Die UNIX-Kommandozeile

Kommando [-*Optionen*] [*Argumente*]

Kommando	eingebautes Shell-Kommando oder ausführbare Datei (Programm)
Option	verändert die Grundeinstellung (voreingestellte Funktionalität) des Kommandos
Argument	werden dem Kommando übergeben, meist Namen von zu verarbeitenden Objekten

Beispiele:

```
ls -a -l
ls -al
ls /home
ls -l /home
```

Die Groß- und Kleinschreibung wird unterschieden!

Das UNIX-Dateisystem

- Eine Datei ist ein Speicherbereich auf einem Sekundärspeicher, der durch einen bestimmten Namen angesprochen wird.
- Text- oder Binärdateien sind Datenströme, bestehend aus Bytes, die auf der Festplatte, in Blöcke aufgeteilt, abgespeichert sind. Sie werden *reguläre Dateien (Files)* genannt.
- Dateinamen können bis zu 255 Zeichen lang sein (ohne ASCII-Null und /). Dateinamenendungen (z.B. `.pdf`) werden ignoriert (`doc.pdf.1` erlaubt). Groß- und Kleinschreibung wird unterschieden!
- *Verzeichnisse/Directories* sind selbst Dateien (!), die nichts als die in ihnen verzeichneten Dateien und zu jeder Datei eine *Inode*-Nummer enthalten.
- Das Verzeichnissystem ist *hierarchisch* (baumartig), d.h. jede Datei steht in einem Verzeichnis, das selbst Unterverzeichnis eines Verzeichnisses ist.

  Ausnahme: Wurzelverzeichnis (root) /

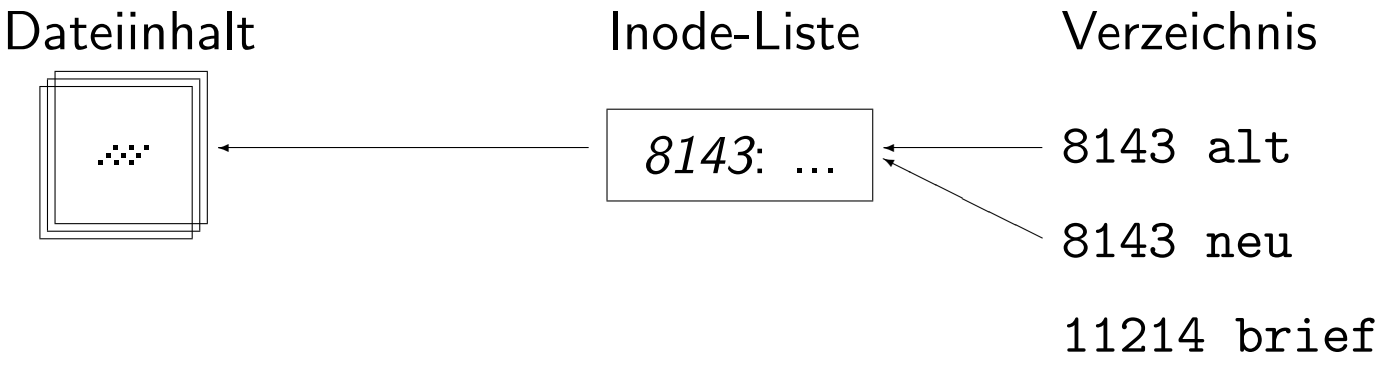
Inode

- eine Datenstruktur mit administrativen Informationen zu der jeweiligen Datei
- in bestimmten Regionen der Festplatte abgelegt
- enthält u.a.
  - Dateityp
  - Dateibesitzer
  - Zugriffsrechte
  - Adressen der Datenblöcke auf der Festplatte

Verzeichnis		Inode-Liste		Datenblöcke
`18395 text`	$\longrightarrow$	*18395:* ...	$\longrightarrow$	... ...
`18701 brief`	$\longrightarrow$	*18701:* ...	$\longrightarrow$	... ...

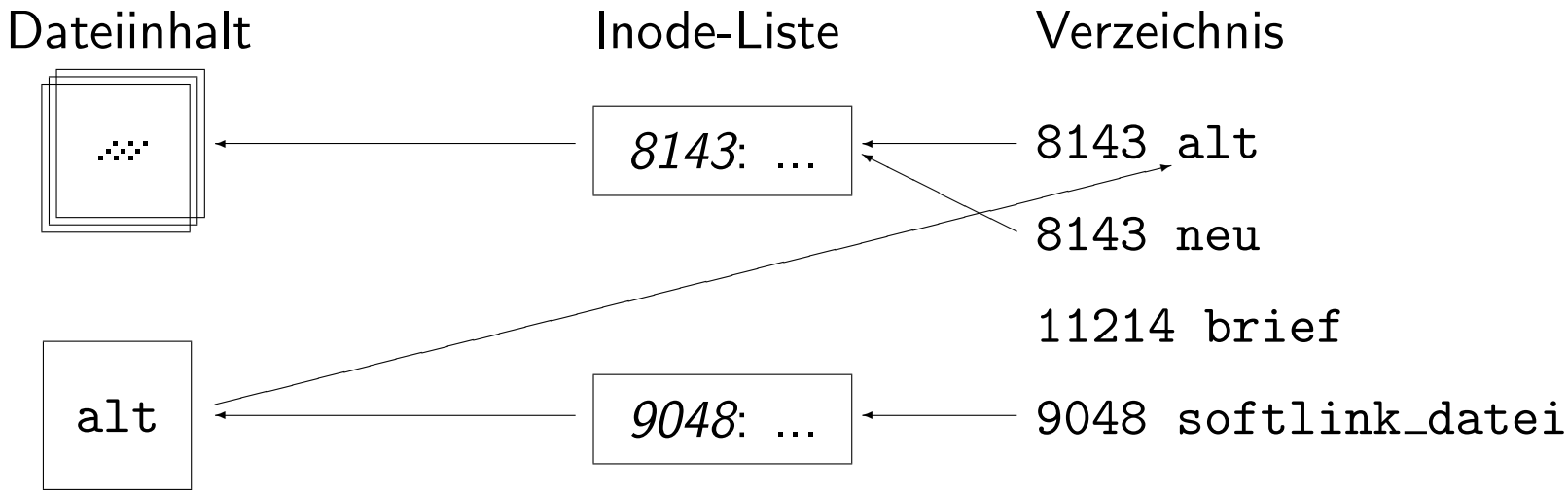
Links

- *Links/Hard-Links:*
  neuer Dateiname für bereits vorhandene Inode
- *symbolische Links/Soft-Links:*
  neue Datei (mit neuer Inode-Nummer), deren Inhalt der Name einer bereits vorhandenen *Ziel*datei ist

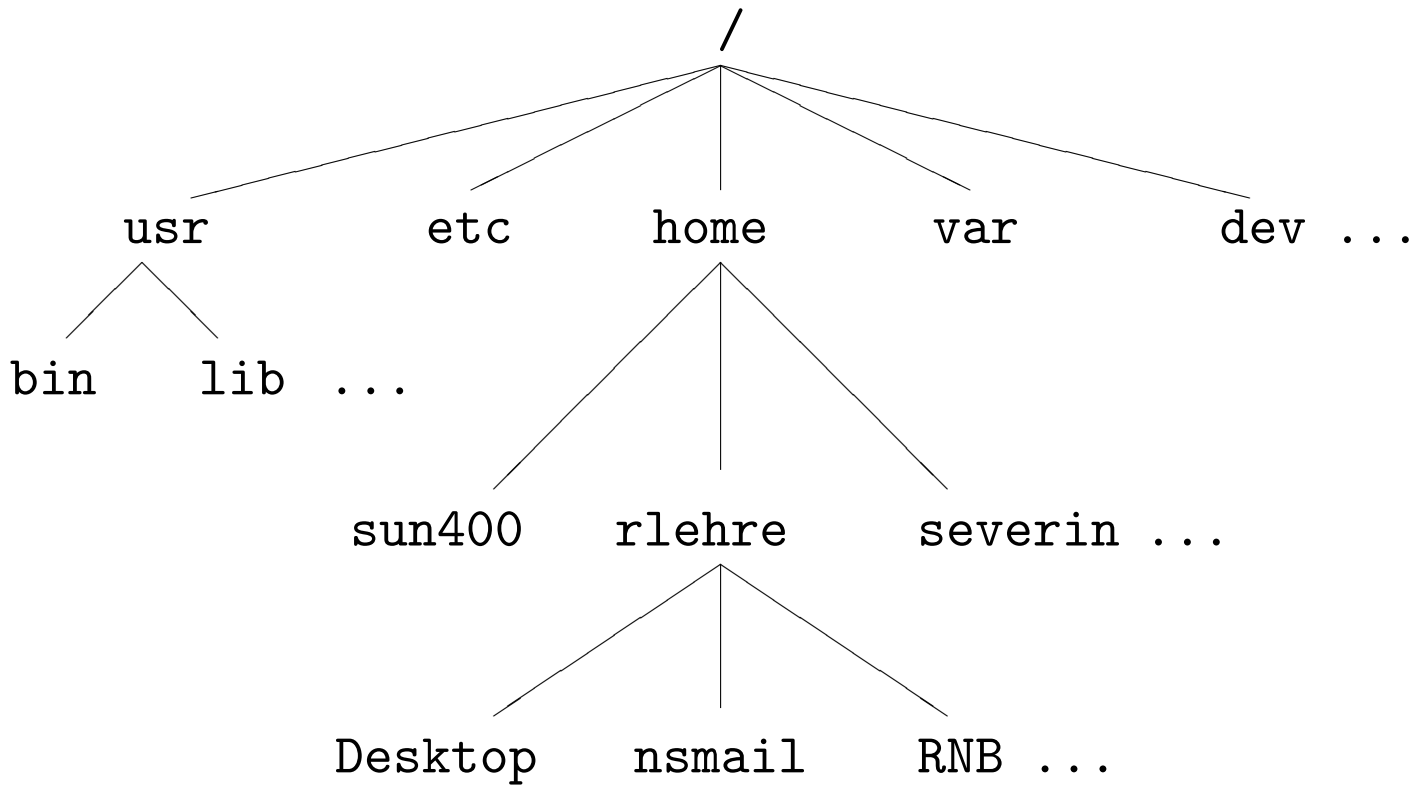


Links

- *Links/Hard-Links:*
  neuer Dateiname für bereits vorhandene Inode
- *symbolische Links/Soft-Links:*
  neue Datei (mit neuer Inode-Nummer), deren Inhalt der Name einer bereits vorhandenen *Ziel*datei ist



Die Verzeichnis-Hierarchie



Standard-Verzeichnisse

`/bin`	Systemkommandos und -tools als ausführbare Binärdateien (`ls`)
`/lib`	Programmbibliotheken
`/usr`	enthält den größten Teil installierter Software
`/etc`	Konfigurationsdateien u.ä. (`/etc/passwd`)
`/home`	Login-Verzeichnisse der Benutzer
`/var`	veränderliche Systemdateien, z.B. Protokolldateien (`/var/adm`), Mailboxen (`/var/mail`), Warteschlangen (`/var/spool`)
`/dev`	Gerätedateien (teilweise in Subdirectories)

Gerätedateien

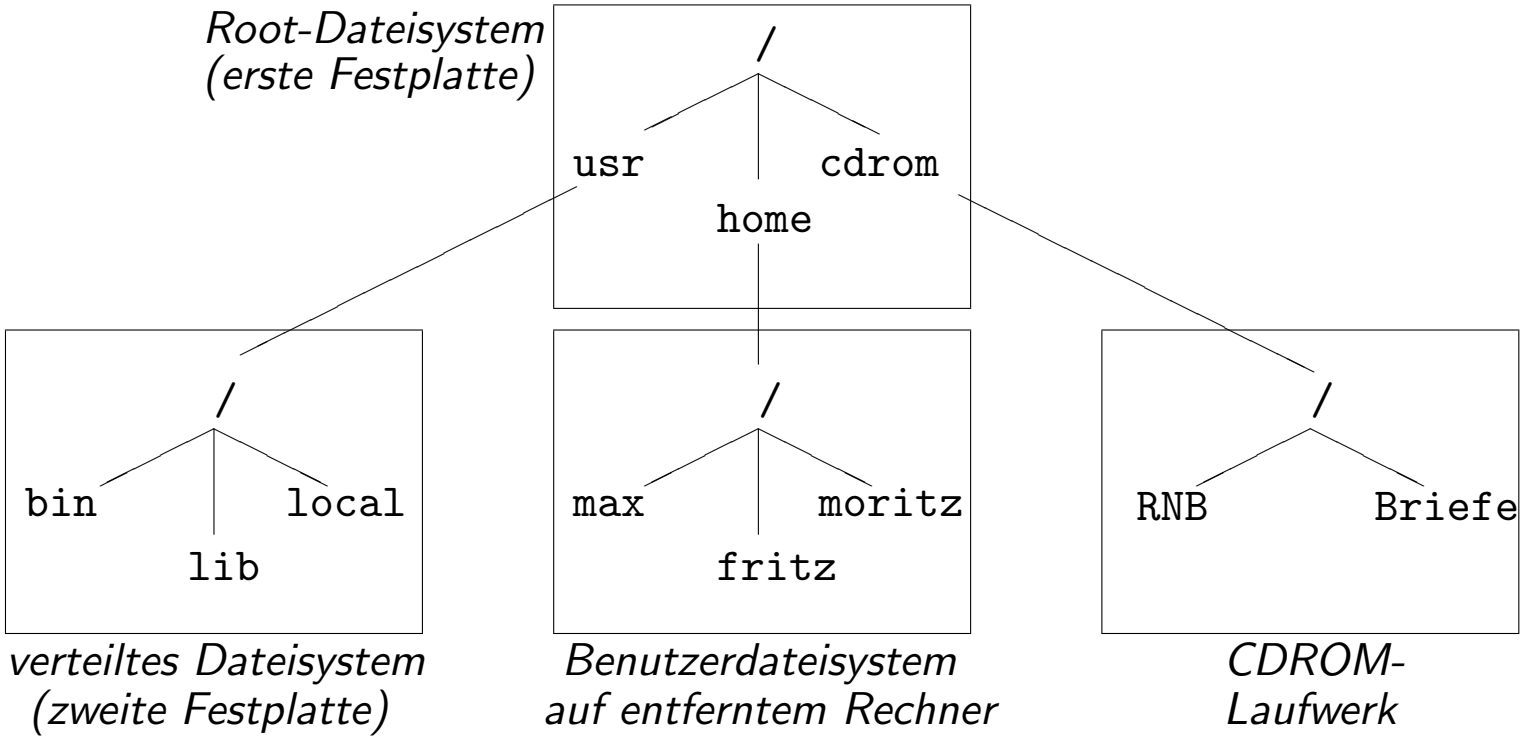
- Gerätedateien sind Spezialdateien ohne eigentlichen Dateninhalt.
  Jedes Gerät (Sekundärspeicher, Plattenpartitionen, Drucker, Terminalfenster, Tastatur etc.) wird vom Dateisystem mit einer Gerätedatei identifiziert.

- Lesen oder Schreiben in Gerätedateien bewirken Ein- bzw. Ausgabe auf dem entsprechenden Gerät.

- Es werden block- und characterorientierte Gerätedateien unterschieden.

Montierte Dateisysteme

Das Dateisystem von UNIX setzt sich aus mehreren Einheiten zusammen, die sich auf verschiedenen Speichermedien befinden können.



Zugriffsrechte

- Drei Klassen von Benutzern beim Zugriff auf Dateien:
  - Besitzer der Datei (***u**ser*)
  - Benutzer in der als Dateiattribut festgelegten Gruppe (***g**roup*)
  - andere Benutzer (***o**thers*)
- Drei Arten von Rechten können vergeben werden:
  - Leserecht (***r**ead*)
  - Schreibrecht (***w**rite*)
  - Ausführungsrecht (*e**x**ecute*)

Bedeutung der Rechte für Verzeichnisse

- **Leserecht:** Auslesen der Informationen über enthaltene Dateien und Unterverzeichnisse
- **Schreibrecht:** Anlegen und Löschen von Dateien (und Unterverzeichnissen) *(unabhängig von den Rechten der verzeichneten Dateien!)*
- **Ausführungsrecht:** Zugriff auf das Verzeichnis oder seine Unterverzeichnisse

  $\rightsquigarrow$ Das Lese- oder Schreibrecht kann nur wirksam erteilt werden, wenn gleichzeitig das Ausführungsrecht vergeben ist!

  $\rightsquigarrow$ Um auf ein Verzeichnis zugreifen zu können, müssen für das Verzeichnis und alle Oberverzeichnisse das Ausführungsrecht vergeben sein! (*durchgängige x-Kette*)

Änderung der Zugriffsrechte

`chmod [-R]` <*wer*><*wie*><*was*> *Datei(en)/Verzeichnis(se)*

wer	Bedeutung
`u`	Modifizierung der Rechte des Besitzers
`g`	Modifizierung der Rechte der Gruppe
`o`	Modifizierung der Rechte für Andere
`a`	Modifizierung der Rechte aller Benutzer
wie	Bedeutung
`+`	Hinzufügen von Rechten
`-`	Entfernen von Rechten
`=`	Ersetzen der Rechte (durch ...)
was	`r`, `w` und/oder `x`

Beispiel mit Kombinationen: `chmod u=rwx,go+r,o-x datei_1 datei_2`

Setzen von Rechten mit Oktalzahlen

400	Leserecht für den Besitzer
200	Schreibrecht für den Besitzer
100	Ausführungsrecht für den Besitzer
040	Leserecht für die Gruppe
020	Schreibrecht für die Gruppe
010	Ausführungsrecht für die Gruppe
004	Leserecht für Andere
002	Schreibrecht für Andere
001	Ausführungsrecht für Andere

Für Kombinationen werden die jeweiligen Oktalzahlen addiert.

Beispiel: `chmod 750 Dir_1` entspricht `chmod u=rwx,g=rx,o= Dir_1`

Zugriffsrechte können nur vom Datei-Eigentümer geändert werden!

Zugriffsrechte für neue Dateien

- `umask` <*Oktalzahl*> gibt an, welche Rechte nicht vergeben werden sollen.
- Als Grundeinstellung (mit `umask 0`) werden Dateien mit Oktal 666 (`rw-rw-rw-`) und Verzeichisse mit Oktal 777 (`rwxrwxrwx`) angelegt.
- In der Oktalzahl von `umask` auftretende Rechte werden von der Grundeinstellung abgezogen.
  (genauer: bitweise UND-Verknüpfung des Wertes der Grundeinstellung und des negierten `umask`-Wertes)
- Beispiel: `umask 022` bewirkt, dass Dateien mit Oktal 644 (`rw-r--r--`) und Verzeichnisse mit Oktal 755 (`rwxr-xr-x`) angelegt werden.
- `umask` (ohne Argument) zeigt die aktuelle Einstellung an.