Systemprogrammierung

Betriebssystemkonzepte: Massenspeicher

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

4. Juni 2014

Gliederung

Langfristige Datenspeicherung

Abstraktion von Informationen tragenden Betriebsmitteln

Da'tei (engl. *file*) Sammlung von Daten, eine...

- zusammenhängende, abgeschlossene Einheit von Daten
- „beliebige" Anzahl eindimensional adressierter Bytes

Dauerhaftigkeit von Dateien ist eine Frage des Speichermediums:

nicht-flüchtige Datenträger	Platte, Band, CD, DVD, ..., EEPROM
flüchtige Datenträger	RAM

- die Datei selbst ist ein durchaus unbeständiges Gebilde

Kommunikationsmittel für kooperierende Prozesse

- Mechanismus zur Weiterleitung (engl. *pipe*) von Informationen

Arten von Dateien

Unterscheidung von Programmtext und Programmdaten

ausführbare Dateien: Binär- und Skriptprogramme

- von einem Prozessor ausführbarer Programmtext

  Binär $\rightsquigarrow$ CPU, FPU, MCU, JVM, ..., Basic, Lisp, Prolog

  Skript $\rightsquigarrow$ `perl(1)`, `python(1)`, {`a`,`ba`,`c`,`tc`}`sh(1)`, `tcl(n)`
- der Prozessor liegt in Hard-, Firm- und/oder Software vor

nicht-ausführbare Dateien: Text-, Bild- und Tondaten

- von einem Prozessor verarbeitbare Programmdaten
  - .{`doc`, `fig`, `gif`, `jpg`, `mp3`, `pdf`, `tex`, `txt`, `wav`, `xls`, ...}
  - .{`a`, `c`, `cc`, `f`, `F`, `h`, `l`, `o`, `p`, `r`, `s`, `S`, `y`, ...}
- der Prozessor liegt in Form von Programmtext vor

Bezeichnung von Dateien

Symbolische und numerische Dateiadressen

Dateien sind „von außen“ über symbolische Adressen erreichbar...

- **benutzerdefinierter Name** von beliebiger aber maximaler Länge
- auch als Dateiname (engl. *file name*) bekannt
  - wird ggf. vom Betriebssystem (teilweise) interpretiert

...„nach innen“ besitzt jede Datei eine numerische Adresse

- **systemdefinierte Kennung** einer Datenstruktur der Dateiverwaltung
- identifiziert den sogenannten Dateikopf (engl. *file head*)

Tupel

- symbolische und numerische Dateiadresse bilden ein (festes) Paar

Erweiterung eines Dateinamens

Anreicherung um semantische Information

Dateinamensuffix (engl. *file extension*): eine meist durch einen Punkt vom Dateinamen abgegrenzte **symbolische Erweiterung** des Dateinamens

- liefert einen Hinweis auf das Dateiformat bzw. den Dateitypen

`.doc`	Textdokumente	MS-Word	
`.fm`		Framemaker	`maker(1)`
`.tex`		LaTeX	`latex(1)`
`.h`	Programme	Präprozessor	`cpp(1)`
`.c`		Kompilierer	`cc(1)`
`.s`		Assemblierer	`as(1)`
`.o`		Binder	`ld(1)`

- ist Dienstprogrammen und/oder dem Betriebssystem bekannt
  - bei UNIX die Dienstprogramme, bei Windows das Betriebssystem

Dateikopf und Dateikopfnummer

Systeminterne Verwaltungsdaten einer UNIX-Datei

„*inode*“ (bzw. „*i-node*“, 1. Edition, 1971) enthält **Dateiattribute**:

- Eigentümer (*user ID*)
- Gruppenzugehörigkeit (*group ID*)
- Typ (reguläre/spezielle Datei)
- Rechte (lesen, schreiben, ausführen; Eigentümer, Gruppe, „Welt“)
- Zeitstempel (letzter Zugriff, letzte Änderung [Typ, Zugriffsrechte])
- Anzahl der Verweise („*hard links*“)
- Größe (in Bytes)
- Adresse(n) der Daten auf dem Speichermedium

„*inode number*“, Index in eine Tabelle von Dateiköpfen („*inode table*“):

- die **numerische Adresse** der Datei (innerhalb des Dateisystems)

Dateityp

Dateien zur Abstraktion von Daten, Geräten und Kommunikationsmitteln

reguläre Datei (engl. *regular file, ordinary file*)

- problemorientiertes, eindimensionales Bytefeld

spezielle Datei ein „Sammelsurium“ von (UNIX) Konzepten:

- Verzeichnis (engl. *directory*)
  - Katalog von regulären und/oder speziellen Dateien
- Gerätedatei (engl. *device file*)
  - Zugang zu zeichen-/blockorientierten Geräte(treiber)n
- symbolische Verknüpfung (engl. *symbolic link*)
  - Abbildung eines Dateinamens auf einen Pfadnamen (S. 19)
- „benannte Leitung“ (engl. *named pipe*)
  - Kommunikationskanal zwischen unverwandten lokalen Prozessen
- Buchse (engl. *socket*)
  - Endpunkt zur bi-direktionalen Kommunikation zwischen Prozessen

Dateiverzeichnis

Konzept zur Gruppierung von Dateinamen

Katalog (engl. *catalogue*, *directory*) von symbolischen Namen

- definiert einen gemeinsamen **Kontext**
  - symbolische Adressen sind nur innerhalb ihrer Kontexte eindeutig
- implementiert eine „Umsetzungstabelle“:

$$\left\{\begin{array}{c}\text{symbolische Adresse}\\ \text{Dateiname}\\ \texttt{foo}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{c}\text{numerische Adresse}\\ \text{Dateikopf}\\ 4711\end{array}\right\}$$

- speichert die Abbildung Dateiname $\mapsto$ Dateikopfnummer

Verknüpfung von Dateiname und Dateikopf

UNIX „*hard link*“ (auch kurz: *link*)

Eintrag im Dateiverzeichnis: Dateiname $\mapsto$ Dateikopfnummer

UNIX V7, `dir.h` [3]

```
typedef unsigned short ino_t;

#define DIRSIZ 14

struct direct {
    ino_t d_ino;
    char  d_name[DIRSIZ];
};
```

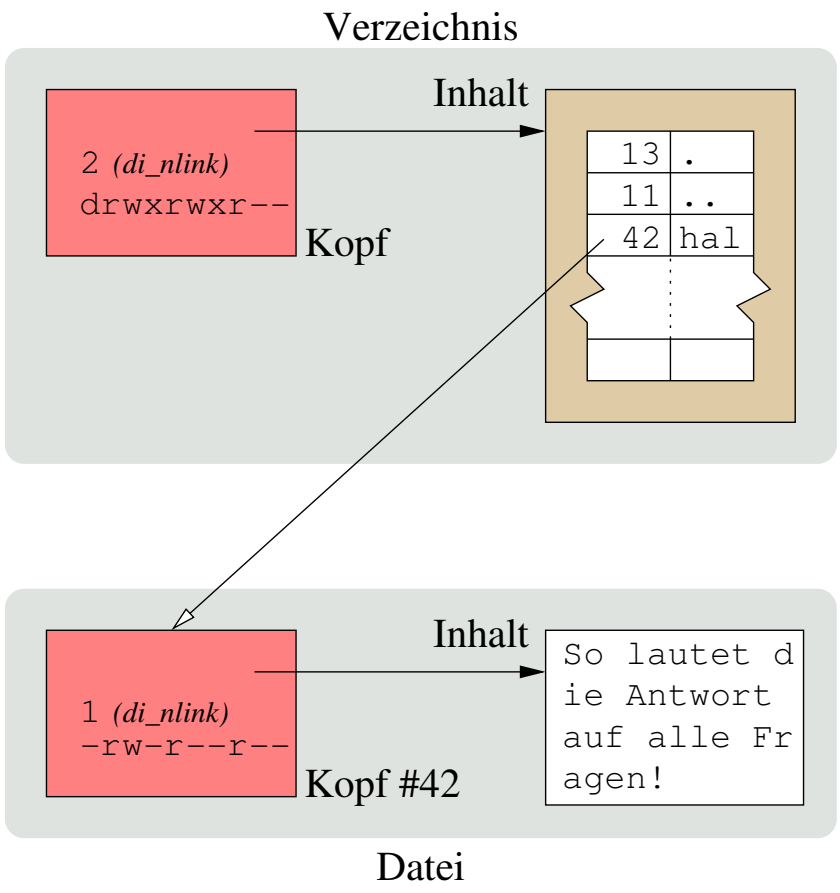
- die Abbildung ist als **Wertepaar** gespeichert
- mehrere Einträge können auf denselben Dateikopf verweisen
  - identische Dateikopfnummern
  - verschiedene Dateinamen
- Referenzzähler im Dateikopf vermerkt Anzahl der Verweise

Anlegen/Löschen erfordert nur Schreibzugriffsrecht auf das Verzeichnis

- unabhängig von den Zugriffsrechten auf die referenzierte Datei

Verknüpfung von Dateiname und Dateikopf (Forts.)

Einträge anlegen/löschen ist eine Operation auf Verzeichnisse



Verzeichnisse sind „Spezialdateien“

- die selbst einen Namen und Dateikopf haben
- die erreichbar sind über eine Verknüpfung
  - eines anderen Verzeichnisses
- die Namen getrennt von Dateien speichern

Verknüpfungen anlegen/löschen zu können, ist eine **Berechtigung**, *die sich nur auf das Verzeichnis der betreffenden Verknüpfungen bezieht!*

Referenzzähler (engl. *reference count*)

Unterstützung der „Müllsammlung“ (engl. *garbage collection*)

Buchführung über die Anzahl der Verknüpfungen zu einem Dateikopf geschieht über einen **Verknüpfungszähler** (engl. *link count*; `di_nlink`)

`di_nlink != 0` Datei/Verzeichnis wird referenziert

- der Dateikopf ist (aus Sicht des Systems) in Benutzung

`di_nlink == 0` Datei/Verzeichnis wird nicht referenziert

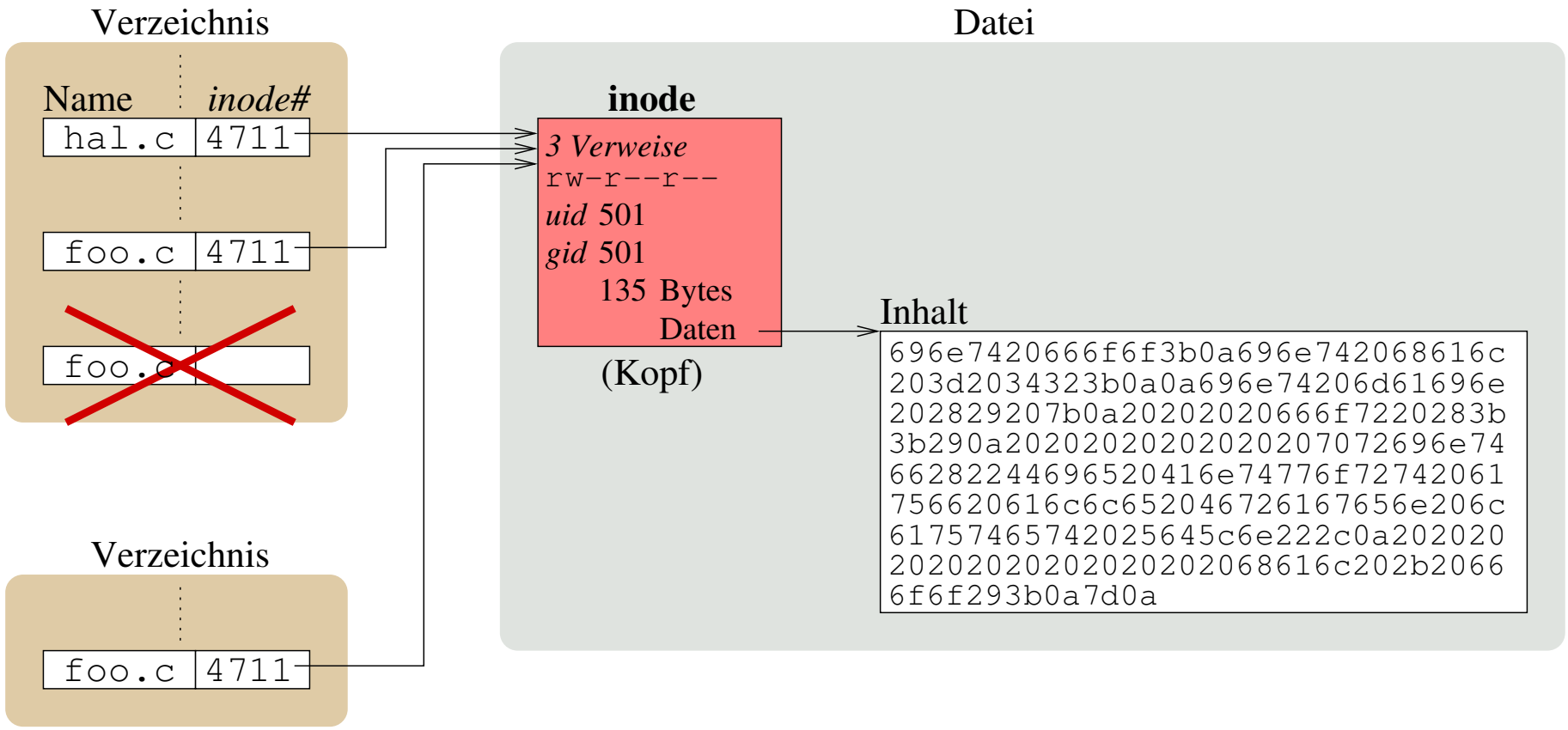
- der Dateikopf samt anhängender Daten kann freigegeben werden

Veränderung des Verknüpfungszählerwertes erfolgt beim Eintragen (++) bzw. Löschen (--) von Verknüpfungen im jeweiligen Verzeichnis:

- **Verzeichnisverknüpfung**: `mkdir(2)`/`rmdir(2)`
  - auf Verzeichnisse verweisen mindestens zwei Verknüpfungen (S. 21):
    1. der Name des Verzeichnisses im Elternverzeichnis „`..`“ und
    2. die Selbstreferenz '`.`' in dem jeweiligen Verzeichnis
- **Dateiverknüpfung**: `link(2)`/`unlink(2)`

UNIX Dateiverzeichnis und Datei

Verknüpfung, Dateikopf und Dateiinhalt



- Namenseinträge in einem Verzeichnis müssen eindeutig sein

UNIX Systemfunktionen

Operationen auf Dateiköpfe

Linux, MacOS, SunOS

```
fd  = open(path, flags, mode)
num = read(fd, buf, nbytes)
num = write(fd, buf, nbytes)
off = lseek(fd, offset, whence)
ok  = close(fd)
ok  = stat(path, buf)
    ⋮
```

Dateideskriptor (engl. *file descriptor*)

- von der Dateiverwaltung implementierter eindeutiger Bezeichner (engl. *identifier*) einer geöffneten Datei
- „nach außen“ meist durch eine Ganzzahl (engl. *integer*) repräsentiert

Gliederung

Bedeutung von Namen

Kontextfreie Namen sind bedeutungslos

Java bedeutet im Kontext...

- „Geographie“ eine Insel
- „Genussmittel“ ein Heissgetränk
- „Informatik“ eine Programmiersprache

C bedeutet im Kontext...

- „Sprache“ einen Buchstaben
- „Musik“ eine Note
- „Informatik“eine Programmiersprache

Namensräume (engl. *name spaces*) ordnen Namen Bedeutungen zu

Aufbau von Namensräumen

flache Struktur: definiert nur einen einzigen Kontext

- Eindeutigkeit muss mit der Namenswahl selbst gewährleistet werden

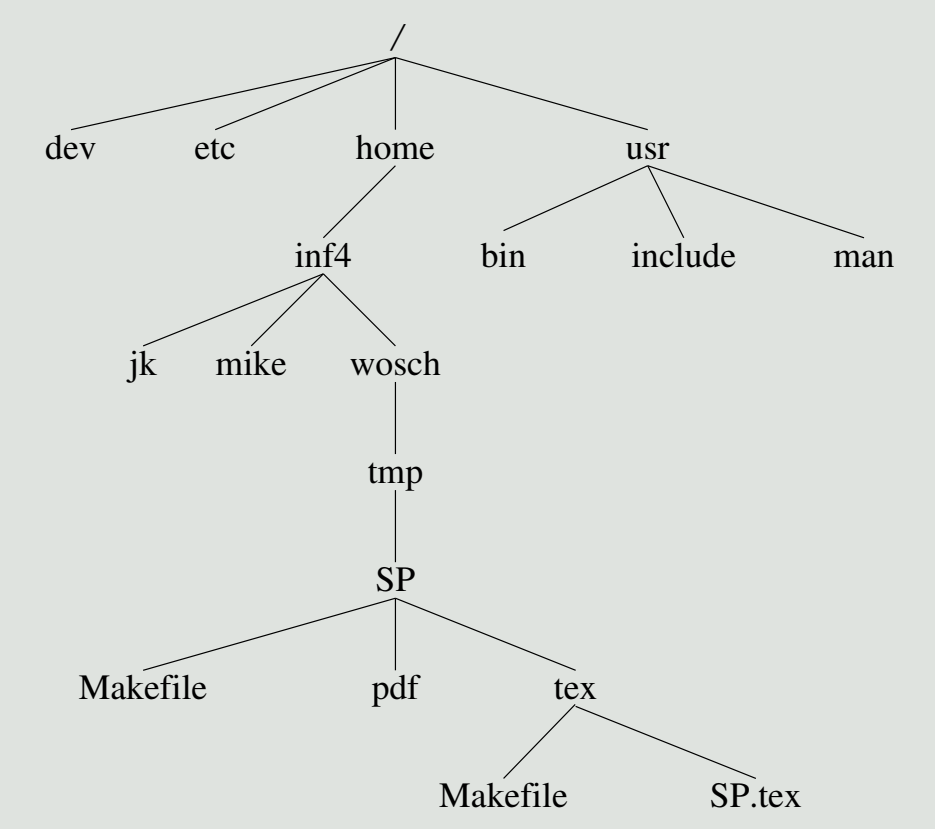
hierarchische Struktur: definiert mehrere Kontexte

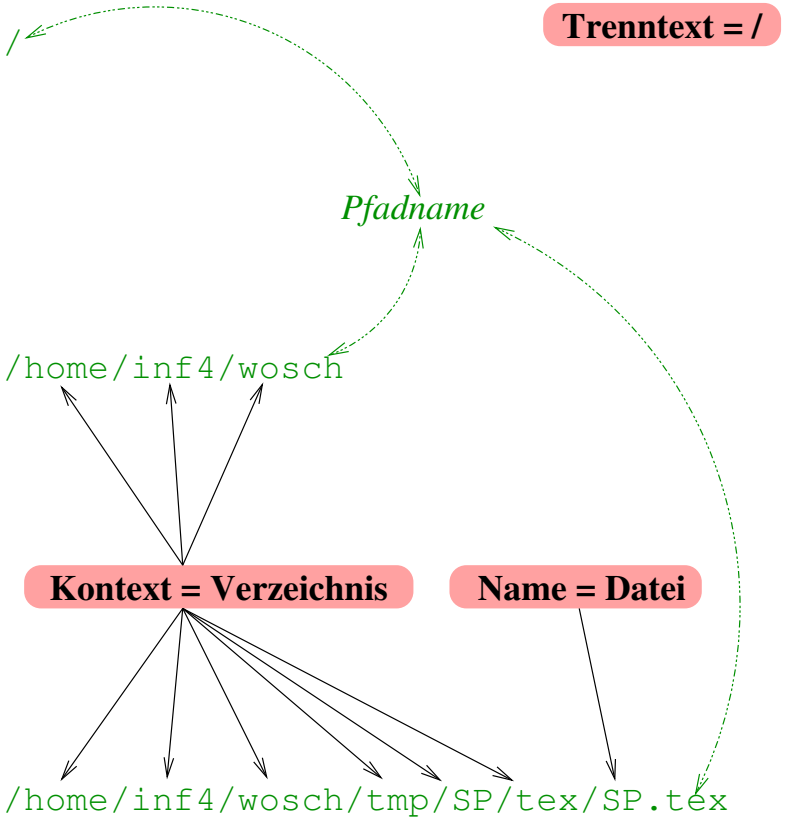
- Eindeutigkeit wird durch einen **Kontextnamen** als Präfix erreicht
  - Kontexte enthalten Namen von Dateien und/oder (anderer) Kontexte
  - der Name einer Datei entspricht einem „Blatt“ des Namensbaums
- Sonderzeichen („Trenntext“) stehen meist für **Separatoren**:

Schrägstrich (*slash*)	⇒ UNIX
zurückgelehnter Schrägstrich (*backslash*)	⇒ Windows

Hierarchischer Namensraum

Dateibaum (engl. *file tree*)





Navigation im Namensraum

Eindeutigkeit der symbolischen Adresse (einer Datei) ist durch einen Pfad (engl. *path*) im Namensraum gegeben

- der Pfadname (engl. *path name*) ist ein vollständiger Dateiname

Formaler Aufbau eines (UNIX) Pfadnamens in EBNF [2]

$$
\begin{aligned}
\textit{pathname} &= \textit{resolver} \mid ([\textit{resolver}], \{\textit{name}, \textit{resolver}\}, \textit{name});\\
\textit{resolver} &= \{\textit{separator}\}-;\\
\textit{separator} &= \text{“/“};\\
\textit{name} &= \{\textit{character}\}-;\\
\textit{character} &= \textit{character set} - \textit{separator};\\
\textit{character set} &= \text{ASCII};
\end{aligned}
$$

z.B.: `/`, `.`, `..`, `foo`, `foo/bar`, `/foo`, `bar/`, `./bar/..`, `../foo/.//bar//`

Spezielle Kontexte

Sonderverzeichnisse

Wurzelverzeichnis (engl. *root directory*)

- bezeichnet die Wurzel des Dateibaums (solitär '/', bei UNIX)
- wird vom System bzw. Administrator (engl. *super user*) gesetzt
  - `chroot(2)`, privilegierte Operation

Arbeitsverzeichnis (engl. *working directory*)

- die gegenwärtige Position eines Programms/Prozesses im Dateibaum
- ändert sich beim „Durchklettern“ des Dateibaums
  - `chdir(2)`

Heimatverzeichnis (engl. *home directory*)

- das initiale Arbeitsverzeichnis eines Benutzers/Prozesses
- wird vom System gesetzt bei Sitzungsbeginn
  - `login(1)`

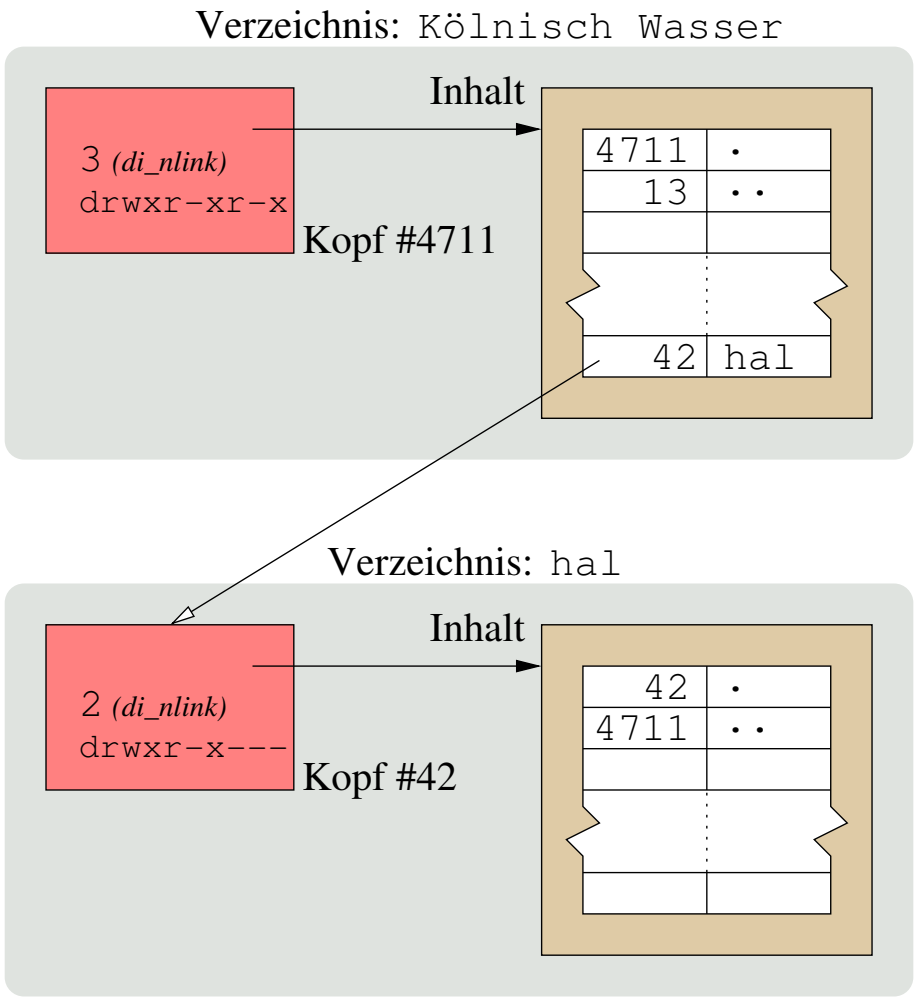
Relative Adressierung von Kontexten

Systemdefinierte Verzeichnisnamen: `mkdir(2)`

- . („*dot*“): aktuelles **Arbeitsverzeichnis** (engl. *current working directory*)
  - benennt die Verknüpfung zu selbigem Verzeichnis (Selbstreferenz)
    - ermöglicht die eindeutige Identifikation eines Arbeitsverzeichnisses, ohne dessen wirklichen Namen kennen zu müssen (`stat(2)`)
    - erzwingt einen lokalen Bezugspunkt (als Namenspräfix „`./`“)
  - erster Eintrag in jedem Verzeichnis
- .. („*dot dot*“): aktuelles **Elternverzeichnis**
  - benennt die Verknüpfung zum übergeordneten Verzeichnis, das die Verknüpfung zum Arbeitsverzeichnis enthält
    - entspricht '`.`', falls es kein Elternverzeichnis gibt (Wurzelverzeichnis)
  - zweiter Eintrag in jedem Verzeichnis

Relative Adressierung von Kontexten (Forts.)

Verknüpfungen für Arbeits- und Elternverzeichnisse



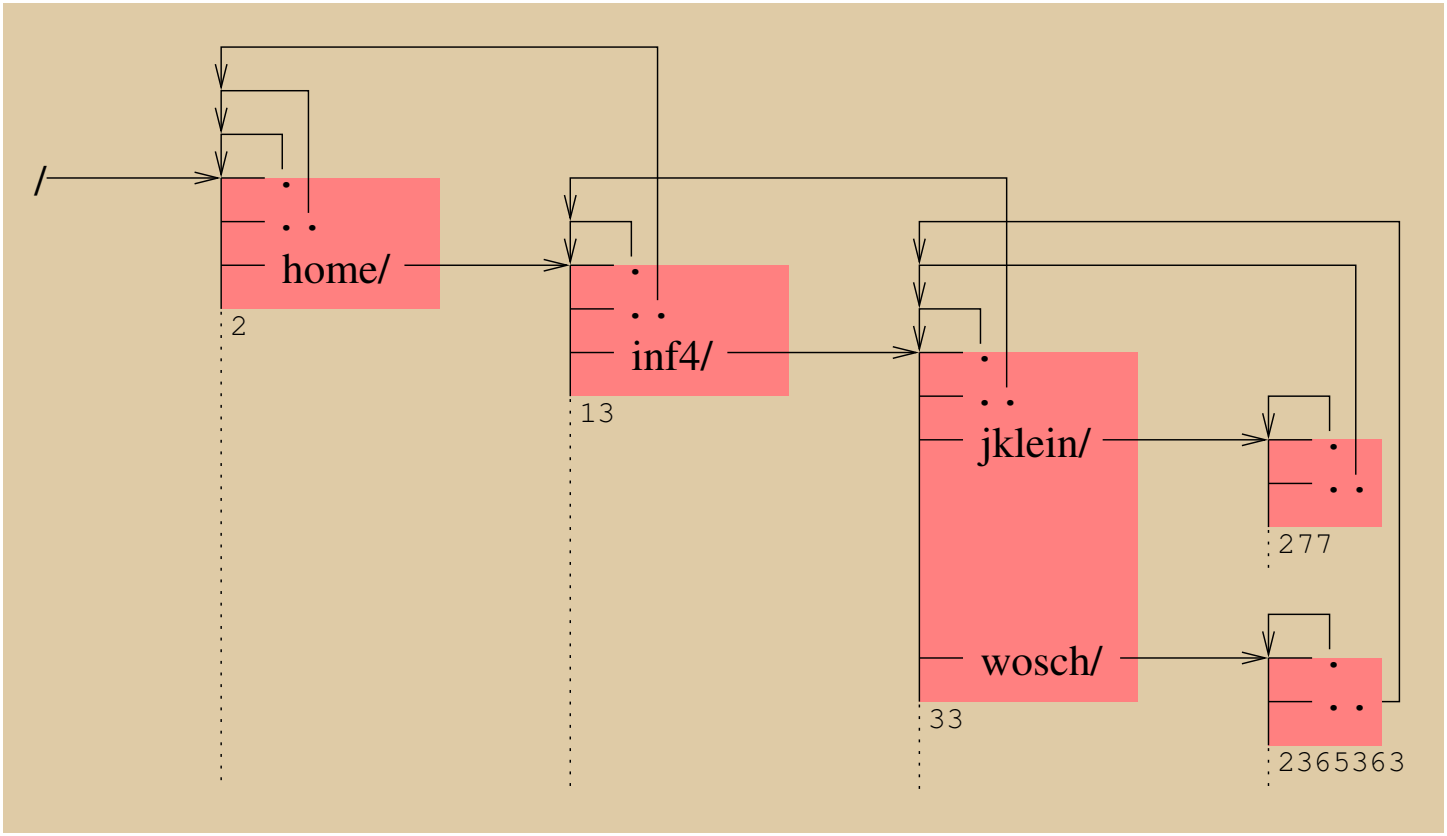
- `.` bezeichnet den Dateikopf, der das Verzeichnis selbst beschreibt
  - Dateikopfnummer des Arbeitsverzeichnisses
- `..` bezeichnet den Dateikopf des Verzeichnisses, das den Verzeichnisnamen speichert
  - Dateikopfnummer des Elternverzeichnisses

`di_nlink` # Verknüpfungen, die `.`

referenzieren SunOS, Linux

speichert MacOSX

Dynamische Datenstruktur „Dateibaum“



Verzeichnisnamen entsprechen einer Vorwärtsverkettung (z.B. `wosch`)

. ist eine Selbstreferenz

.. entspricht einer Rückwärtsverkettung

Arten von Pfadnamen

relativer Pfadname — vom gegenwärtigen Arbeitsverzeichnis ausgehend,
z.B. von `/home/inf4/wosch` aus:

- `tmp/SP/tex/SP.tex`
- `./tmp/SP/tex/SP.tex`
- `../wosch/tmp/SP/tex/SP.tex`

...oder von `/home/inf4/jk` aus:

- `../wosch/tmp/SP/tex/SP.tex`

absoluter Pfadname — vom Wurzelverzeichnis ausgehend:

- `/home/inf4/wosch/tmp/SP/tex/SP.tex`

Bindung und Auflösung von Namen bzw. Pfadnamen

Abbildung/Umsetzung: symbolische Adresse $\mapsto$ numerische Adresse

Namensbindung (engl. *name binding*) bedeutet die Abbildung der symbolischen Adresse in eine numerische Adresse

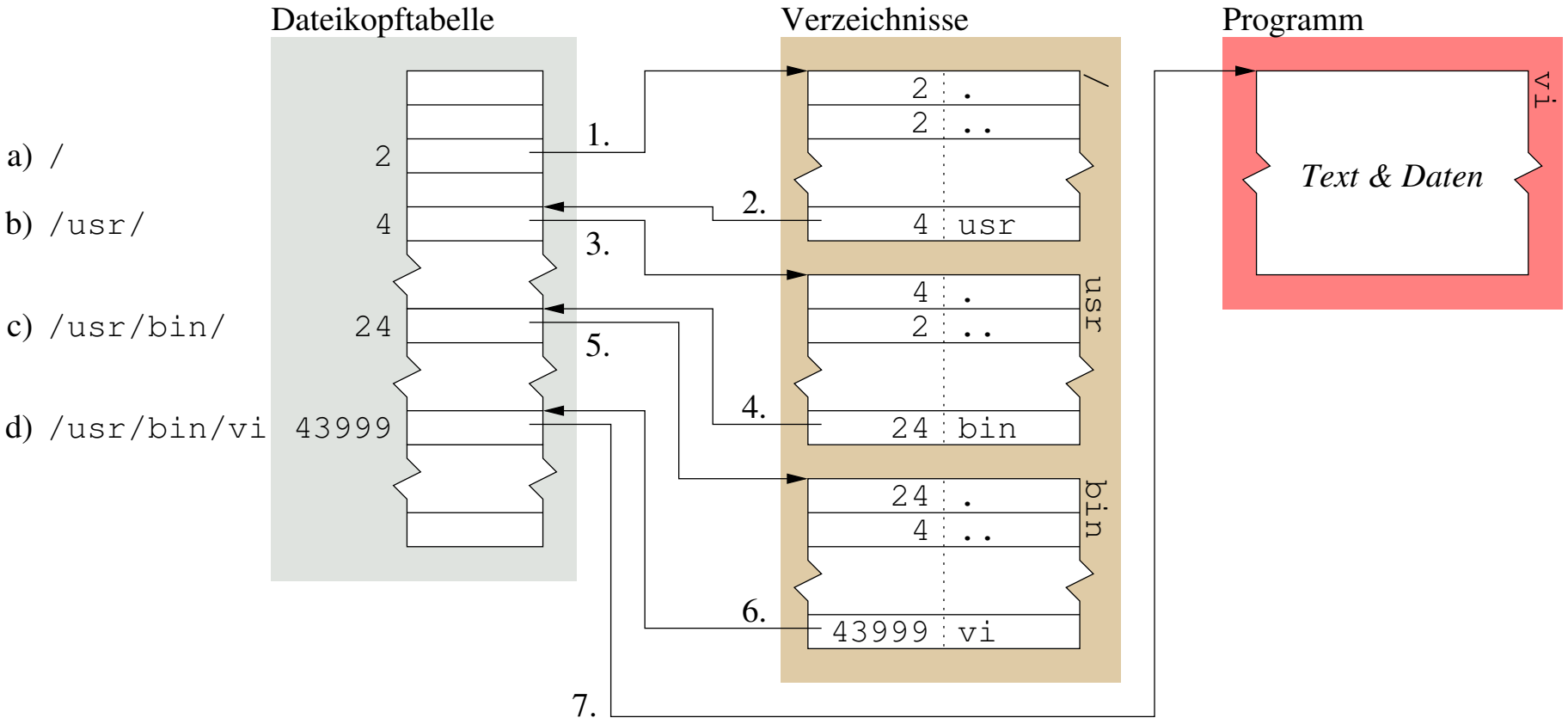
- zum Erzeugungszeitpunkt einen Datei-/Verzeichnisnamen...
  - mit einem freien/belegten Dateikopf verknüpfen
  - in ein Dateiverzeichnis eintragen
- Pfadnamen mit Dateikopf assoziieren: `creat(2)`, `link(2)`

Namensauflösung (engl. *name resolution*) bedeutet die Umsetzung der symbolischen Adresse in eine numerische Adresse

- zum Benutzungszeitpunkt Dateiverzeichnisse durchsuchen...
  - schrittweise für jeden einzelnen Verzeichnisnamen im Pfad
  - schließlich für den Dateinamen
- Dateikopf des Pfadnamens lokalisieren: `open(2)`

Auflösung von Namen bzw. Pfadnamen

Beispiel: `/usr/bin/vi`



Verwaltung von Dateien und Dateibäumen

Datenstrukturenkomplex zur Verwaltung von Hintergrundspeicher

- der Dateisystemkopf (UNIX *super block*)
  - speichert Verwaltungsinformationen und Systemparameter
  - legt die Grenzwerte des Dateisystems fest
- die Dateikopftabelle (UNIX *inode table*)
  - zur Beschreibung von Dateien und/oder Verzeichnisse
- die Datenblöcke (engl. *data blocks*)
  - zur Speicherung der Inhalte der Dateien/Verzeichnisse

Beschreibung einer Partition (engl. *partition*) im Hintergrundspeicher

- logische Unterteilung in einen Satz zusammenhängender Sektoren

Montieren von Dateisystemen

Auf- und Abbau einer Dateisystemhierarchie

Befestigungspunkt (engl. *mount point*) ist eine Stelle im **Wirtsdateisystem**, an der ein **Gastdateisystem** eingebunden werden kann

- ein (beliebiges) Verzeichnis im Wirtsdateisystem
- wird mit der Wurzel (S. 20) des Gastdateisystems überlagert

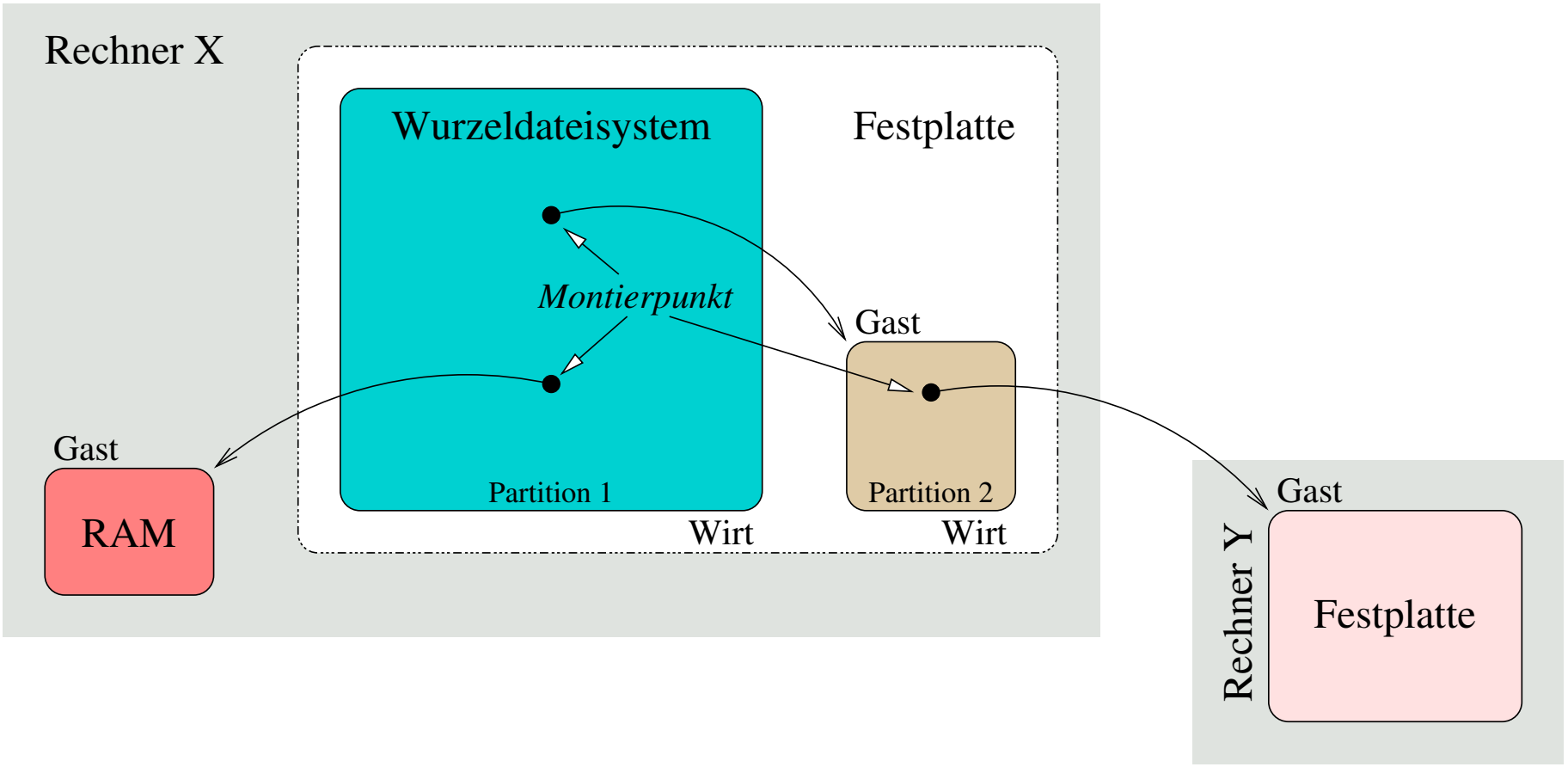
Wirts- und Gastdateisystem bilden jeweils eigene Partitionen...

- auf demselben oder einem anderen (dateisystemverträglichen) Gerät
  - z.B. Band, Fest-/Wechselplatte, CD, DVD, EEPROM, ..., RAM
  - ggf. auch auf unterschiedlichen Rechnern eines Rechnerverbunds
- von gleicher oder verschiedener (logischer) Struktur
  - ggf. ein Mix z.B. von S5FS, UFS, FFS, EXT2 und NTFS

Ausgangspunkt ist das Wurzeldateisystem (engl. *root file system*)

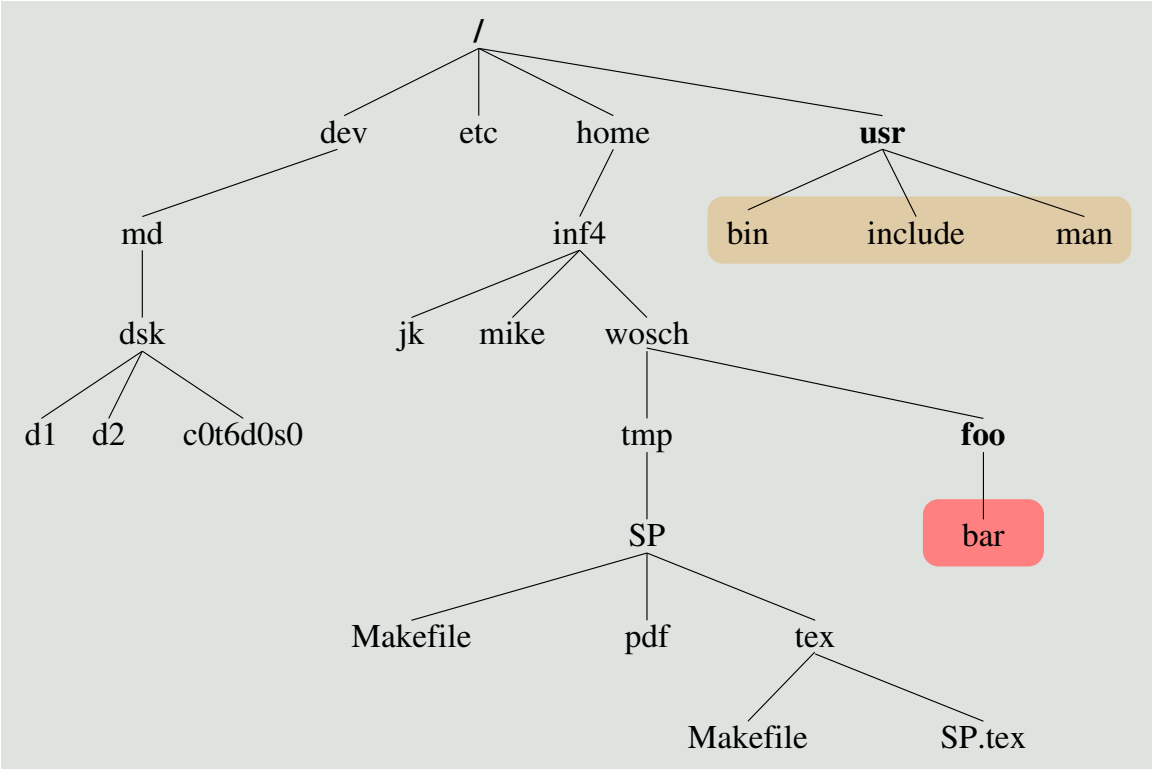
- d.h. das Dateisystem, von dem das Betriebssystem urgeladen wird

Hierarchiebildung von Dateisystemen



Einbindung von Namensräumen

Integration von Dateibäumen montierbarer Dateisysteme



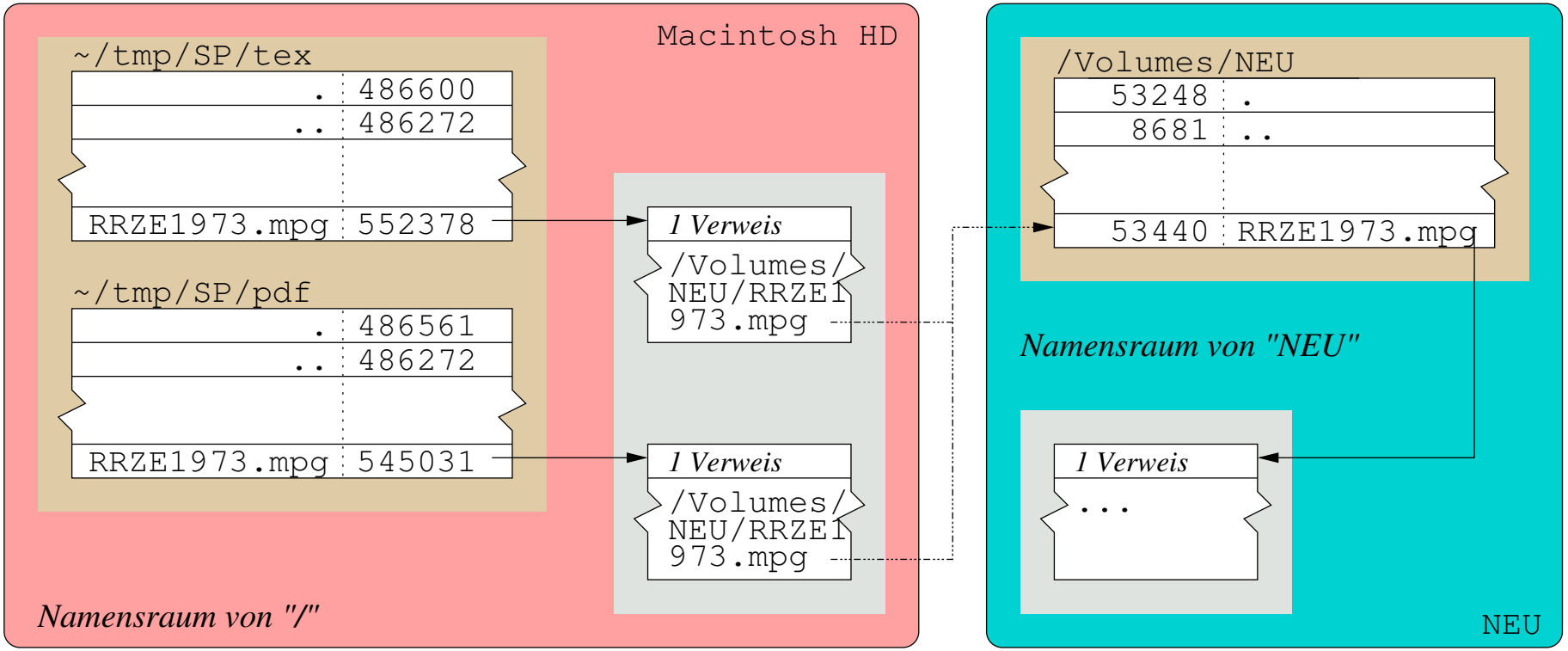
`/ -> /dev/md/dsk/d1`
Wurzel–/Wirtsdateisystem

`usr -> /dev/md/dsk/d2`
Gastdateisystem

`foo -> /dev/md/dsk/c0t6d0s0`
Gastdateisystem

Verweis hinein in einen anderen Namensraum

Symbolische Verknüpfung (engl. *symbolic link*)



Ursprung ist der symbolische Name (engl. *symbolic name*) in Multics [1]

- zur dynamischen Bindung von Namen an (besondere) E/A-Geräte

UNIX Systemfunktionen

Operationen auf Verzeichnisse: Betriebssystem

Linux, MacOS, SunOS

```
fd = creat(path, mode)
    ↦ open(path, O_CREAT | O_TRUNC | O_WRONLY, mode)
ok = link(path1, path2)
ok = symlink(path1, path2)
ok = unlink(path)
ok = mkdir(path, mode)
ok = rmdir(path)
ok = mount(type, dir, flags, data)
ok = umount(dir, flags)
    ⋮
```

UNIX Systemfunktionen (Forts.)

Operationen auf Verzeichnisse: Laufzeitsystem

Linux, MacOS, SunOS

(☞ Aufgabe 5)

```
dirp = opendir(path)
dp   = readdir(dirp)
loc  = telldir(dirp)
void   seekdir(dirp, loc)
void   rewinddir(dirp, loc)
ok   = closedir(dirp)
     ⋮
```

Vorsicht: `readdir(3)`'s Implementierung ist eintrittsvariant

- nebenläufige Ausführung kann Nebeneffekte zur Folge haben
- eintrittsinvariant (engl. *reentrant*) dagegen ist `readdir_r(3)`

Gliederung

Resümee

- **Datei** als Abstraktion von Informationen tragenden Betriebsmitteln
  - das Dateiverzeichnis ist ein Katalog von Dateinamen
  - ein Dateikopf führt Buch über die Systemdaten einer Datei
  - jedem Dateikopf ist eine Dateikopfnummer eindeutig zugeordnet
  - Referenzzähler verwalten Verzeichnis- und Dateiverknüpfungen
- **Dateisystem** als Ablageorganisation eines Datenträgers
  - Namensräume sind flach oder hierarchisch aufgebaut
  - unterschieden werden Wurzel-, Heimat-, Eltern- Arbeitsverzeichnisse
  - symbolische Verknüpfung macht „physikalische Barrieren“ durchlässig
  - Gastdateisysteme lassen sich in Wirtsdateisysteme „montieren“

Literaturverzeichnis

[1] Feiertag, R. J. ; Organick, E. I.:
The Multics Input/Output System.
In: *Proceedings of the Third ACM Symposium on Operating System Principles (SOSP 1971), October 18–20, 1971, Palo Alto, California, USA*, ACM, 1971, S. 35–41

[2] ISO/IEC 14677:
Information technology — Syntactic metalanguage — Extended BNF.
http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf, 1996

[3] *http://minnie.tuhs.org/UnixTree*

Symbolische Verknüpfungen „*Considered Harmful*“

Namen sind Schall und Rauch . . . auf den Inhalt kommt es an!

```
wosch@lorien 1$ mkdir -p Laptop/faui43w; cd Laptop
wosch@lorien 2$ ln -s faui43w lorien
wosch@lorien 3$ ls -l
total 8
drwxr-xr-x  2 wosch  wosch  68 29 Apr 13:01 faui43w
lrwxr-xr-x  1 wosch  wosch   7 29 Apr 13:02 lorien -> faui43w
wosch@lorien 4$ cd lorien
wosch@lorien 5$ cd ..; rmdir faui43w; cd lorien
-bash: cd: lorien: No such file or directory
wosch@lorien 6$ ls -l
total 8
lrwxr-xr-x  1 wosch  wosch   7 29 Apr 13:02 lorien -> faui43w
wosch@lorien 7$ mkdir faui43w; cd lorien
wosch@lorien 8$ ln -s fata\ morgana SOS1
wosch@lorien 9$
```