

Shell:

Zugang: GUI Terminal-Fenster, Text-Mode-Konsole,
ssh (via Netz, unter Windows: **putty**)

Verschiedene Shells („Shell“ = Commandline-Interpreter):

- sh-Familie: **sh**, **ksh**, **bash** ("Klassische" Shell, AT & T)
 - csh-Familie: **csh**, **tcsh** (Ursprung: BSD Unix)
 - **zsh** (Teile von beidem)
 - **dash**, busybox-Shell: Minimal-Shells (z.B. für Embedded Systems)
- ==> Bei interaktiver Nutzung zu 90 % gleich, bei Shell-Skripten sehr verschieden!
- ==> Vom Admin pro User festlegbar
- Heute in Verwendung: Zu >90 % **bash**

Mehrere Arten von Befehlen:

- Normalfall: **Normale Programme** bzw. Anwendungen:
Befehl = Name eines **Exe-Files von Platte**
- Shell **Builtins** (= Kommandos, die die Shell selbst macht, z.B. **cd**)
- Shell-Befehle (**for**, **if**, ...)
- **Aliases**, Shell-Funktionen, ... (selbst definierbar)

Unterscheidung mit Befehl "**type** ..."

Exe-Files werden in den Dir's gesucht, die in **PATH** stehen
(. steht normalerweise nicht in **PATH**)
==> **prog** in aktuellem Verzeichnis explizit **./prog** starten!)

Hilfe: man (Welche Sektionen? 1, 8)

Schnellster Ausstieg mit Ctrl/D (= End of File), sonst **exit** oder **logout**

"Superuser" (Admin) **root** (bzw. jeder User mit Uid **0**) darf alles,
alle anderen sind "normaler" User

- User anzeigen mit **id** (==> primäre und sekundäre Gruppen), **whoami**
- Sessions anzeigen mit **who**
- **/etc/passwd** anschauen

Rechnername: **hostname**

Betriebssystem usw: **uname**, **uname -a**

Alte Befehlszeile holen, Befehlszeile editieren mit Cursor-Tasten usw.,
alten Befehls-Anfang suchen mit PageUp (hoffentlich eingestellt?)

Filennamen:

- Trennzeichen / (nicht \ wie bei Windows!)
- Unterschied absolute (beginnen mit /) und relative Pfade (beginnen mit name, ., . ., ~)
==> / ist das Wurzel-Verzeichnis
- Bedeutung . und . ., aktuelles Verzeichnis
- Befehl: **pwd**
- Begriff Home-Verzeichnis (beim Anmelden, pro Benutzer!),
abgekürzt ~ (nur am Anfang eines Pfades!)
auch: ~user_name ... Home-Verzeichnis eines anderen Users
- Befehl: **cd** (Tricks:
Nur **cd** ==> Home-Verzeichnis
cd - ==> Voriges aktuelles Verzeichnis
cd . . ==> eins rauf
- Keine Extensions nötig, Exe-Files heißen nicht .exe
- Bedeutung von . am Anfang eines File- oder Directory-Namens:
 - „hidden“ (bei **ls** nur mit **-a** angezeigt)
 - meist für interne Dateien von irgendwelchen Programmen unter Home
 - Spielen bei *-Expansion nicht mit
(Achtung: werden bei "**cp** *" usw. nicht mitkopiert!)
- Filennamen-Wildcards
 - * (steht für 0 oder mehr bel. Zeichen)
 - ? (steht für genau 1 bel. Zeichen)
 - [...] (mehrere Zeichen zur Wahl)
 - und weitere
- Wildcard-Expansion und Expansion von ~ für Home
macht die Shell vor dem Start des Programms (Unterschied zu DOS!),
geschieht nicht durch das aufgerufene Programm
(das Programm sieht die bereits expandierten Filennamen)
- Wildcards passen auf File- und Verzeichnis-Namen, kann man nicht unterscheiden!
- Quoting mit \ (ein Ziechen), ' oder "
 - Für Sonderzeichen und Zwischenräume in Filennamen
 - Verhindert Wildcard-Expansion, Variable-Expansion, ...
(z.B. wenn man * an ein Programm übergeben will, kommt u.a. bei Befehl **find** vor)
- Schnelles Eintippen von Dateinamen mit Tab (Namens-Vervollständigung)

Konzept des einheitlichen Verzeichnis-Baumes statt Laufwerks-Buchstaben:

- **mount** und **df** anschauen
- "Echte" Filesysteme (**/**, **/home**)
 - tmpfs**-Filesysteme = "RAM-Disk" (beim Reboot weg)
 - Pseudo-Filesysteme des Kernels (**/dev**, **/sys**, **/proc**) (Skript S. 40)
- Root-Filesystem **/**: "Oberstes" Filesystem, erstes und einziges Filesystem beim Booten (Restliche Filesysteme werden erst später durch Start-Skripte gemountet)
Heute oft erst beim zweiten Boot-Schritt:
 - Beim ersten Boot-Schritt ist die "initrd" das Root-Filesystem
 - initrd ist ein kleines, vor-initialisiertes RAM-Filesystem
 - Wird mit dem Kernel ausgeliefert und mit dem Kernel beim Booten geladen
 - Enthält u.a. modular nachladbare Device-Treiber für Disk-I/O-Chips
 - Wenn wichtigste Treiber usw. geladen: Wird durch echtes Root-Filesystem ersetzt

Standard-Dir-Aufteilung:

- **/bin**, **/usr/bin**: Ausführbare Befehle
- **/boot**: Kernel, initrd, boot-Loader
- **/dev**: Pseudo-Files (keine Daten!) für Devices
- **/etc**: Systemweite Konfig-Files
(Benutzerspezifische Konfig-Files stehen unter **/home/user_name** !)
- **/home**: Benutzer-Home-Directories, für jeden **/home/user_name**
- **/lib**, **/usr/lib**: Libraries, Library-ähnliche Daten und Module, Firmware-Code und Module für Kernel, ...
- **/lib64**, **/usr/lib64**: Falls getrennt: **/lib** für 32-Bit-Lib's, **/lib64** für 64-Bit-Lib's
- **/lost+found**, **/home/lost+found**: Ein Mal pro Filesystem, normalerweise leer
Hier lädt der Filesystem-Checker die gefundenen "Leichen" ab
- **/media**: Modernes Linux: Unterverzeichnisse zum Mounten externer Datenträger
- **/mnt**: Traditionelles Unix: Unterverzeichnisse zum Mounten externer Datenträger
- **/proc**, **/sys**: Pseudo-Filesysteme
(keine echten Files & Dir's auf Platte, sondern vom Kernel vorgetäuscht):
Informationen über Kernel, HW, laufende Prozesse, Devices, Netzwerk, ...
- **/opt**: Große Anwendungs-Pakete von Drittherstellern (z.B. Java, Datenbanken, ...)
- **/root**: Home-Verzeichnis von "root"
(muss im root-Filesystem liegen, nicht unter **/home** oder **/usr**)
- **/run**: Temporäre Files von Daemons (Diensten, Hintergrund-Programmen)
- **/sbin**, **/usr/sbin**: Administrative ausführbare Befehle
- **/tmp** oder **/usr/tmp** oder **/var/tmp**:
Temporäre Dateien, meist beim Reboot weg (oft RAM-Disk)
(auch für Benutzer als "Schmier-Verzeichnis")

- **/usr**: Siehe unten
- **/var**: Persistente (über Reboot erhaltene), aber veränderliche & system-lokale Files: Drucker-Warteschlange, systemweite Logfiles und Cachefiles, ev. systemweite Verzeichnisse wie alle installierten Pakete, ...

Trennung in **/bin** und **/sbin** (bzw. **/usr/bin** und **/usr/sbin**):

- **/bin**: Allgemeine Befehle, für alle
 - **/sbin**: Administrator-Befehle
(**/sbin** ist bei **root** im Pfad, bei normalen Usern meist nicht)
- => Tendenz in modernen Linux-Systemen: Zusammenlegen?

Trennung von **/bin**, **/sbin** und **/lib** in **/** und **/usr**:

- Historisch (wegen zu kleiner Root-Platte:
/ und **/usr** waren zwei oder mehr getrennte Platten/Filesysteme!)
- In **/**: Alles, was "boot-notwendig" ist
(bis zu laufendem System mit Shell im Text-Modus), System-Befehle
- In **/usr**: Alles, was nicht "boot-notwendig" ist (GUI, Anwendungen, ...)
- Soll in Zukunft verschwinden:
/bin = **/usr/bin**, **/lib** = **/usr/lib**, **/sbin** = **/usr/sbin**

/bin, **/sbin**, **/lib** und alles unter **/usr** enthalten nur "Installierte" Dateien:

- Keine sich im laufenden Betrieb ändernden Dateien, keine temporären Dateien, ...
 - Keine Benutzer-Dateien, keine maschinenspezifischen Konfigurations-Dateien, ...
- => **/usr** und die darin enthaltenen Dateien ändern sich nur, wenn SW installiert wird
/usr könnte theoretisch im Normalbetrieb readonly sein!
- => **/usr** kann zwischen mehreren "gleichartigen" Rechnern über Netz geshart werden!

Unter **/usr** (außer **/bin**, **/lib**, **/sbin**)

- **/usr/include**: System-C-Headerfiles
- **/usr/libexec**: Nachladbare Module für Exe-Files
- **/usr/local/...**: Selbstinstallierte, lokale Anwendungen und Systemerweiterungen, lokale Dokumentation, ...
=> Alles, was lokal zum System dazugebastelt wurde
(nicht aus Distribution, nicht aus Anwendungs-Installations-Paketen)
- **/usr/share**: Plattform- bzw. Binärformat-unabhängige Inhalte
(Doku und Hilfe, Man-Pages, Fonts, Vorlagen, Icons, Bildschirm-Hintergründe, Ländereinstellungs- & Spellcheck-Daten, Druckerbeschreibungs-Daten, Kryptozertifikate, Daten und fixe Konfig-Files im Textformat, Skriptsprachen- und Source-Files, ...)
- **/usr/src**: Vor allem Linux-Kernel-Sourcen

Befehls-Format:

- Eigentlich Standard: *befehlsname optionen filenames*
(Optionen **vor** den Filenamen!)
 - Optionen beginnen mit **-**
==> Filenamen mit **-** am Anfang sind sehr unklug
 - Laut Standard: 2 Options-Varianten
 - a) Ein **-**: Für Optionen mit einem Buchstaben
Mehrere Options-Buchstaben hinter einem **-** möglich
Bei Optionen mit Wert: Zwischenraum zwischen Option und Wert
 - b) Zwei **--**: Für Optionen, die ganze Worte sind
Hier muss jede Option für sich mit eigenen **--** stehen
Bei Optionen mit Wert: **--option=wert**
- Aber: Viele Programme halten sich nicht an den Standard!
==> Oft Wort-Optionen hinter einfachen **-** (z.B. **gcc**)
- Manche Befehle sehen ganz anders aus („historisch gewachsen“): Beispiele:
find (sucht Dateien und Verzeichnisse): *Verzeichnisse vorn, Optionen hinten*
dd (Kopiert "rohe" Daten, meist von / nach Devices):
Keine alleinstehenden Filenamen, nur Optionen oder option=wert, aber ohne **-**

Befehle für Files & Dirs:

- **cd**, **pwd**, **/bin/pwd** (Unterschied)
- **ls**, **stat**, **du**
- **cat**, **more** bzw. **less**, **head** und **tail**, **vi** oder **nano**
- **cp**, **mv**, **rm** (viele Optionen, u.a. rekursiv)
- **touch**
- **mkdir**, **rmdir**
- **chmod** (oktal oder symbolisch), **chown**, **chgrp**
- **file**, **wc**
- **find**, **xargs**

Zu **ls** : Viele Optionen, siehe **man ls**!

ls -l, **ls -a**, **ls -ltr**, ...

Zum Output von **ls -l**: Siehe auch **man 2 stat**, **man inode**:

- Rechte-Konzept, Owner & Gruppe
(2 Möglichkeiten für Gruppen-Zuteilung: Vom Anleger oder vom Verzeichnis!)
- Für Rechte-Prüfung bei Zugriff: Process Owner & Gruppe, sekundäre Gruppen
- Beim Anlegen: Umask wird von Rechte-Bits abgezogen
- Sticky-Bit, Suid und Setgid (2 Bedeutungen)
- Mögliche File-Typen: File, Dir, Block-Device, Char-Device, Sym-Link, Socket, Fifo
(Skript S. 40)

- Konzept der Inodes (**ls -li**: Inode-Nummer) (Skript S. 38)
- Konzept der Hard-Links (+ Link Count im **ls**), Konzept der Symbolic-Links
Befehl **ln** und **ln -s** (Skript S. 38/39)
- Datum erklären (Birth- bzw. Create-Date fehlt, Access-Date oft abgeschaltet)
- Bei Devices: Major und Minor Number
- Sparse Files: Größe != Platzbedarf (**-s**)

Pipe und Redirect:

- Redirect: **>**, **<**, **>>**, **2>**, **2>&1**
- Pipe: *befehl1 | befehl2*
Am häufigsten: ... | **less**
Speziell für Pipes: Befehl **tee**

Andere Befehle:

- **ps**, **pstree**, **top**, **kill**
- **ulimit**, **umask**
- **ping**, **traceroute**, **ifconfig**, **netstat**
- **echo**, **printf**

Linux kennt viele Befehle, die zeilenweise Text verarbeiten
Entweder alleinstehend verwendet oder mit Pipe verbunden

- **grep**, **fgrep**
- **diff**, **cmp**
- **sort**, **uniq**, **tac**
- **comm**, **join**, **paste**
- **cut**, **tr**
- **sed**, **awk**

Archive & Komprimierung unter Linux:
tar & diverse Kompressions-Programme

Fortgeschrittene Themen:

- Das Background-&, Ctrl/Z, **bg**, **fg**, **jobs**
- **||** und **&&**
- Variable Substitution
- Command Substitution: **`...`** oder **\$(...)**