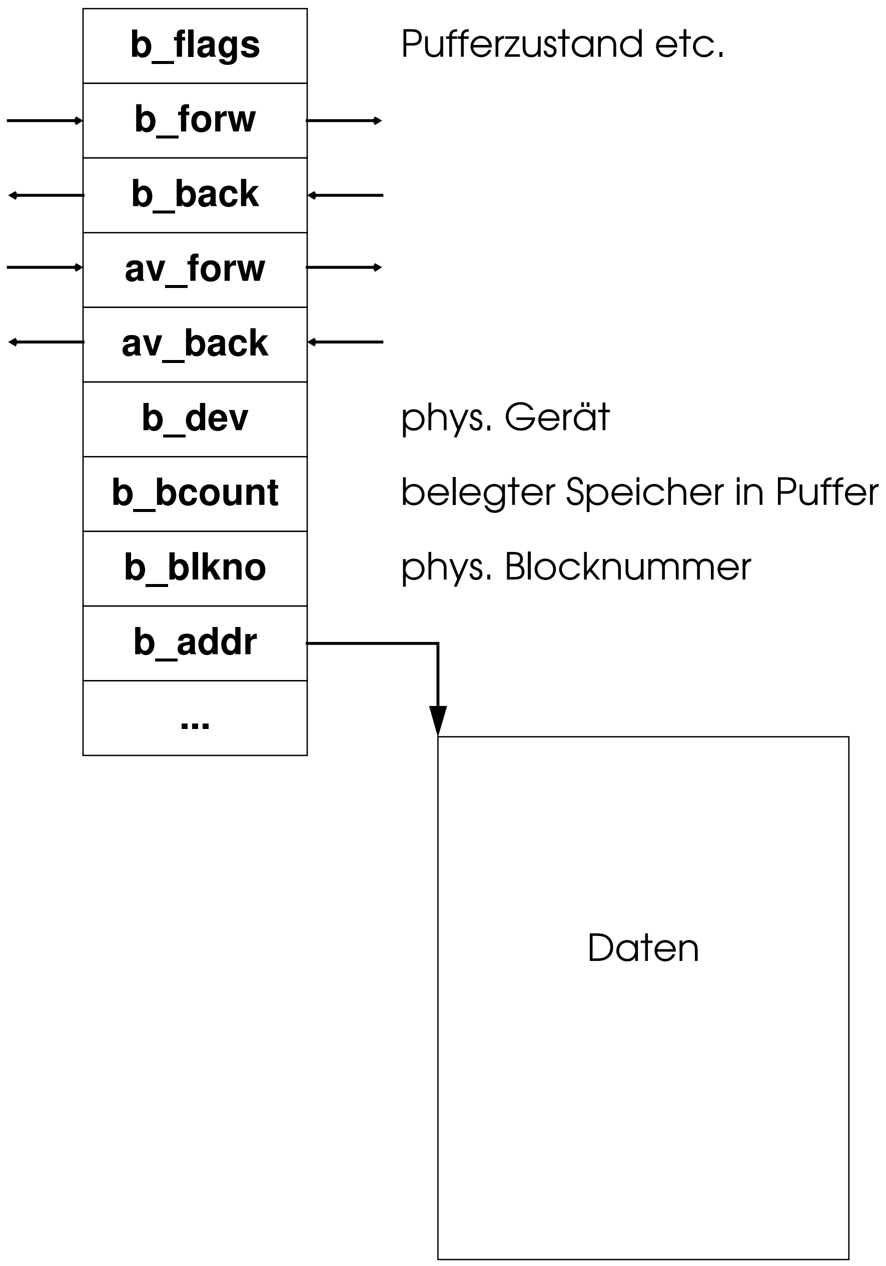
❑ Pufferverwaltungssystem mit Verwaltungsalgorithmen, ähnlich denen in einem Paging System (nicht demand paging!) mit folgenden Eigenschaften:

- **read ahead**
  wird ein Block gelesen, so wird auch der Transfer des folgenden Blocks angestoßen (wenn sequentiell gelesen wird)
- **write after**
  wird ein Block geschrieben, bedeutet dies nicht den sofortigen physikalischen Transfer auf die Platte
- **geschrieben** wird u. a. bei folgenden Ereignissen:
  - Datei wird geschlossen
  - keine freien Puffer mehr vorhanden
  - vom System *(clock)* wird ein ***update*** angestoßen
  - Systemaufruf ***sync(2)*** oder ***fsync(2)*** (BSD)
- Werden ausgelagerte Blöcke, deren Puffer noch nicht anderweitig benutzt wurde, erneut angesprochen, so werden sie reaktiviert ***(reclaim).***
- wenn zur Gewinnung von freien Blöcken eine Auslagerung angestoßen wird, werden die Kandidaten für die Freiliste nach dem LRU-Algorithmus ausgewählt

Kontrollstrukturen



Buffer-Header (struct buf, siehe *<sys/buf.h>*)



Einschlägige Systemaufrufe bei Solaris (Auswahl)

- ❑ Bestimmung der Kachelgröße
  - get system page size

```
int getpagesize()
```

- ❑ Abbildung von Dateien in den virtuellen Adreßraum
  - map pages of memory

```
caddr_t mmap(caddr_t addr, size_t len, int prot,
             int flags, int fd, off_t off)

PROT_READ     0x1   /* page can be read */
PROT_WRITE    0x2   /* page can be written */
PROT_EXEC     0x4   /* page can be executed */
PROT_NONE     0x0   /* page can not be accessed */

MAP_SHARED      1   /* Share changes */
MAP_TYPE      0xf   /* Mask for type of mapping */
MAP_FIXED    0x10   /* Interpret addr exactly */
```

  - unmap pages of memory

```
int munmap(caddr_t addr, int len)
```

- ❑ Beeinflussung des Seitenaustausches
  - memory management control

```
int mctl(caddr_t addr, size_t len,
         int function, void *arg)


MC_LOCK        lock pages
MC_LOCKAS      lock all pages of address space
MC_SYNC        synchronize pages
MC_UNLOCK      unlock pages
MC_UNLOCKAS    remove all address space locks
```