File Management

Peter Puschner
Institut für Technische Informatik
peter@vmars.tuwien.ac.at

Motivation

- Prozess kann nur eine eingeschränkte Informationsmenge in seinem Adressraum halten …
- Adressraum des Prozesses ist nur während der Lebensdauer des Prozesses verfügbar ...
- Notwendigkeit, dass Daten von mehreren Prozessen verwendet werden können ...
- Computer als Dokumentenspeicher

⇨ Files/Filesystem auf Festplatte, etc.

File Management

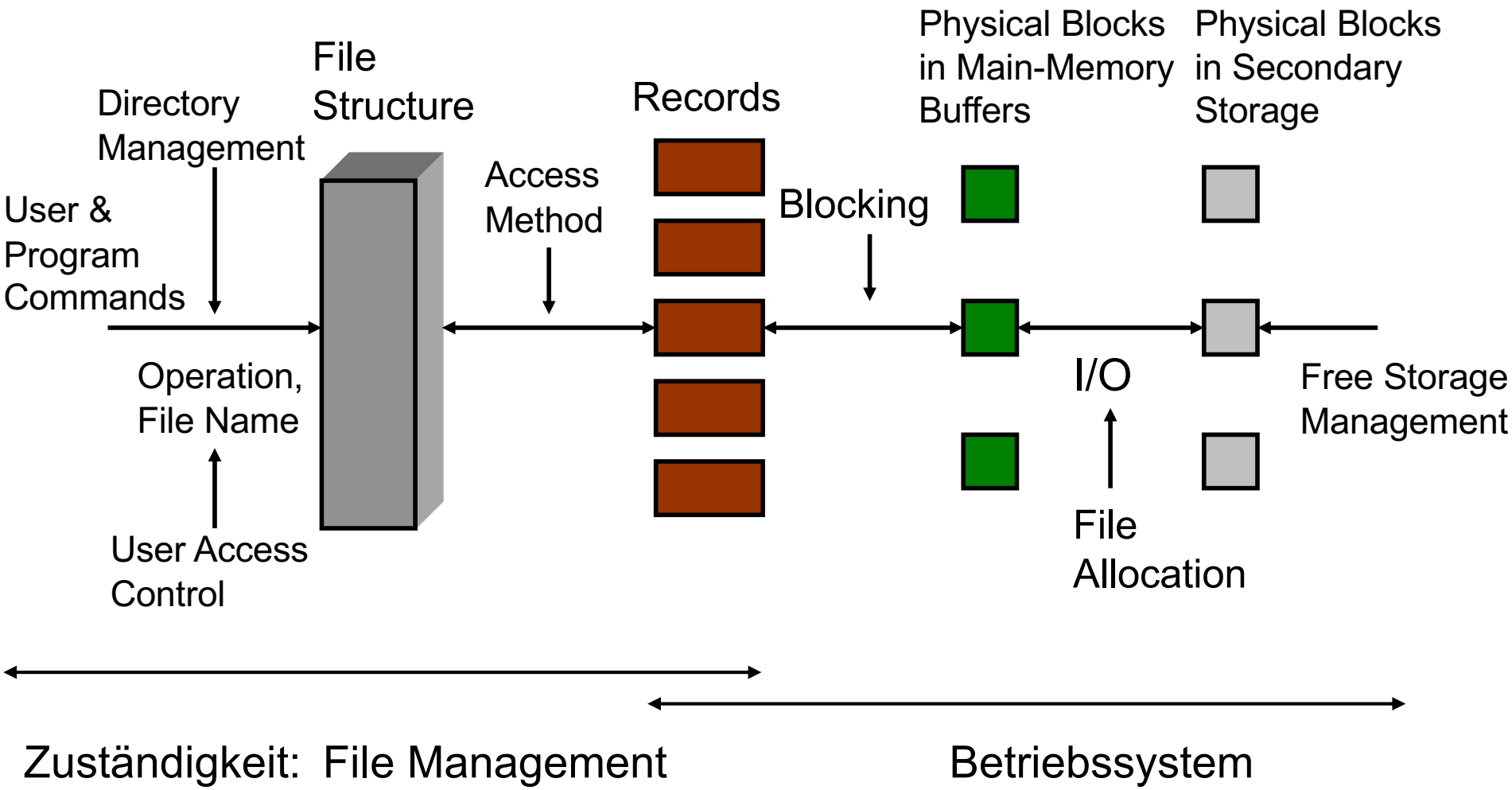
- File: zentrales Element in vielen Prozessen
- File speichert große Informationsmengen
- Persistenz: Informationslebendauer in Files wird nicht durch Prozessstart/termination beeinflusst
- Zugriff durch mehrere Prozesse

File System verwaltet Files:

Namensgebung (Naming), Struktur, Lokalisierung und Zugriff, Schutz (Protection)

Benutzersicht versus BS-Designer-Sicht

Elemente des File Management



Dateiorganisation: Ziele

- Kurze Zugriffszeit
- Leichte Aktualisierbarkeit/Veränderbarkeit
- Geringer Platzverbrauch
- Gute Wartbarkeit
- Zuverlässigkeit

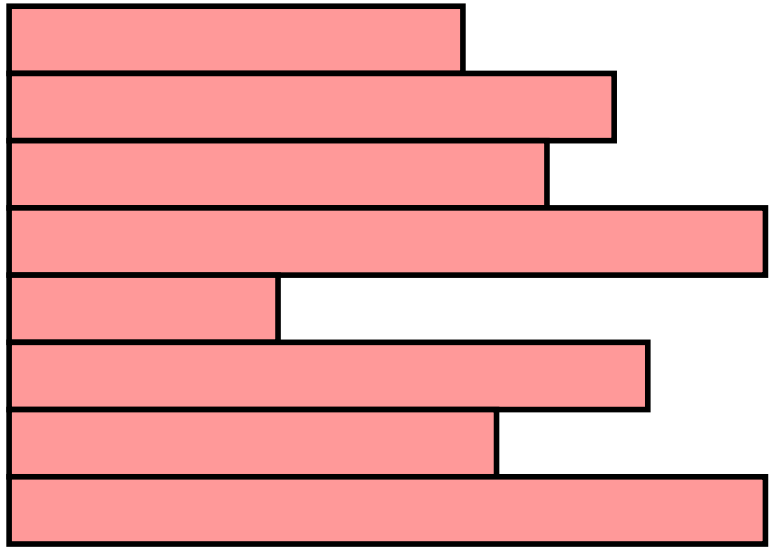
Datei-Organisation und –Zugriff

mögliche Arten der Datei-Organisation:

- unstructured sequence of bytes
- pile: Records variabler Länge werden in Reihenfolge des Ankommens gespeichert.
- sequential file: lauter Records mit fixem Format, ein Key-Feld bestimmt die Position innerhalb der Datei.
- indexed-sequential file: Index für direkten Zugriff.
- indexed file: Für alle Suchfelder eigene Indizes.
- direct (hashed) file: Hashfunktion über Key-Feld, keine sequentielle Reihenfolge der Dateien.

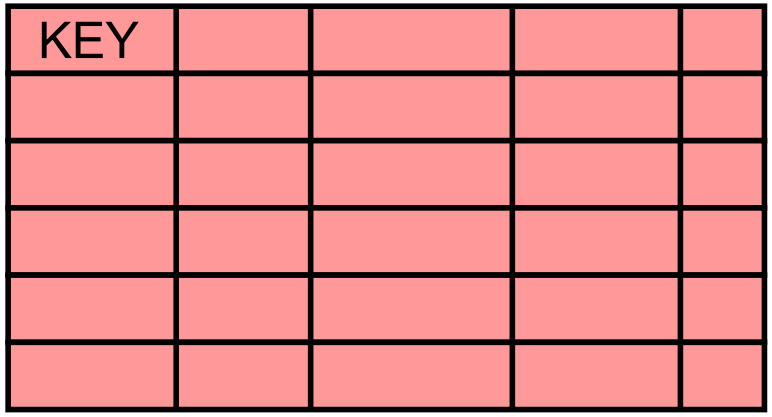
Datei-Organisation

Pile:



- Records variabler Länge
- Variable Menge an Feldern
- Chronologische Reihenfolge

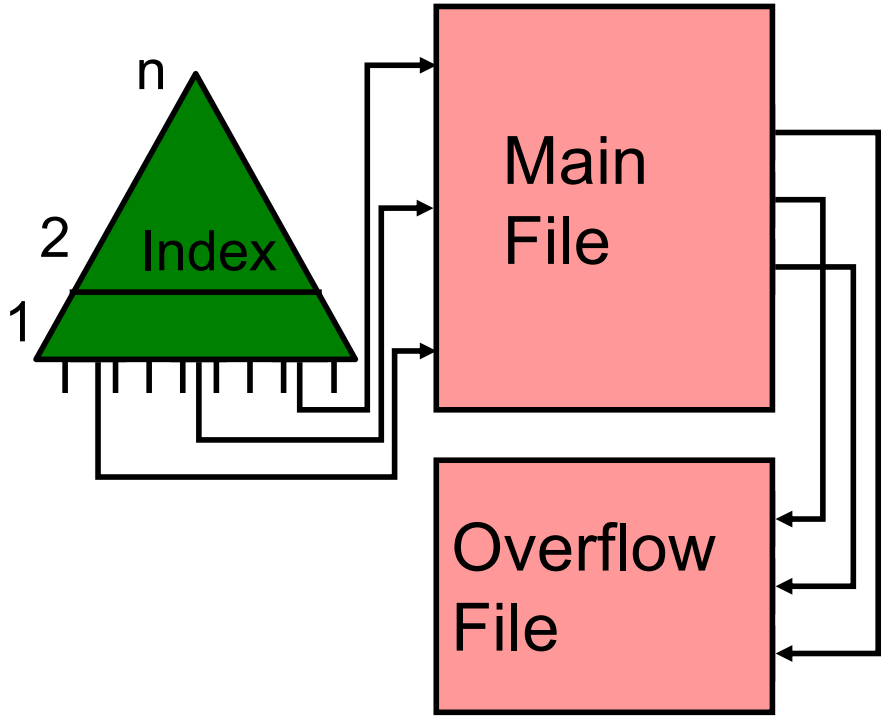
Sequential File:



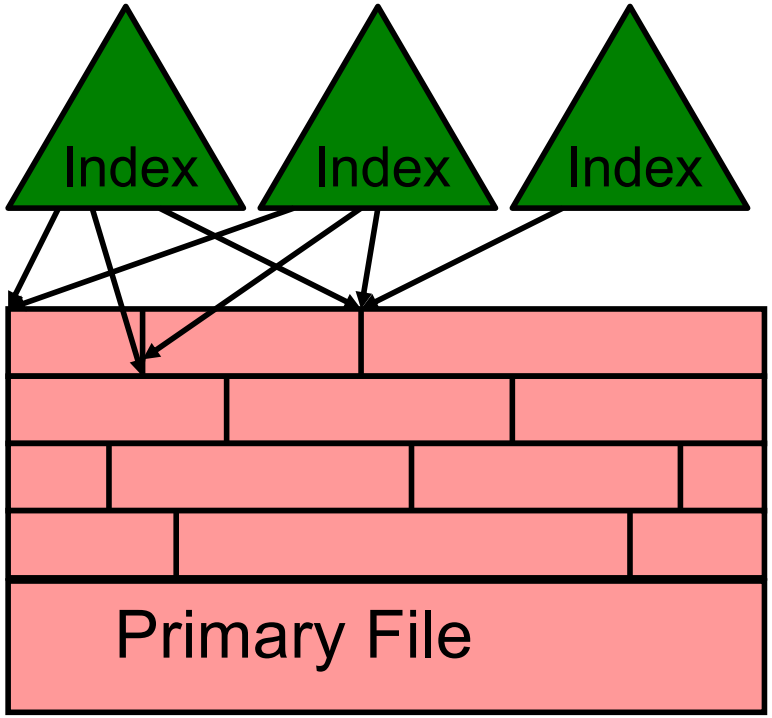
- Records fixer Länge
- Fixe Menge an Feldern mit fixer Reihenfolge
- Reihenfolge durch Key

Datei-Organisation (2)

Indexed Sequential File:

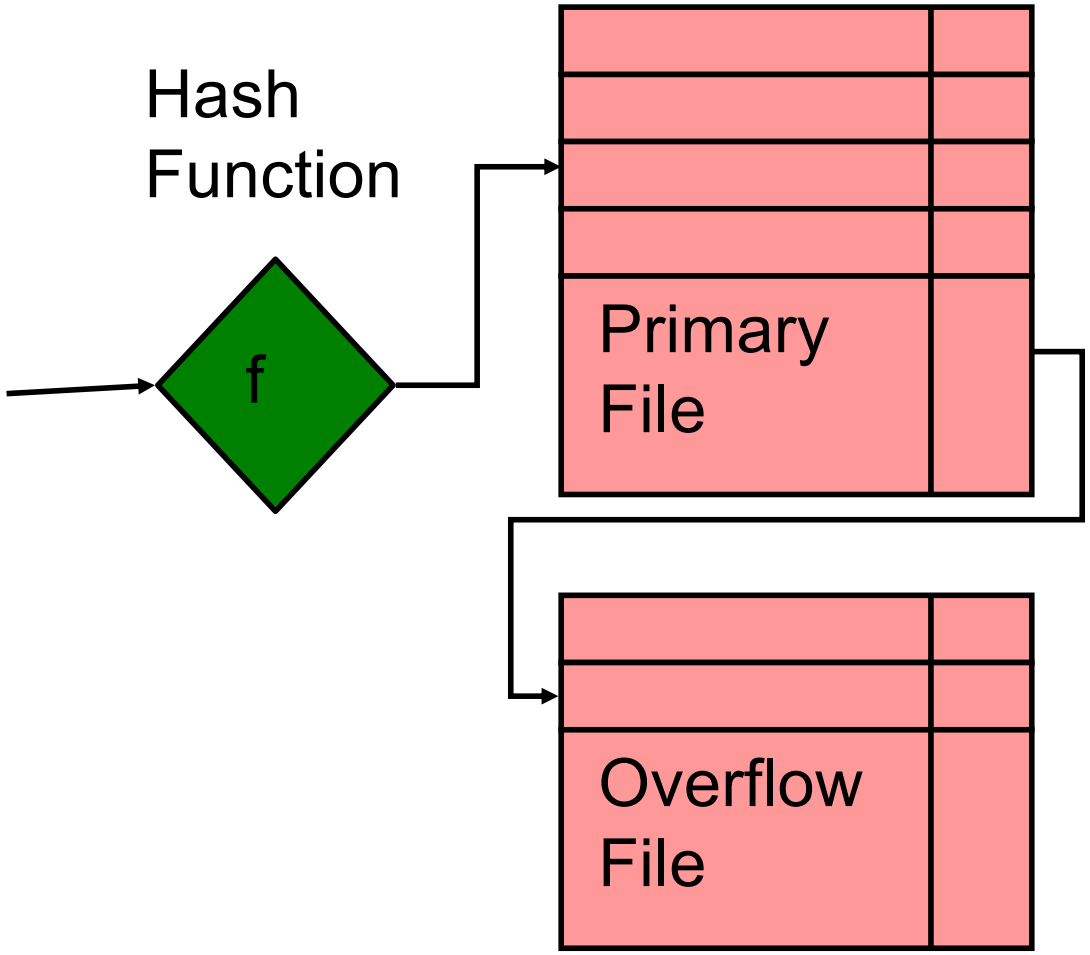


Indexed File:



Datei-Organisation (3)

Hash File:



File Types

- Regular Files
  ASCII Files versus binary Files
  binary Files: Daten, Executables, etc.
  - BS muss eig. Exe-File Format interpretieren können:
    Header, Text, Data, Relocation Bits, Symbol Table
  - Magic Number im Header kennzeichnet exe-File
- Directories
- Character Special Files
  Repräsentation sequentieller I/O Geräte
- Block Special Files
  Repräsentation von Platten

File Attributes

- Creator, Owner, Protection, Password
- Read-only flag, Hidden flag, System flag,
  Archive flag, ASCII/binary flag,
  Random access flag, Temporary flag, Lock flags
- Creation time, Time of last access,
  Time of last change
- Record length, Key position, Key length
- Current size, Maximum size

File Names

- Anzahl der Zeichen (8...255)
  - Bsp.: MS-DOS (Win95, Win98): 8 Zeichen + Extension, Unix: 255 Zeichen
- Relevanz von Groß-/Kleinschreibung, erlaubte Zeichen
  - Bsp: Unix unterscheidet Gr./Kl.Schreibung, Win95/98 nicht
- File Name Extensions
  - Teil des Namens (Win) vs. Konvention (Unix)
    Anzahl der Extensions: eine (Win) vs. beliebig (Unix)

File-Operationen

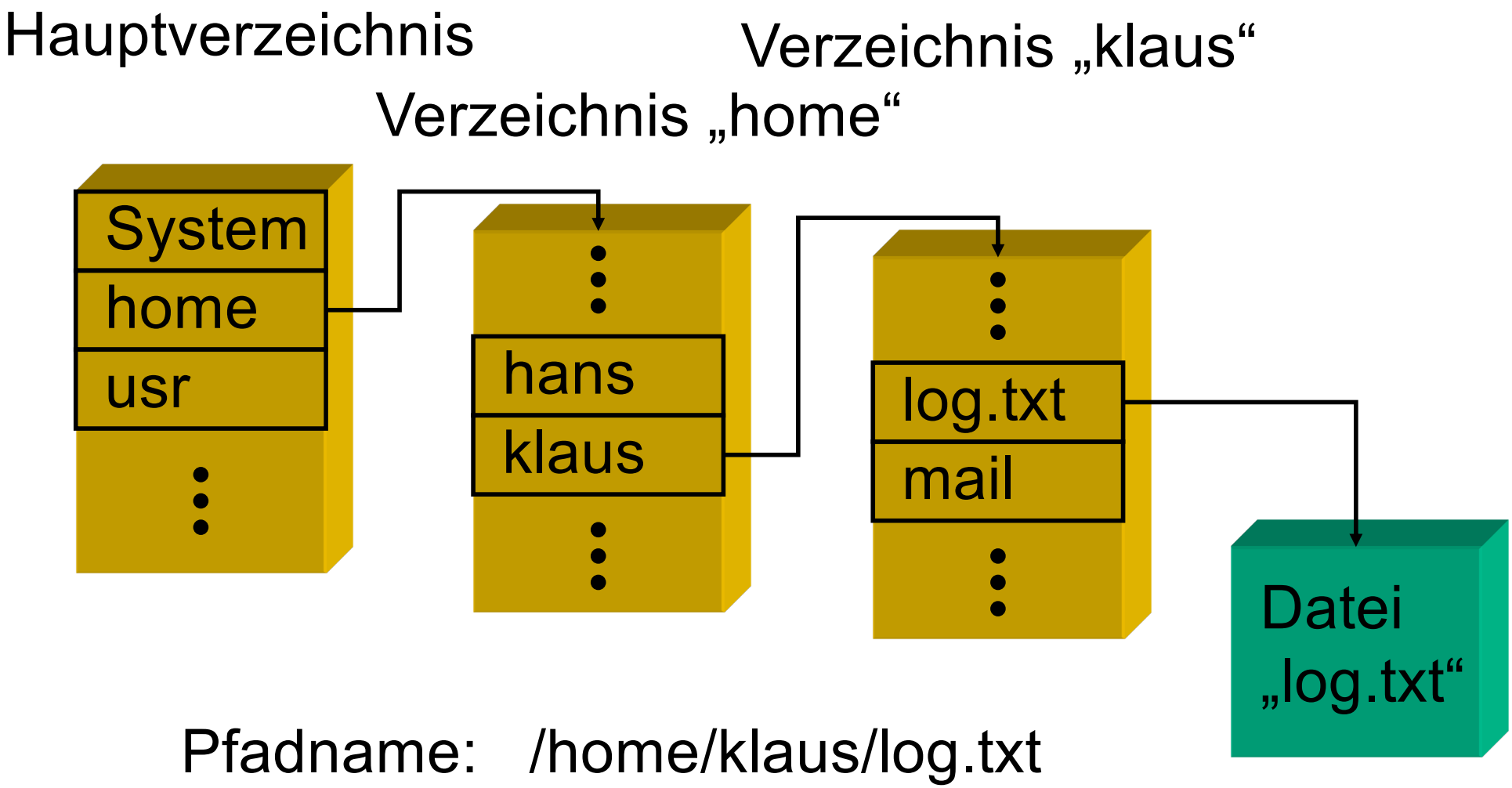
Typische Operationen:

- Create, Delete
- Open, Close
- Read, Write, Append
- Seek
- Get Attributes, Set Attributes
- Rename
- Lock

Datei-Verzeichnis (Directory, Folder)

- Verzeichnis gespeicherter Dateien
- Liefert Abbildung von Dateinamen auf Daten.
- Einträge: Name, Attribute,
  phys. Adresse der Daten
- Struktur:
  - einfache Liste (erste BS)
  - hierarchische Baumstruktur
- Verzeichnisse in Hash-Struktur gespeichert für schnellere Zugriffszeiten (vs. separate Dateien).

Verzeichnis mit Baumstruktur



Pfadnamen

- absolut (absolute path name): identifiziert Datei durch Beschreibung des Pfads von der *Root* ausgehend

  Windows: \usr\hans\mailbox
  Unix: /usr/hans/mailbox

- relativ (relative path name): lokalisiert Datei vom *Working Directory* (*Current Directory*) aus

  working directory /usr relative path: hans/mailbox

- Current directory: . (dot) ./hans/mailbox
- Parent directory: .. (dotdot) ../home/klaus/log.txt

Directory-Operationen

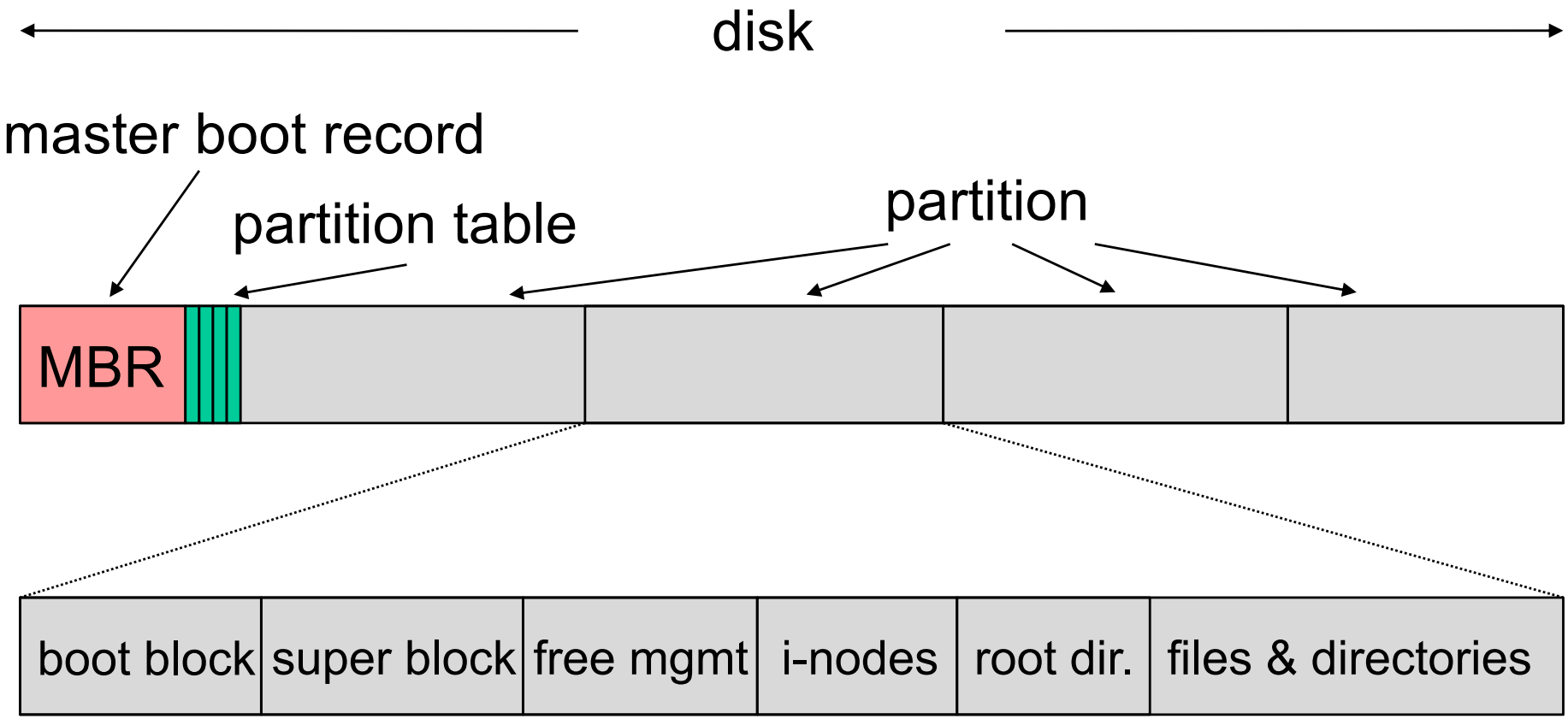
Typische Operationen:

- Create, Delete
- Opendir, Closedir
- Readdir,
- Rename
- Link, Unlink
- Änderung der Zugriffsrechte, etc.

File System Implementierung

- Wie werden Files und Directories gespeichert?
- Wie wird der Plattenplatz verwaltet?
- Performance? Zuverlässigkeit? ...

Beispiel: File-System Layout



Unix/linux partition

[figure: Operating Systems. Tanenbaum, Woodhull; PEARSON, 2006]

Disk und File-System Layout

- Disk-Unterteilung in *Partitionen* mit unabh. FS
- *Master Boot Record* (MBR) in Sektor 0 der Disk
  - Boot Code
  - Partition Table (start/end of partitions, active partition)
- Systemstart: BIOS exekutiert Code des MBR
  - Lokalisieren der aktiven Partition
  - Ausführen des ersten Blocks (=Boot Block): Laden des BS der aktiven Partition

Alternativ: Boot Menü zur BS/Partitionswahl
Boot Block im ersten Sektor (Floppy Disks)

MBR = initial program loader (IPL), volume boot code, masterboot

Datei Implementierung

- Eine Datei besteht im Sekundären Speicher als Sammlung von Blöcken
- Wo sind die Blöcke eines Files zu finden?
- Verschiedene Strategien der Block-Allokierung

Datei Implementierung (2)

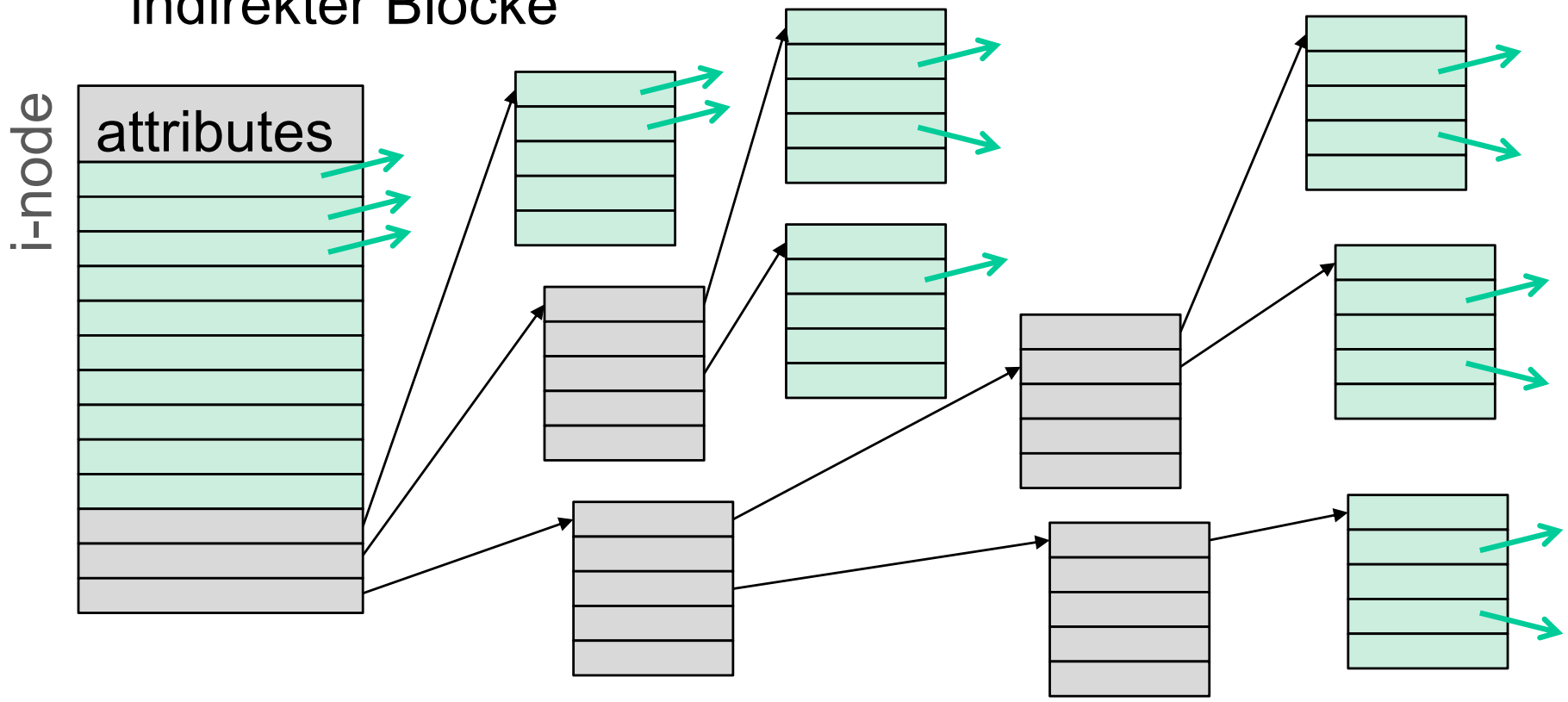
- Contiguous Allocation: eine Datei belegt eine einzige, aneinander grenzende Menge von Blöcken.
  Vorteil: gute Performance beim Lesen
  Nachteile: Platzprobleme beim Vergrößern einer Datei; externe Fragmentierung.
  Verwendung bei CD-ROMs, DVD-ROMs
- Chained Allocation: Belegung typ. einzelner Blöcke, die über Zeiger auf den Folgeblock verkettet sind.
  Vorteil: keine externe Fragmentierung.
  Nachteile: keine Lokalität der Blöcke; langsamer Zugriff bei Random Access; Nutzdaten pro Block < 2^n ,da Zeiger auf Nachfolgeblock selbst Platz im Block braucht.

Datei Implementierung (3)

- Indexed Allocation: wie Chained Allocation, allerdings werden die Pointer in einer Tabelle im Speicher (File Allocation Table, FAT) und nicht in den Blöcken der Datei gehalten.
  Vorteile: Sowohl direkter als auch sequentieller Zugriff gut unterstützt; Blöcke ganz für Nutzdaten verfügbar
  Nachteil: großer Platzbedarf für FAT im Arbeitsspeicher
- I-Nodes (Index Node): Datenstrukur für jedes File, enthält Fileattribute und Referenzen auf die Blöcke des Files.
  Vorteil: I-node wird nur im Memory gebraucht, wenn ein File verwendet wird (notwendig: Array, das i-nodes für maximale Anzahl offener Dateien halten kann)

Datei Implementierung (4)

- I-Nodes (Forts.):
  Nachteil: Anzahl der Blockreferenzen pro i-node ist begrenzt → Verwendung indirekter, doppelt und dreifach indirekter Blöcke



Sequential Files – Blocking

seq. File: Records logische Einheit für Dateizugriff

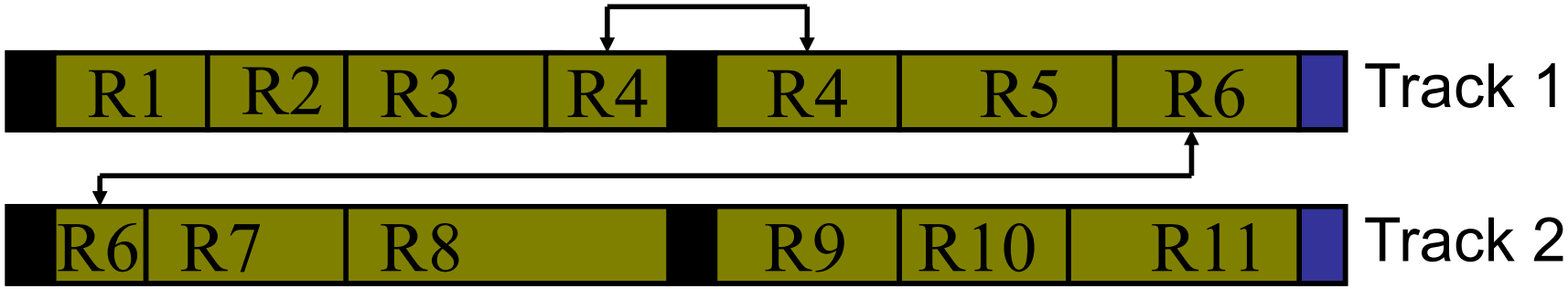
Methoden zur Abbildung von Records auf Blöcke:

- Fixed blocking: Records fixer Länge, integrale Anzahl von Records pro Block → Verschnitt
- Variable-length spanned blocking: Records variabler Länge, Records können auch auf zwei Blöcke verteilt sein → kein Verschnitt
- Variable-length unspanned blocking: Records variabler Länge, Record als ganzes in einem Block gepeichert → Verschnitt

Block-Verwaltung (2)

Fixed Blocking:

Variable Blocking Spanned:



Variable Blocking Unspanned:

Root-Directory Implementierung

Auffinden von Dateien

1. Lokalisierung des Root Directories
2. Interpretation des Pfadnamens

Position des Root Directories

- fixe Position vom Partitionsanfang aus
- Unix: Startadresse der i-nodes im Super Block
  erster i-node verweist of Root Directory
- Win: Boot Sector enthält Information über Adresse der *Master File Table* (*MFT*)

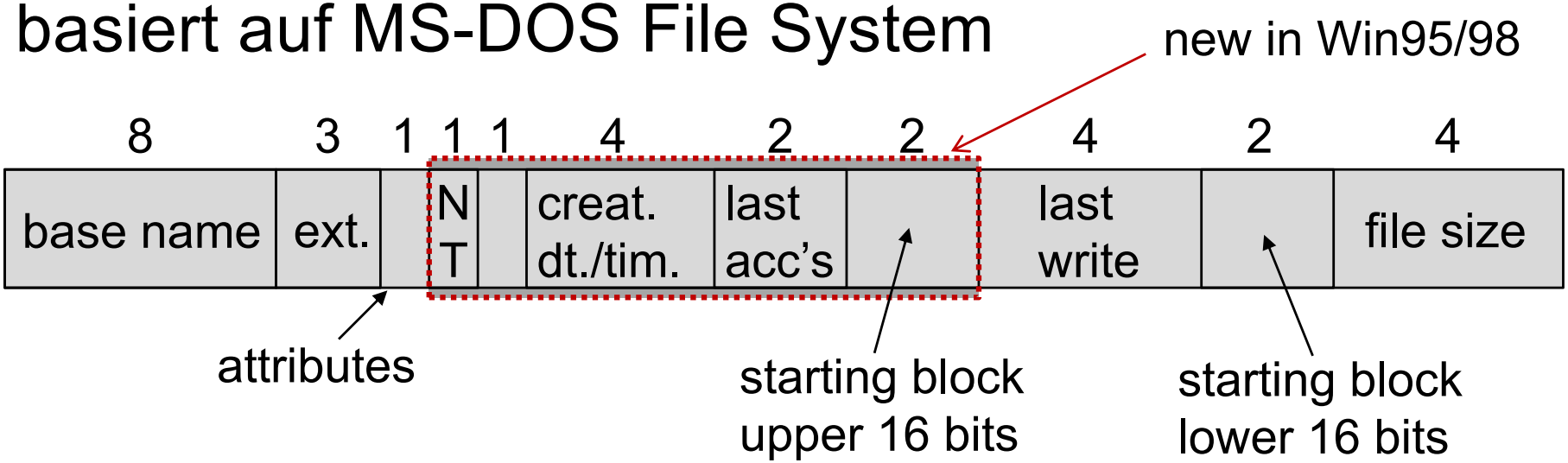
Directories und File-Attribute

Wo findet das BS die File-Attribute?

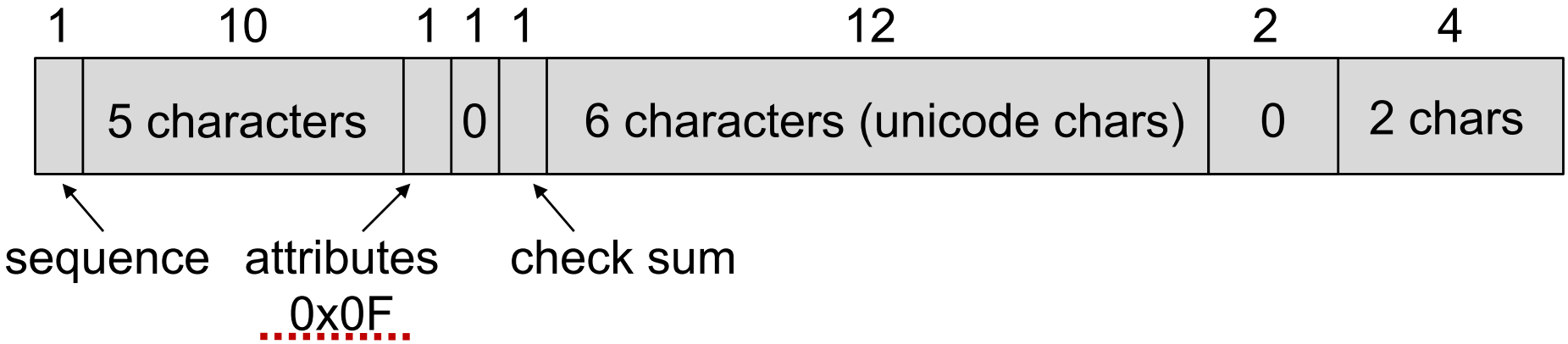
- File-Attribute in Directory-Einträgen
  Bsp.: Directory-Einträge fixer Größe der Form:
  Filename (fixe Größe), Struktur mit File Attributen, eine oder mehrere Block-Adressen
- Ablegen der Attribute in i-nodes
  Directory Einträge: Filename, i-node Nummer

Bsp.: Directories in Win95/98

basiert auf MS-DOS File System

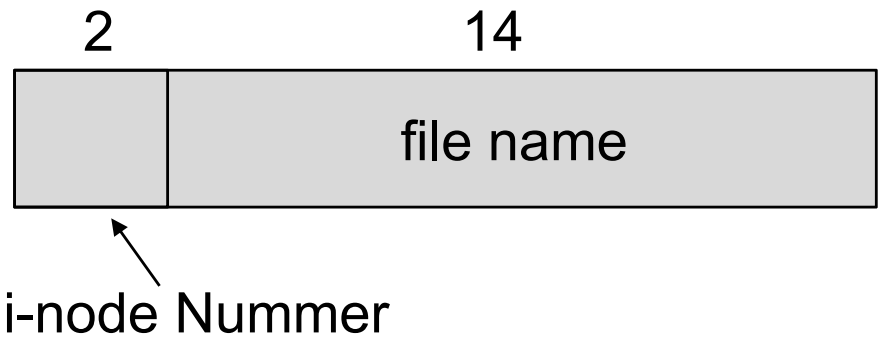


Erweiterungseinträge für lange Filenamen



Bsp.: Directories in Unix

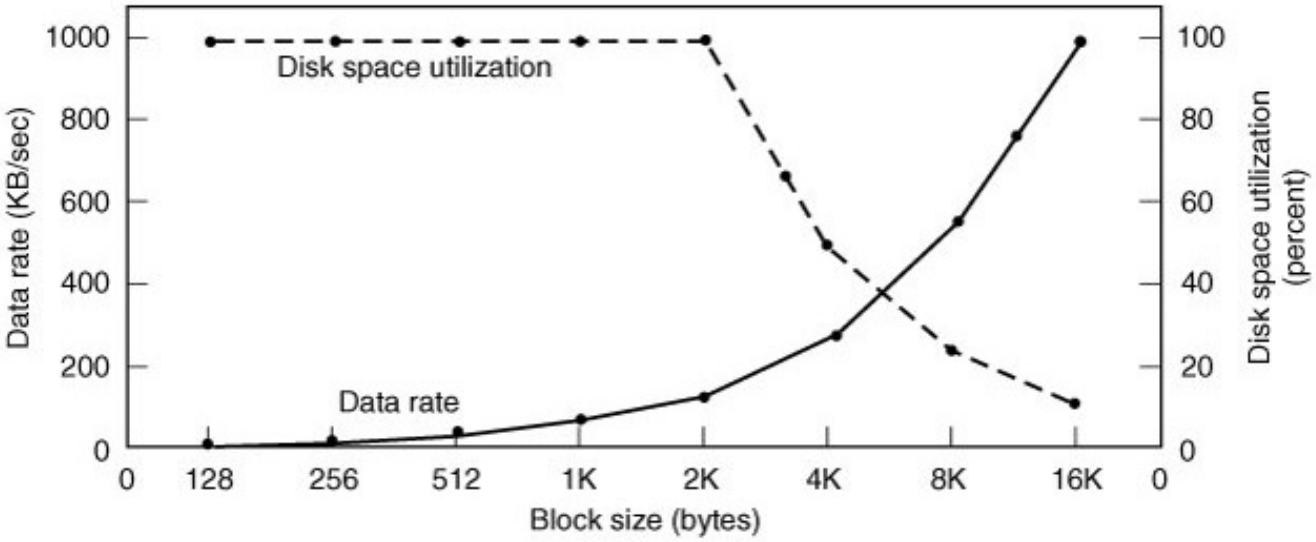
ursprüngliche Basisstruktur



File Block Size

Wie groß sollen die Blöcke von Dateien sein?

- Nutzung des Speicherplatzes
- Zugriffszeit



[figure: Operating Systems. Tanenbaum, Woodhull; PEARSON, 2006]

Speicherplatznutzung vs. Datenrate, Dateigröße konst. 2Kbyte

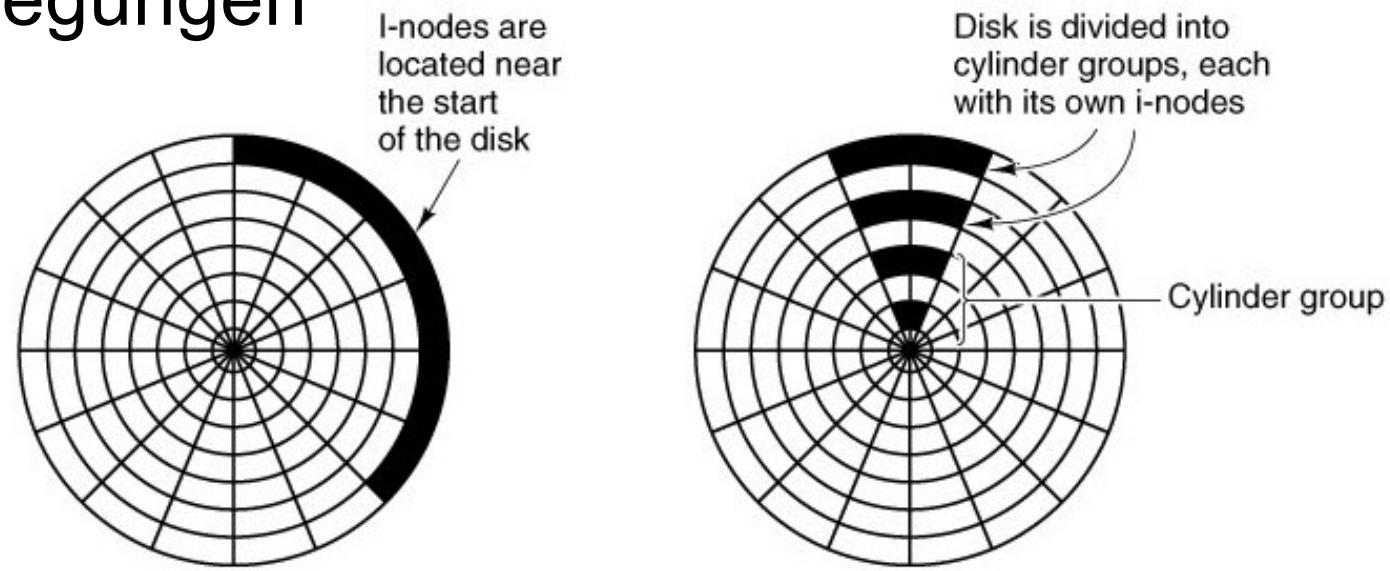
Verwaltung freier Blöcke

Disk Allocation Table zur Markierung freier Blöcke

- Chained Free Portions: Alle freien Bereiche verbunden per [Zeiger/Länge]-Eintrag. Problem: mit der Zeit Fragmentierung, Overhead für Zeigerupdate (=R/W) bei Fileoperationen.
- Bit Tables: Bitvektor mit je einem Bit pro Platten-block. Geringer Platzbedarf, guter Überblick über Folgen von freien Blöcken.
- Indexing: Freie Blöcke als eigenes File betrachtet. Effizient für alle Datei-Belegungsverfahren.

Performance

- Disk Caching
  - Ausnutzung der Lokalität
  - Bedeutung eines Blocks für die Konsistenz des Filesystems → Write Back
- Block Read Ahead
- Kopfbewegungen der Disk



[figure: Operating Systems. Tanenbaum, Woodhull; PEARSON, 2006]

Zusammenfassung

- File ist zentrales Element der meisten Programme.
- User-Sicht versus Designer-Sicht
- Filesystemstruktur
- Files, Directories, Special Files
- Abbildung von Files auf Dateiblöcke, Verwaltung freier Blöcke