Grundlagen von Betriebssystemen

Institut für Informatik
Universität Potsdam

Henning Bordihn

Was haben wir heute vor?

- Organisatorisches
- Einführende Bemerkungen zu Betriebssystemen
- Das UNIX-Prozesskonzept
- Das UNIX-Dateissystem

Grundlagen von Betriebssystemen (GBS)

Grundlegende Konzepte von Betriebssystemen am Beispiel von UNIX, wobei der praktische Umgang mit dem System, der Systemkonfiguration und den Systemtools im Mittelpunkt steht.

- Einführung in UNIX als Betriebssystem
- Prozesskonzept, Dateisystem, Editoren in UNIX
- Systemkommandos, Systemdateien und -konfigurationen
- UNIX-Shells und weitere UNIX-Tools
- Shell-Programmierung
- Dienste und Kommunikation im Netzwerk

GBS: Lehrform

- 4 Stunden pro Woche, 7 Wochen lang $\Longrightarrow$ 2 SWS (3 LP)
- **1. Woche:** 2 h Vorlesung + 2 h Übung (*Organisatorisches*)

GBS: Lehrform

- 4 Stunden pro Woche, 7 Wochen lang $\Longrightarrow$ 2 SWS (3 LP)
- **1. Woche:** 2 h Vorlesung + 2 h Übung (*Organisatorisches*)
- **2. bis 7. Woche:** *keine* Vorlesung; (Stoff für) 4 h Übung
- **8. Woche:** 90-minütige Klausur

GBS: Lehrform

- 4 Stunden pro Woche, 7 Wochen lang $\Longrightarrow$ 2 SWS (3 LP)
- **1. Woche:** 2 h Vorlesung + 2 h Übung (*Organisatorisches*)
- **2. bis 7. Woche:** *keine* Vorlesung; (Stoff für) 4 h Übung
- **8. Woche:** 90-minütige Klausur

  $\rightsquigarrow$ 1. Teilprüfung im Modul „Rechner- und Netzbetrieb“
  (Fortsetzung ab 8. Woche durch „Einführung in die Programmierung“
  mit erster Vorlesung am Montag, dem 5. Dezember)

Rechner- und Netzbetrieb (RNB)

- 4-SWS-Modul (6 LP), besteht aus zwei Lehrveranstaltungen

1. Grundlagen von Betriebssystemen (3 LP)
2. Einführung in die Programmierung (3 LP)

- organisatorisch und formal unabhängig
- getrennte Anmeldungen zu den LV und zu den Prüfungen
- inhaltlich stark abhängig

RNB, Forts.

1. Grundlagen von Betriebssystemen
   - 1.–7. Vorlesungswoche
   - grundlegende Kenntnisse, praktische Fertigkeiten zum Umgang mit Betriebssystemen (UNIX)
2. Einführung in die Programmierung
   - ab 8. Vorlesungswoche (erste Vorlesung: 5. Dezember 2011)
   - mehr Konzepte zu Betriebssystemen und Netzwerken,
   - „Programmierung im Kleinen“ mit `C`
   - *setzt die wesentlichen Inhalte aus GBS voraus!*

RNB, Forts.

1. Grundlagen von Betriebssystemen
   - 1.–7. Vorlesungswoche
   - grundlegende Kenntnisse, praktische Fertigkeiten zum Umgang mit Betriebssystemen (UNIX)
2. Einführung in die Programmierung
   - ab 8. Vorlesungswoche (erste Vorlesung: 5. Dezember 2011)
   - mehr Konzepte zu Betriebssystemen und Netzwerken,
   - „Programmierung im Kleinen“ mit `C`
   - *setzt die wesentlichen Inhalte aus GBS voraus!*

- *Sommersemester:* Programmierung
  - objektorientierte Programmierung
  - setzt RNB voraus!

GBS: Übungen

- Gruppe 1: dienstags ab 16.15 Uhr
  Gruppe 2: mittwochs ab 12.15 Uhr
  Gruppe 3: freitags ab 12.15 Uhr

GBS: Übungen

- Gruppe 1: dienstags ab 16.15 Uhr
  Gruppe 2: mittwochs ab 12.15 Uhr und ab 14.15 Uhr (1. Woche)
  Gruppe 3: freitags ab 12.15 Uhr

GBS: Übungen

- Gruppe 1: dienstags ab 16.15 Uhr
  Gruppe 2: mittwochs ab 12.15 Uhr und ab 14.15 Uhr (1. Woche)
  Gruppe 3: freitags ab 12.15 Uhr

- selbständige Erarbeitung des Stoffs anhand von Aufgaben
  www.cs.uni-potsdam.de/ml (*ausgedruckt mitbringen!!!*)

GBS: Übungen

- Gruppe 1: dienstags ab 16.15 Uhr
  Gruppe 2: mittwochs ab 12.15 Uhr und ab 14.15 Uhr (1. Woche)
  Gruppe 3: freitags ab 12.15 Uhr

- selbständige Erarbeitung des Stoffs anhand von Aufgaben
  www.cs.uni-potsdam.de/ml (*ausgedruckt mitbringen!!!*)

- Stoff für 4 Stunden

- Betreuung durch Tutoren im Labor (mindestens) in den ersten 2 Stunden jeder Übung

- eigenständiges Nacharbeiten nicht geschaffter Aufgaben

Prüfungsleistung

- 90-minütige Klausur in der 8. Vorlesungswoche:

  Freitag, 9. Dezember, 16.15 – 17.45 Uhr, Hörsaal 03 im Haus 6

- Anmeldung zur Klausur über PULS spätestens 8 Werktage vorher

- *jetzt:* Anmeldung zur LV in genau einer der Gruppen

Was haben wir heute vor?

- Organisatorisches
- Einführende Bemerkungen zu Betriebssystemen
- Das UNIX-Prozesskonzept
- Das UNIX-Dateissystem

Betriebssystem

- Basissoftware, die den Betrieb eines Rechners ermöglicht
- übernimmt Aufgaben wie zum Beispiel
  - Steuerung der Hardware,
  - Speicherverwaltung,
  - Verwaltung der Prozesse,
  - Nutzerverwaltung,
  - Kommandointerpretation (Schnittstelle Benutzer – Rechner)

Charakteristika moderner Betriebssysteme

Charakteristika moderner Betriebssysteme

- Multiprogramming: Halten mehrerer Programme gleichzeitig im Hauptspeicher;

Charakteristika moderner Betriebssysteme

- Multiprogramming: Halten mehrerer Programme gleichzeitig im Hauptspeicher;
- Multitasking: Anwendung des Multiprogramming, wobei mehrere Prozesse gleichzeitig verwaltet werden, denen die CPU abwechselnd (zeit- oder ereignisgesteuert) zugeteilt wird;

  $\rightsquigarrow$ Time-Sharing, Scheduling, Pseudoparallelität

Charakteristika moderner Betriebssysteme

- Multiprogramming: Halten mehrerer Programme gleichzeitig im Hauptspeicher;
- Multitasking: Anwendung des Multiprogramming, wobei mehrere Prozesse gleichzeitig verwaltet werden, denen die CPU abwechselnd (zeit- oder ereignisgesteuert) zugeteilt wird;

  $\rightsquigarrow$ Time-Sharing, Scheduling, Pseudoparallelität

- Multiuser-Betrieb: Anmeldung meherer Nutzer gleichzeitig möglich

Charakteristika moderner Betriebssysteme

- Multiprogramming: Halten mehrerer Programme gleichzeitig im Hauptspeicher;
- Multitasking: Anwendung des Multiprogramming, wobei mehrere Prozesse gleichzeitig verwaltet werden, denen die CPU abwechselnd (zeit- oder ereignisgesteuert) zugeteilt wird;

  $\rightsquigarrow$ Time-Sharing, Scheduling, Pseudoparallelität

- Multiuser-Betrieb: Anmeldung meherer Nutzer gleichzeitig möglich
- Multiprozessing: für Rechner-Architekturen mit mehreren Prozessoren

Charakteristika moderner Betriebssysteme

- Multiprogramming: Halten mehrerer Programme gleichzeitig im Hauptspeicher;
- Multitasking: Anwendung des Multiprogramming, wobei mehrere Prozesse gleichzeitig verwaltet werden, denen die CPU abwechselnd (zeit- oder ereignisgesteuert) zugeteilt wird;

  $\rightsquigarrow$ Time-Sharing, Scheduling, Pseudoparallelität

- Multiuser-Betrieb: Anmeldung meherer Nutzer gleichzeitig möglich
- Multiprozessing: für Rechner-Architekturen mit mehreren Prozessoren
- Netzwerkbetriebssysteme: für Rechenanlagen, die in lokale Netze eingebunden sind und den Betrieb von Server- und Clientcomputern unterstützen

Lokale Netze (1)

Bezeichnung	Enfernung zw. Prozessoren
Local-Area-Network (LAN)	meist 1m bis 10 km
Wide-Area-Network (WAN)	meist über 10 km

Lokale Netze (1)

Bezeichnung	Enfernung zw. Prozessoren
Local-Area-Network (LAN)	meist 1m bis 10 km
Wide-Area-Network (WAN)	meist über 10 km

?

Lokale Netze (1)

Bezeichnung	Enfernung zw. Prozessoren
Local-Area-Network (LAN)	meist 1m bis 10 km
Wide-Area-Network (WAN)	meist über 10 km

?

- LAN meist in Gebäuden, Gebäudekomplexen, z.B. innerhalb eines Betriebes
- Hauptzweck eines LAN: gemeinsame Nutzung von Betriebsmitteln (Dateien, Rechenleistung, Drucker, allgemeine Dienste etc.)
- *Server* stellen die Betriebsmittel bereit, z.B.
  - Dateien (Fileserver),
  - Rechenleistung (Applikationsserver),
  - Geräte (Printserver),
  - allgemeine Dienste (Mail-, WWW-Server)

Lokale Netze (2)

- Von *Clients* aus können diese Betriebsmittel genutzt werden.
- Das Zusammenspiel von Clients und Servern, die im LAN miteinander verbunden sind, erfordert spezielle Software (Netzwerkbetriebssystem).

Lokale Netze (2)

- Von *Clients* aus können diese Betriebsmittel genutzt werden.

- Das Zusammenspiel von Clients und Servern, die im LAN miteinander verbunden sind, erfordert spezielle Software (Netzwerkbetriebssystem).
  - Server stellen ihre Betriebsmittel den anderen Rechnern im Netz bereit, warten auf Anfragen der Clients, verwalten die Ressourcen
  - Clients stellen Anfragen an die Server und machen deren Betriebsmittel verfügbar, als wären sie lokal vorhanden
  - Netzwerkbetriebssysteme sind immer Multiuser-Betriebssysteme
    $\rightsquigarrow$ Zugangskontrolle für Benutzer,
    $\rightsquigarrow$ Verwaltung der Zugangsberechtigung zu Betriebsmitteln

Das Betriebssystem UNIX

- UNIX: eingetragenes Warenzeichen von *The Open Group*[1]
  $\rightsquigarrow$ Festlegung einer Spezifikation für ein Betreibssystem

- UNIX-System: ein Betriebssystem, das dieser Spezifikation gerecht wird

- Trennung von Warenzeichen und Code, so dass mehrere Implementierungen möglich sind

  $\rightsquigarrow$ Es gibt kein einheitliches UNIX (den Umfang und die Interna betreffend).

[1] http://www.opengroup.org

UNIX-Versionen und -Derivate

- Zwei wesentliche Entwicklungslinien:
  1. AT&T: Version 1, ... Version 7, System III, ... , System V.4
  2. Berkeley: 1BSD, ... , 4.4BSD (*Berkeley Software Distribution*)

  $\rightsquigarrow$ dominierend im Serverbereich
- Entwicklung weiterer UNIX-Systeme für die Hardware verschiedener Hersteller: HP-UX, IMB AIX, **Solaris** (ein BSD-Derivat, entwickelt aus SUN-OS) etc.
- POSIX: ein Standard, der gewisse Interna von UNIX-Systemen (vor allem die Systemaufrufe) vereinheitlicht

LINUX

Für PC's und Notebooks: LINUX-Distributionen, z.B. GNU/LINUX

- LINUX bezeichnet nur den Kern eines Betriebssystems
  (1991 von finnischem Studenten Linus Torvalds programmiert)
- weitere System-Software nötig, verbreitet die Software des GNU-Projektes
- Der LINUX-Kernel und die GNU-Software mit ihren Quellcodes stehen als *Open Source* kostenlos zur Verfügung. (Weiterentwicklung, Anpassung an neue Hardware und Fehlerkorrekturen leicht möglich)
- Unabhängige Organisatoren (*Distributoren*) sammeln und vertreiben Software für LINUX.

  $\rightsquigarrow$ In einer Distribution sind (fast) alle Programme enthalten, die man normalerweise braucht (Compiler, Editoren, Office-Programme etc.), z.B.:

  `www.suse.de`, `www.ubuntu.com`, `www.redhat.com`, (`www.knoppix.de`)

Eigenschaften von UNIX

- Multitasking-Betriebssystem
- Multiuser-Betriebssystem
- Multiprozessor-Betriebssystem
- Netzwerkbetriebssystem
- sehr stabil, gute Portabilität, weite Verbreitung
- besitzt kommandobasierte und graphische Benutzeroberflächen
- Aufbau:
  - Betriebssystemkern (*kernel*)
  - Systemprogramme
  - Anwendungsprogramme

Aufgabenverteilung bei UNIX

- **Kern:** Programme, die in einem geschützten Modus arbeiten, z.B. zur
  - Steuerung der Hardware,
  - Prozessverwaltung einschl. Scheduling
  - Speicherverwaltung

Aufgabenverteilung bei UNIX

- **Kern:** Programme, die in einem geschützten Modus arbeiten, z.B. zur
  - Steuerung der Hardware,
  - Prozessverwaltung einschl. Scheduling
  - Speicherverwaltung
- **Systemprogramme:** zur Bedienung der Mechanismen des Kerns, die über *Systemaufrufe* angesprochen werden, z.B. zur
  - Realisierung der Multiuser-Umgebung
  - System-Anmeldungen, Benutzererkennung,
  - Kommandointerpreter (Shell)

Aufgabenverteilung bei UNIX

- **Kern:** Programme, die in einem geschützten Modus arbeiten, z.B. zur
  - Steuerung der Hardware,
  - Prozessverwaltung einschl. Scheduling
  - Speicherverwaltung
- **Systemprogramme:** zur Bedienung der Mechanismen des Kerns, die über *Systemaufrufe* angesprochen werden, z.B. zur
  - Realisierung der Multiuser-Umgebung
  - System-Anmeldungen, Benutzererkennung,
  - Kommandointerpreter (Shell)
- **Anwendungsprogramme**, wie Editoren und Compiler

Was haben wir heute vor?

- Organisatorisches
- Einführende Bemerkungen zu Betriebssystemen
- Das UNIX-Prozesskonzept
- Das UNIX-Dateissystem

Das UNIX-Prozesskonzept

- Beim Start eines Programms durch einen Benutzer erzeugt der Kern einen Prozess.

Das UNIX-Prozesskonzept

- Beim Start eines Programms durch einen Benutzer erzeugt der Kern einen Prozess.
- Wird dasselbe Programm mehrfach gestartet, so entsteht jeweils ein eigener Prozess.

Das UNIX-Prozesskonzept

- Beim Start eines Programms durch einen Benutzer erzeugt der Kern einen Prozess.
- Wird dasselbe Programm mehrfach gestartet, so entsteht jeweils ein eigener Prozess.
- Das UNIX-Prozesskonzept ist *hierarchisch*, d.h., jeder Prozess hat einen Elternprozess, der ihn erzeugt.

  Ausnahme: *Initializer*, den das System automatisch beim Booten erzeugt

Das UNIX-Prozesskonzept

- Beim Start eines Programms durch einen Benutzer erzeugt der Kern einen Prozess.
- Wird dasselbe Programm mehrfach gestartet, so entsteht jeweils ein eigener Prozess.
- Das UNIX-Prozesskonzept ist *hierarchisch*, d.h., jeder Prozess hat einen Elternprozess, der ihn erzeugt.

  Ausnahme: *Initializer*, den das System automatisch beim Booten erzeugt
- Verwaltung der *Prozessattribute* in einer Prozesstabelle:
  - jede Zeile repräsentiert einen Prozess
  - die Spalten beinhalten die Prozessattribute
  - kann mit dem Systemkommando `ps` abgefragt werden

Prozessattribute

- Adressräume des Programms im Primär- und Sekundärspeicher,
- Inhalte der CPU-Register (Befehlszähler),
- Prozesszustand (running/sleeping/ready),
- bislang verbrauchte Zeit (TIME),
- Prozesskennung (PID),
- Elternprozesskennung (PPID),
- Programmname (CMD),
- Benutzerkennung (UID),
- Terminal (TTY)

Was haben wir heute vor?

- Organisatorisches
- Einführende Bemerkungen zu Betriebssystemen
- Das UNIX-Prozesskonzept
- Das UNIX-Dateissystem

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:

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

Die Groß- und Kleinschreibung wird unterschieden!

UNIX-Kommandos zur Arbeit im Dateisystem

`pwd`	Anzeigen des aktuellen Verzeichnisses (**Arbeitsverzeichnis**)
`ls` *name*	**Auflisten** aller Dateien im Verzeichnis (Ordner) *name* (`ls` ohne Argument: aller Dateien im Arbeitsverzeichnis)
`cp` *datei1 datei2*	**Kopieren** von *datei1* in *datei2* (Achtung: verdoppelt die Bytes auf dem Datenträger)
`mv` *datei1 datei2*	**Umbenennen** von *datei1* in *datei2*
`rm` *datei*	**Löschen** von *datei*
`cd` *name*	**Wechsel des Arbeitsverzeichnisses** nach *name*
`mkdir` *name*	**Verzeichnis anlegen** mit dem Namen *name*
`rmdir` *name*	**Verzeichnis löschen** mit dem namen *name* (Achtung: Verzeichnis *name* muss leer sein!)

Das UNIX-Dateisystem

- Eine Datei ist ein Speicherbereich auf einem Sekundärspeicher, der durch einen bestimmten *Namen* angesprochen wird.
- Dateinamen können bis zu 255 Zeichen lang sein (ohne ASCII-Null und /).
  Dateinamenendungen (z.B. `.pdf`) für UNIX bedeutungslos (`doc.pdf.1` erlaubt).
  Groß- und Kleinschreibung wird unterschieden!

Das UNIX-Dateisystem

- Eine Datei ist ein Speicherbereich auf einem Sekundärspeicher, der durch einen bestimmten *Namen* angesprochen wird.
- Dateinamen können bis zu 255 Zeichen lang sein (ohne ASCII-Null und /).
  Dateinamenendungen (z.B. `.pdf`) für UNIX bedeutungslos (`doc.pdf.1` erlaubt).
  Groß- und Kleinschreibung wird unterschieden!
- Text- oder Binärdateien sind Datenströme, bestehend aus Bytes, die auf der Festplatte, in Blöcke aufgeteilt, abgespeichert sind.
  Sie werden *reguläre Dateien (Files)* genannt.

Das UNIX-Dateisystem

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

Das UNIX-Dateisystem

- Eine Datei ist ein Speicherbereich auf einem Sekundärspeicher, der durch einen bestimmten *Namen* angesprochen wird.
- Dateinamen können bis zu 255 Zeichen lang sein (ohne ASCII-Null und /).
  Dateinamenendungen (z.B. `.pdf`) für UNIX bedeutungslos (`doc.pdf.1` erlaubt).
  Groß- und Kleinschreibung wird unterschieden!
- Text- oder Binärdateien sind Datenströme, bestehend aus Bytes, die auf der Festplatte, in Blöcke aufgeteilt, abgespeichert sind.
  Sie werden *reguläre Dateien (Files)* genannt.
- 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.

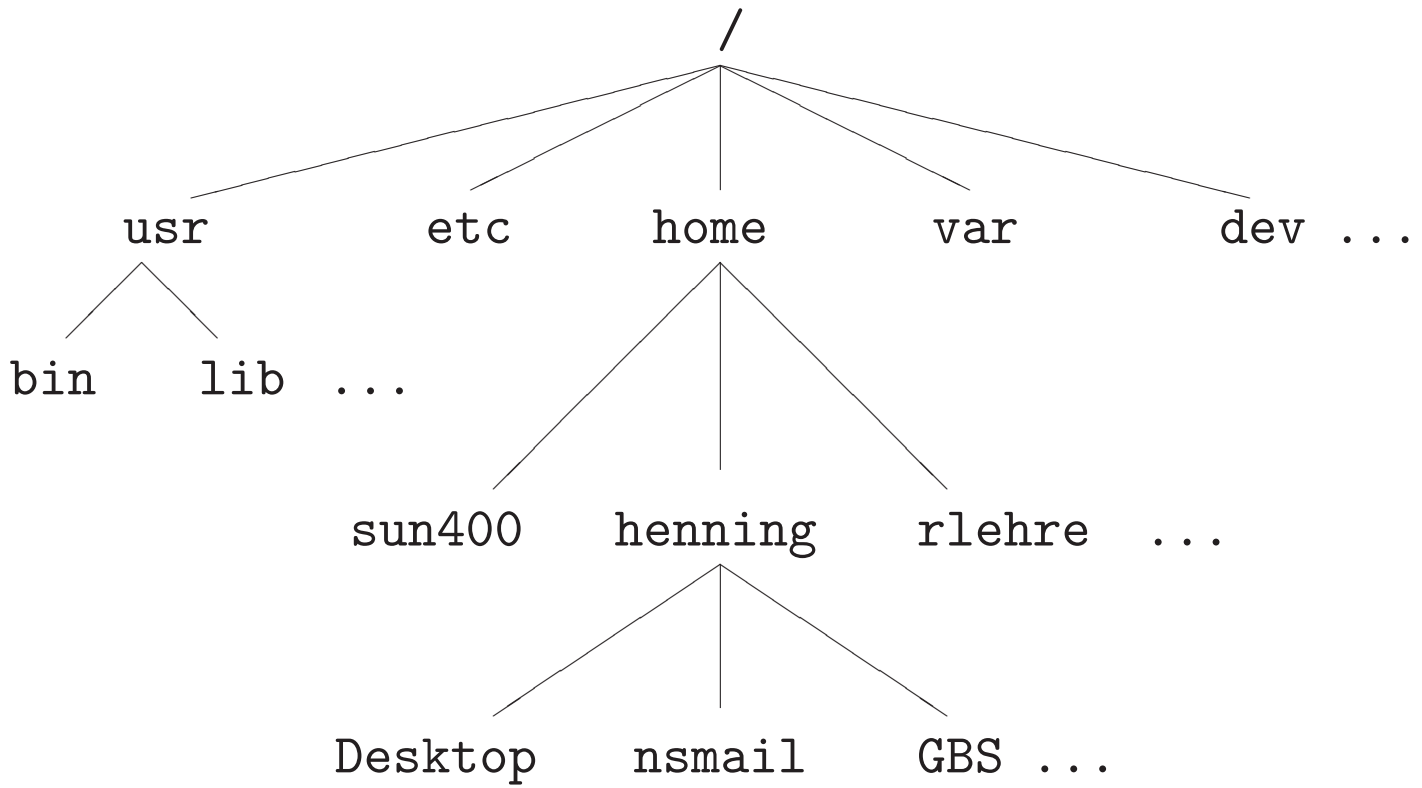
  Ausnahme: Wurzelverzeichnis (root) /

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:* ...	$\longrightarrow$	... ...

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`	enthält **Login-Verzeichnisse** der Benutzer (Login-Verzeichnis: benutzereigener Standort im Verzeichnisbaum nach jedem Login)
`/var`	veränderliche Systemdateien, z.B. Protokolldateien (`/var/adm`), Mailboxen (`/var/mail`), Warteschlangen (`/var/spool`)
`/dev`	Gerätedateien (teilweise in Subdirectories)

Gerätedateien

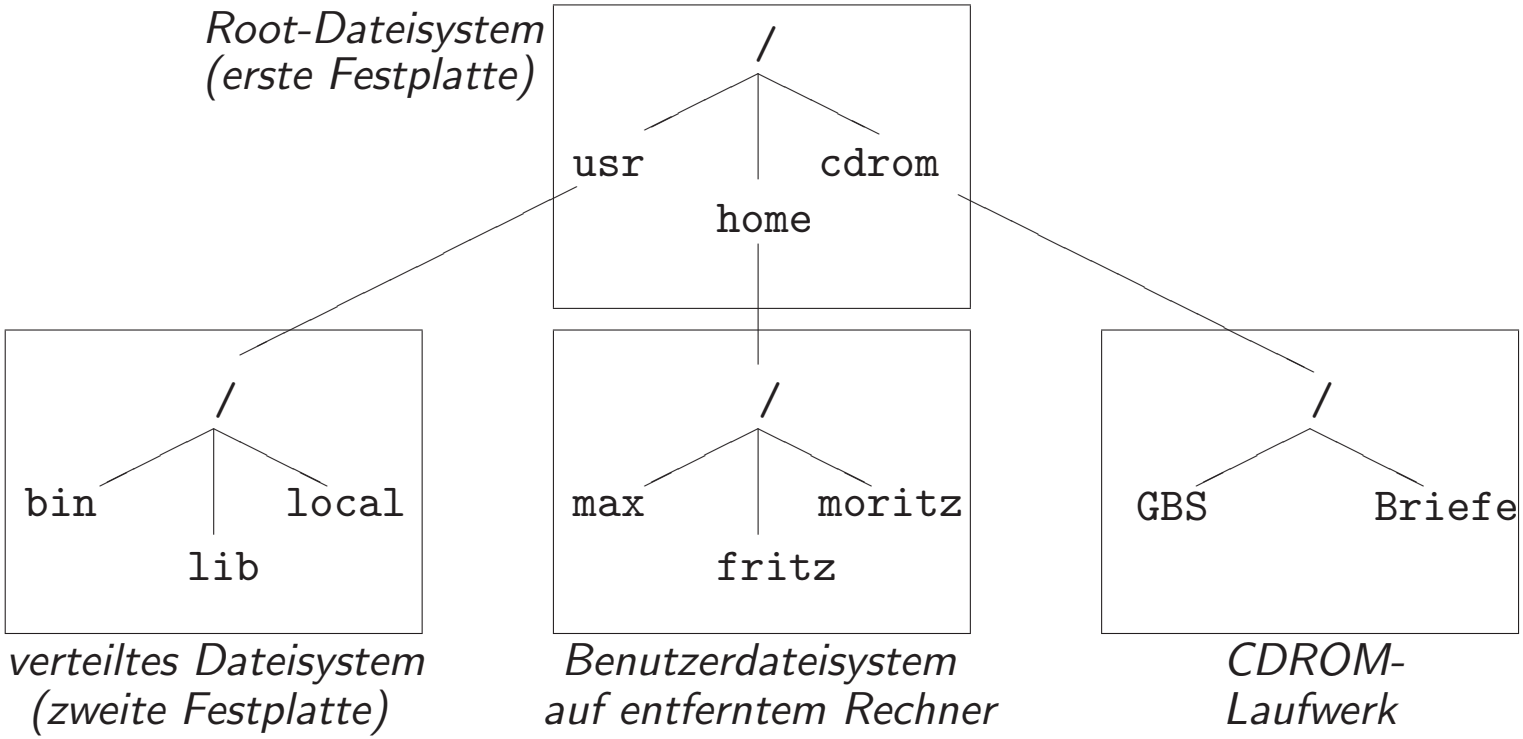
- Gerätedateien sind Spezialdateien ohne eigentlichen Dateninhalt.
  Jedes Gerät (Sekundärspeicher, Plattenpartitionen, Drucker, Terminalfenster, Tastatur etc.) wird vom Dateisystem mit einer Gerätedatei identifiziert.

- 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.



Absolute und relative Pfadnamen

In Kommandos wie `ls`, `cd`, `cp` usw. können die Argumente auf zwei Arten angegeben werden:

Absolute und relative Pfadnamen

In Kommandos wie `ls`, `cd`, `cp` usw. können die Argumente auf zwei Arten angegeben werden:

- als absoluter Pfad mit `/` beginnend,

```
ls /home
ls /home/henning/Lehre/GBS
```

Absolute und relative Pfadnamen

In Kommandos wie `ls`, `cd`, `cp` usw. können die Argumente auf zwei Arten angegeben werden:

- als absoluter Pfad mit `/` beginnend,

```
ls /home
ls /home/henning/Lehre/GBS
```

- als relativer Pfad, der im Arbeitsverzeichnis beginnt

```
ls ..
ls Lehre/GBS
```

Absolute und relative Pfadnamen

In Kommandos wie `ls`, `cd`, `cp` usw. können die Argumente auf zwei Arten angegeben werden:

- als absoluter Pfad mit `/` beginnend,

```
ls /home
ls /home/henning/Lehre/GBS
```

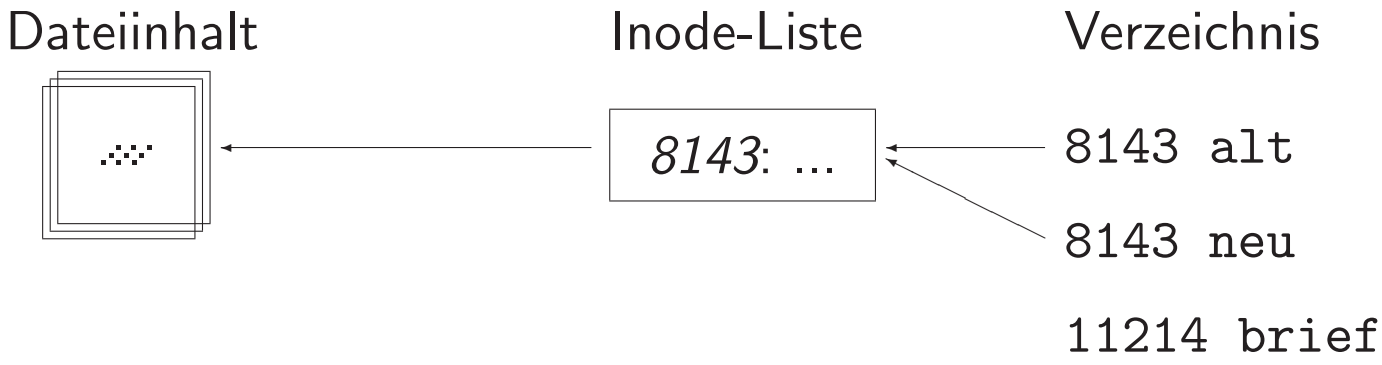
- als relativer Pfad, der im Arbeitsverzeichnis beginnt

```
ls ..
ls Lehre/GBS
```

- Verzeichnistrenner: `/`
- Oberverzeichnis: `..`
- Arbeitsverzeichnis: `.`

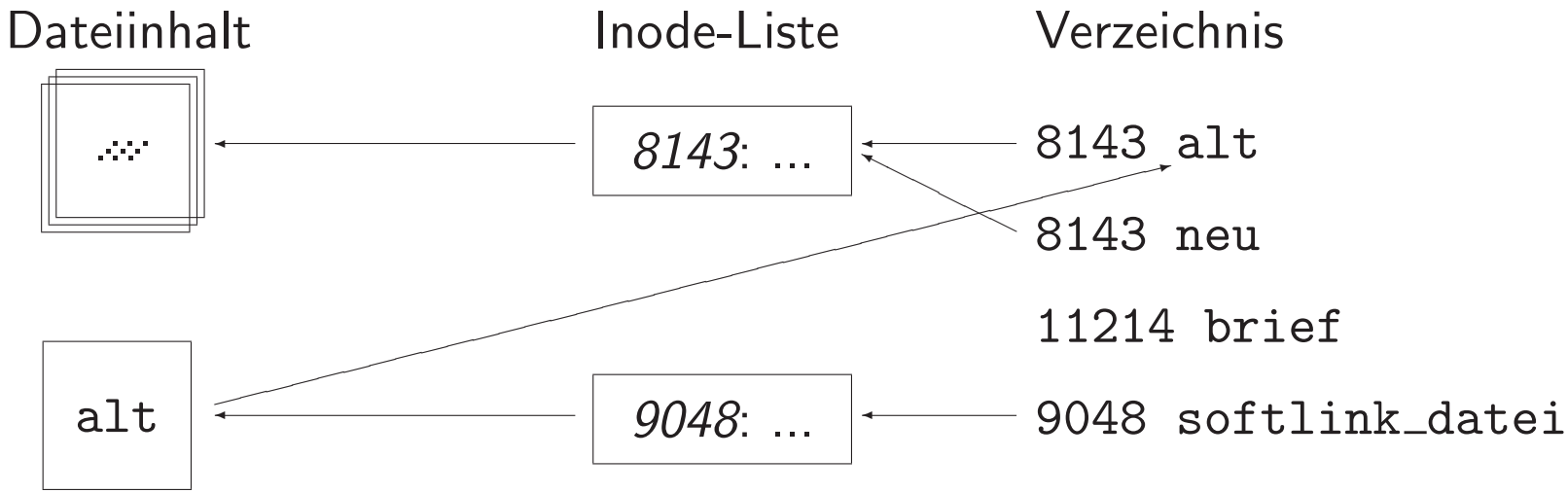
Links

- *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

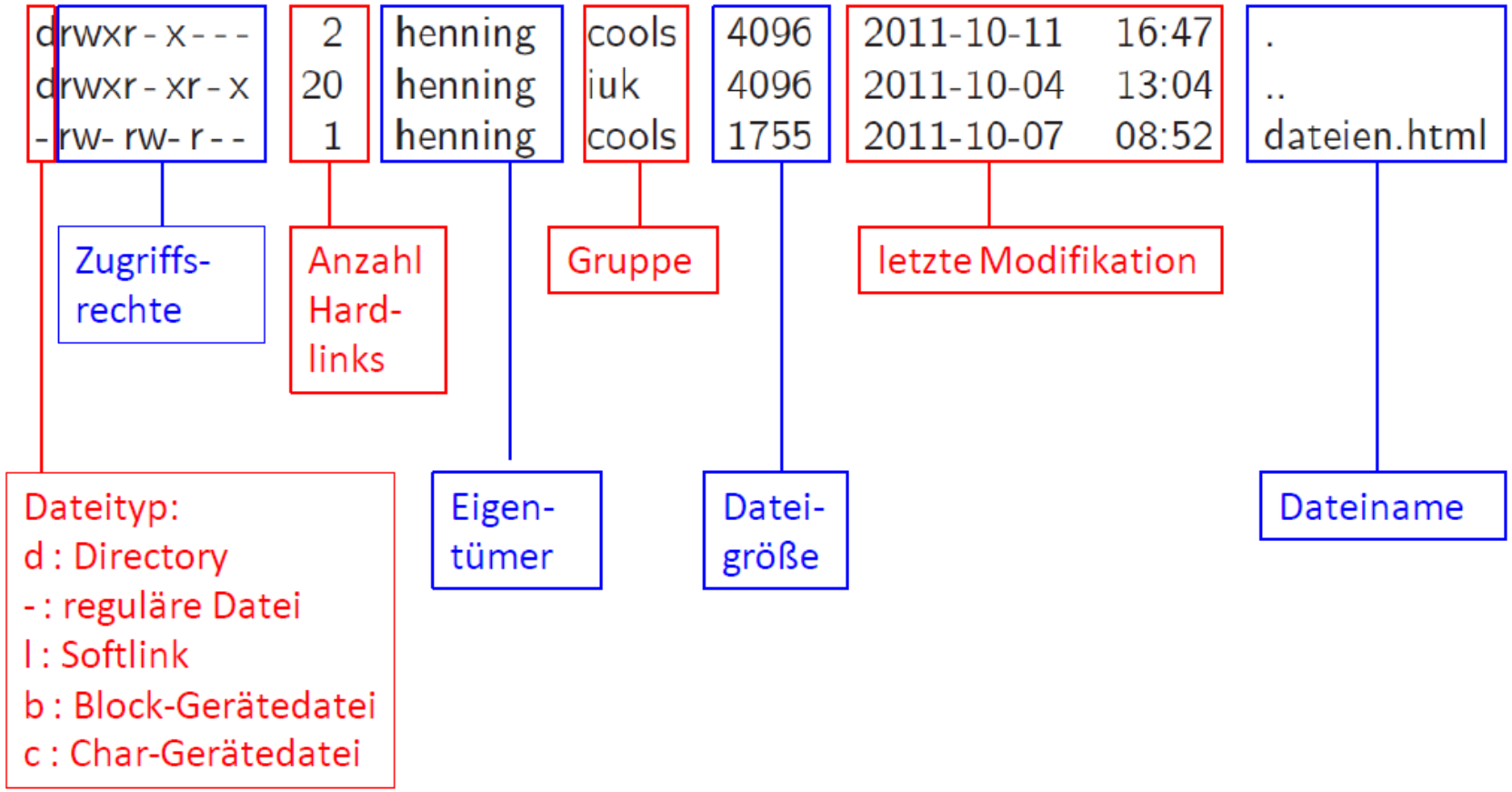


Links

- *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



Dateiattribute bei `ls -al`



Versteckte Dateien

- bginnen mit einem Punkt `.`
- beim Erzeugen eines Verzeichnisses werden automatisch angelegt:
  - `.` als Hardlink auf das neue Verzeichnis
  - `..` als Hardlink auf das Oberverzeichnis (Elternverzeichnis)
- werden bei `ls` mit der Option `-a` angezeigt

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

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!)*

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.

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.

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)

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.

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.