Geschichtliche Hintergründe – die wichtigsten Stationen

1965: MULTICS (Bell Telephone Laboratories)

- Ziel: Multiuser-Betriebssystem,
- Fehlschlag, Aufgabe nach ein paar Jahren

ab 1969: UNIX-Vorversion von ehemaligen Mitarbeitern des MULTICS-Projektes (THOMPSON, KERNIGHAN & RITCHIE); Entwicklung der Programmiersprache B

1973: UNIX

- Weiterentwicklung von B nach C,
- UNIX ist fast ausschließlich in C geschrieben $\Longrightarrow$ Portabilität

1983: UNIX System V (AT&T);
parallele Entwicklung von BSD-UNIX (Berkeley System Distribution) in Berkeley (University of California)

Mitte der 90er Jahre:

- Mittlerweile gibt es viele herstellerspezifische Varianten, aber die Hauptvarianten System V und BSD bleiben.
  Beispiel RZ:
  - SunOS (cipsuns): BSD,
  - Solaris (RZ-suns): System V,
  - AIX (IBM-Maschinen des RZs)
- weitere Varianten: HP/UX, SGI, DEC ...
- Bestrebungen der Hersteller, die verschiedenen UNIX-Strömungen wieder zu vereinen
- Entwicklung freier UNIXe für den PC (Linux, FreeBSD u. a.)

Unix-Konzepte

UNIX ist sehr flexibel da

- Programme durch Pipes verbunden werden können,
- einfache, spezialisierte Werkzeuge zu mächtigen Werkzeugen zusammensetzbar sind:

```
$ tar cf - Verzeichnis1 | ssh sun2 '( cd temp; tar xvfp -)'
```

- alles wie eine Datei behandelt wird, die Hardware – Festplatten, Tastaturen, Drucker usw. – eingeschlossen.

Vergleich mit anderen Betriebssystemen

Betriebs-system	Benutzer	gleichzeitige Prozesse	Scheduler	
DOS	eine/r	nein	nein	Singleuser-Singletasking-System
DOS/MS-Windows	eine/r	jein	Applikationen	Singleusersystem ohne präemptives Multitasking
OS/2	eine/r	ja	Betriebssystem	Singleusersystem, präemptives Multitasking
UN*X	viele	ja	Betriebssystem	Multiusersystem, präemptives Multitasking

Die ersten Schritte mit Unix

1. Voraussetzung: UserID und Paßwort

2. Anmelden beim System:

   (a) System meldet sich mit *Login-Prompt*:

   ```
   $ cipsun1 login:
   ```

   (b) Paßwort eingeben (wird nicht angezeigt)

3. System eröffnet eine *Shell*

   Benutzer $\Longleftrightarrow$ Shell $\Longleftrightarrow$ Betriebssystemkern

4. Die Shell meldet sich mit dem *Prompt* (meistens $ oder %) bereit:

   ```
   kursse00cipsun1.ruf:/u/kurs/se/kursse00$
   ```

5. Fenstersystem (hier OpenWindows) starten:

   ```
   $ W ov
   ```

- Verschieben der Fenster: linke Maustaste im Rahmen klicken
- Fontvergrößerung: CTRL+rechte Maustaste ⟶ Menü

6. Paßwort ändern! (Datensicherheit)

```
$ ssh cipsun0
$ passwd
$ exit
```

Achtung:

Bei UNIX muß man auf Groß- und Kleinschreibung achten!

7. Hilfe zu den ca. 400 UNIX-Kommandos holen:

```
$ man passwd
$ man man
$ man -k Suchbegriff
```

8. Fenstersystem verlassen und abmelden:

```
$ exit
```

Achtung:

- Man ist erst dann abgemeldet, wenn sich das System wieder mit dem Login-Prompt meldet!
- Niemals einen Unixrechner ausschalten (Multiuser-Multitasking-System)!

Dateien

Von UNIX werden Hard- und Software wie Dateien behandelt.

Beispiele:

`/dev/mouse`	die Maus
`/usr/bin/passwd`	das Programm `passwd`
`/usr/local/man/cat1/ksh.1`	eine Textdatei[1]
`/tmp`	ein Verzeichnis

[1] Manpage zu `ksh`

Editieren von Textdateien – der vi

Textdateien werden mit *Editoren* erstellt bzw. verändert. Der Editor `vi` ist auf allen UNIX-Systemen vorhanden.

`$ vi` *filename*

Der `vi` befindet sich nun im *Kommandomodus*.

Kommandomodus: Positionieren des Cursors, Texte suchen, Texte speichern usw.; erreichbar durch Betätigen der `ESC`-Taste

Eingabemodus: Texteingabe; erreichbar durch `i`, `a` und andere Tastenkombinationen.

Lernprogramm:

```
$ vilearn
```

Befehle im Kommandomodus:

`h, j, k, l`	– Positionieren des Cursors (meist auch durch die Cursortasten möglich)
`CTRL-f`	– ganzer Bildschirm nach unten
`CTRL-b`	– ganzer Bildschirm nach oben
`G`	– bewegt den Cursor an den Anfang der letzten Zeile der Datei
`x`	– löscht das Zeichen unter dem Cursor
`dd`	– löscht die Zeile, in der der Cursor steht
`o`	– erzeugt eine neue Zeile hinter der aktuellen und wechselt in den Eingabemodus
`O`	– erzeugt eine neue Zeile vor der aktuellen und wechselt in den Eingabemodus
`:w`	– Speichern der Datei
`:q`	– Verlassen der Datei
`:wq`	– Speichern und Verlassen der Datei

Anzeigen und Bewegen von Dateien

$ `cat` *filename*

$ `more` *filename* (seitenweise Anzeige der Datei)

Tasten-Kommandos für more:

Leertaste	– blättert eine Seite vor
Entertaste	– blättert eine Zeile vor
`b`	– blättert eine Seite zurück
`h`	– Hilfe (zur Verfügung stehende Kommandos in `more`)
`q`	– beendet more

$ `ls`	(list) Auflisten des Inhalts eines Verzeichnisses
$ `cp` *file1 file2*	(copy) Kopieren von Datei *file1* nach *file2*
$ `rm` *file1*	(remove) Löschen der Datei *file1*
$ `mv` *file1 file2*	(move) Umbenennen der Datei *file1* in *file2*

Dateiverzeichnisse

`$ mkdir` *dirname*	(make directory)
`$ ls`	
`$ ls -F`	Die Option `-F` hängt einen / an den Namen von Verzeichnissen an.
`$ mv` *file1 dir1*	bewegt *file1* in das Verzeichnis *dir1*
`$ ls` *dir1*	Auflisten des Inhalts des Verzeichnisses *dir1*
`$ cp` *file2 dir1*	*file2* wird nach *dir1* kopiert, bleibt aber auch im ursprünglichen Verzeichnis erhalten.
`$ rmdir` *dir1*	Löschen von *leeren* Verzeichnissen
`$ rm -r` *dir1*	rekursives Löschen: Löschen von *dir1* einschließlich aller darin enthaltenen Dateien und Unterverzeichnisse

EINMAL GELÖSCHTE DATEIEN UND VERZEICHNISSE KÖNNEN UNTER UNIX NICHT WIEDER HERGESTELLT WERDEN!!!

Die Dateihierarchie

Das Verzeichnis, in dem man sich gerade befindet, wird *aktuelles Arbeitsverzeichnis* („working directory") genannt. Das Verzeichnis, in dem man sich nach dem Anmelden am System befindet, heißt *Homedirectory* des Benutzers/der Benutzerin.

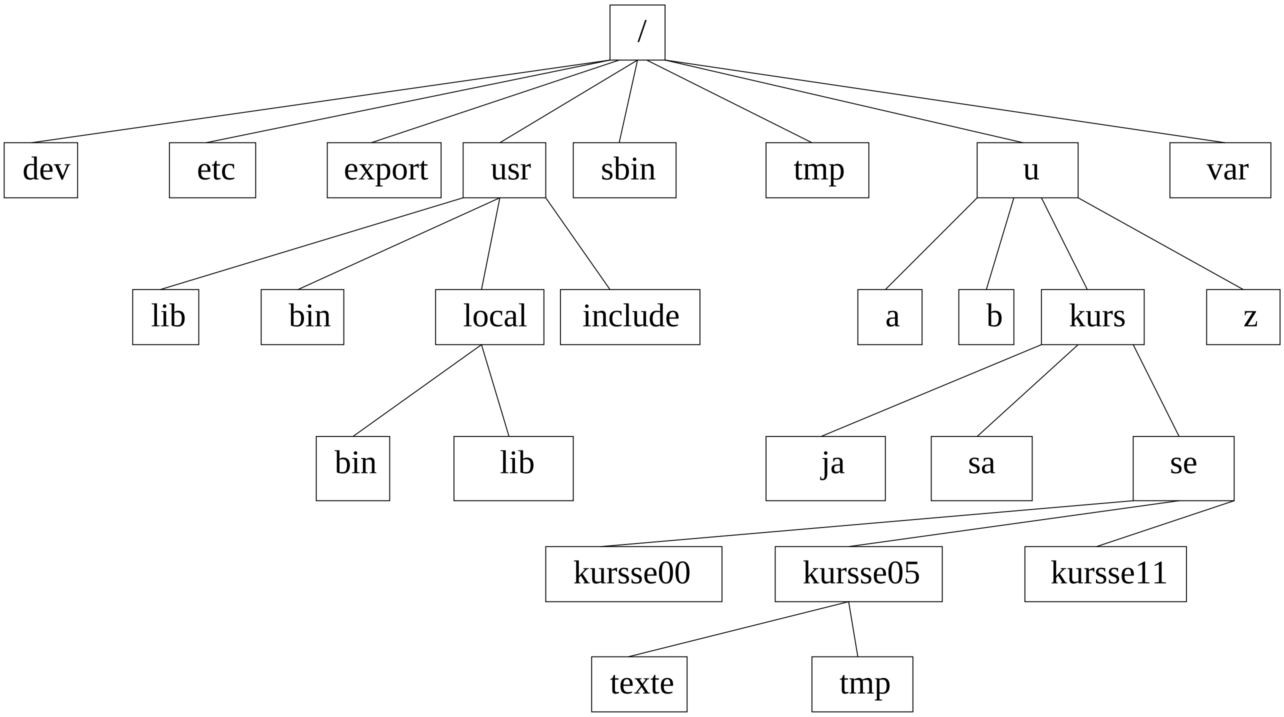
Wechseln des Verzeichnisses:

`$ cd` *dir1*	change working directory (macht *dir1* zum aktuellen Verzeichnis)
`$ cd ..`	geht eine Hierarchiestufe nach oben
`$ cd`	wechselt in das eigene Homedirectory

Name des aktuellen Verzeichnisses:

`$ pwd`	print working directory

Der Dateibaum



absoluter Pfad: Angabe des Pfades vom Wurzelverzeichnis aus

```
$ cd /u/kurs/se/kursse05/texte
```

relativer Pfad: Angabe des Pfades relativ zum aktuellen Verzeichnis

```
$ cd ../texte
```

Zugriffsrechte

Beispiel zum Ausprobieren:

```
$ cd /u/p
$ ls -F
$ cd pjung
$ pwd
```

Warum darf man nicht in das Verzeichnis `pjung` wechseln?

In UNIX wird jeder Datei und jedem Verzeichnis ein Satz von Zugriffsrechten (permissions) zugeteilt:

`r` wie readable (file),
`w` wie writeable (file) und
`x` executable (file) für

- den Besitzer einer Datei/eines Verzeichnisses (user, abgekürzt: `u`),
- die Mitglieder der Gruppe, der der Besitzers zugeordnet ist (group, abgekürzt: `g`) sowie
- alle anderen Benutzer (others, abgekürzt: `o`).

Beispiel: $\underbrace{\texttt{r w x}}_{user}\ \underbrace{\texttt{r - x}}_{group}\ \underbrace{\texttt{r - -}}_{others}$

Information über Zugriffsrechte einer Datei mit `ls -l`:

```
total 9
drwxr-xr-x  2 tantzen   512 Apr 28  1992 tantzen
drwx------  4 test1     512 Nov 23  1993 test1
drwxr-xr-x  4 test2     512 Nov 23  1993 test2
drwxr-xr-x  4 test3     512 Nov 19  1993 test3
lrwxrwxrwx  1 root       14 Oct 16 14:42 testuser -> /u2/t/testuser
lrwxrwxrwx  1 root       14 Oct 16 15:05 theckert -> /u2/t/theckert
drwxr-xr-x  4 thies     512 Nov 22  1993 thies
drwxr-xr-x 16 tseng    1536 Oct 24 21:26 tseng
```

Die Angaben bedeuten im einzelnen:

Spalte 1	: Dateityp (z. B. `d` für directory)
Spalten 2–10	: Zugriffsrechte für die Datei
Spalte 11	: Anzahl der Verweise (hier nicht wichtig)
Spalte 12	: Benutzer/Besitzer
Spalte 13	: Größe der Datei/des Verzeichnisses
Spalte 14–16	: Datum bzw. Zeitpunkt der letzten Inhaltsänderung der Datei
Spalte 17	: Name der Datei

Rechte verändern mit `chmod`, spezifiziert

- durch Angabe
  - der betroffenen Benutzergruppe,
  - der Änderungsoperation sowie
  - des Zugriffsrechts.

```
$ chmod g-r   filename
$ chmod o+x   filename
$ chmod o+x,g-r filename
$ chmod go-r filename
$ chmod go=rw filename
```

- durch Angabe einer Oktalzahl.

1 entspricht executable
2 entspricht writeable
4 entspricht readable

$$\underbrace{\texttt{r}\quad\texttt{w}\quad\texttt{x}}_{\substack{4\quad 2\quad 1\\ 7}}\quad\underbrace{\texttt{r}\quad\texttt{-}\quad\texttt{x}}_{\substack{4\quad 0\quad 1\\ 5}}\quad\underbrace{\texttt{r}\quad\texttt{-}\quad\texttt{-}}_{\substack{4\quad 0\quad 0\\ 4}}$$

```
$ chmod 754   filename
```

Kommandos – Kommunikation mit dem Betriebssystem

- Kommunikation zwischen Benutzer und UNIX via Kommandointerpreter (shell, e. g. `ksh`)

  Ein Kommandointerpreter

  - liest Kommandoeingaben,
  - analysiert sie,
  - verarbeitet sie selbst bzw.
  - leitet sie zur Ausführung weiter.

- Betriebssystembefehle (in der Regel C-Programme) greifen über *system calls* auf den UNIX-Kern (kernel) zu

Kommandoaufrufe

- Login-Prozeß startet Shell ⟶ Prompt erscheint
- Eingabe eines Befehls durch den Benutzer
- Shell sucht entsprechendes Programm
  - ... und übergibt ihm die Kontrolle ⟶ Shell wartet auf das Ende des Programms (Vordergrundprozeß)

    `$ tea`
  - Programm wird im Hintergrund gestartet ⟶ Shell sofort wieder bereit für Eingabe (Multitasking!); Benutzer erhält Nachricht nach Ende des Programms

    `$ tea &`

Anzeigen aller Hintergrundprozesse:

`$ jobs`

Argumente

$ `mv` *file1 file2*	ändert den Dateinamen von *file1* nach *file2*
$ `mv` *file1 dir1*	verschiebt die Datei *file1* in das Verzeichnis *dir1*
$ `mv` *file1 dir1/file2*	verschiebt die Datei *file1* in das Verzeichnis *dir1* und ändert ihren Namen in *file2*
$ `mv` *dir1 dir2*	benennt das Verzeichnis *dir1* in *dir2* um (falls es *dir2* noch nicht gibt) oder verschiebt das Verzeichnis *dir1* (samt Unterverzeichnissen) in das bereits vorhandene Verzeichnis *dir2*
$ `mv` *dir1 dir2/dir3*	verschiebt einen ganzen Verzeichnisbaum in ein anderes (vorhandenes) Verzeichnis und ändert den Namen

Optionen

- unabhängige Optionen

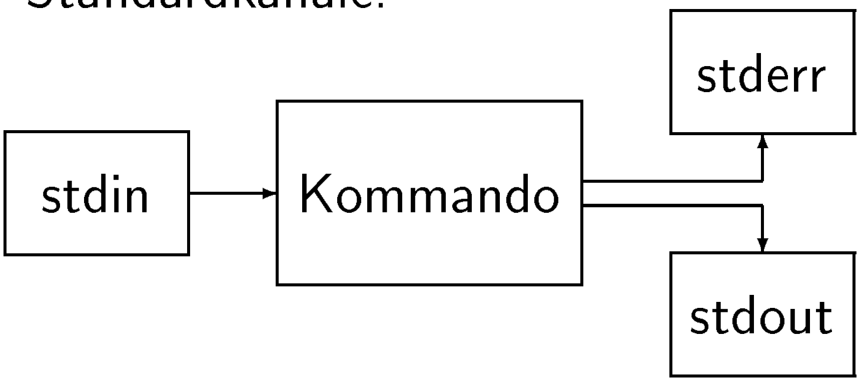
`$ ls -l`	zeigt ein langes Listing (die Zugriffsrechte werden mitangezeigt)
`$ ls -F`	hängt einen / an Verzeichnisse an
`$ ls -lF`	zeigt ein langes Listing mit zusätzlichem / am Ende von Verzeichnisnamen

- abhängige Optionen ⟶ Reihenfolge u. U. nicht beliebig

`$ ls -l`	zeigt den Zeitpunkt der letzten *Inhaltsänderung* an
`$ ls -lc`	zeigt den Zeitpunkt der letzten *allgemeinen Änderung* an (e. g. Änderung der Rechte)
`$ ls -lu`	zeigt den Zeitpunkt des letzten (e. g. lesenden) *Zugriffs* an

Datenumlenkung

Standardkanäle:



Defaults:

- *Standardeingabe* ⟶ Tastatur,
- *Standardausgabe* ⟶ Bildschirm,
- *Standardfehlerausgabe* ⟶ Bildschirm.

Umlenkung:

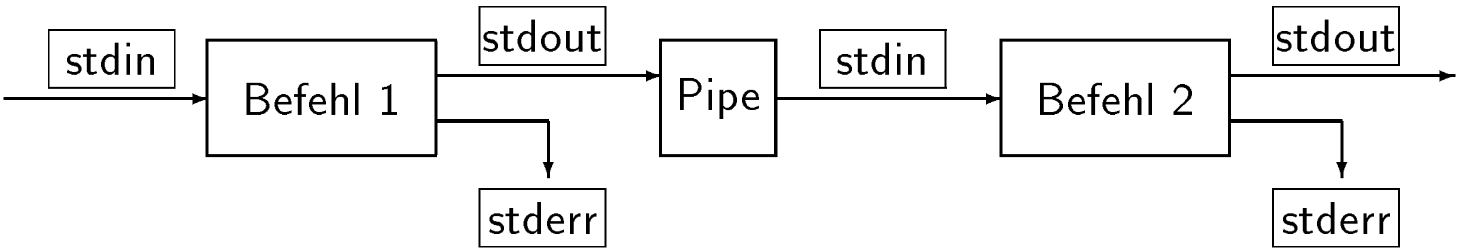
$ *Unixkommando* > *Ausgabedatei*
$ *Unixkommando* >> *Ausgabedatei*
$ *Unixkommando* < *Eingabedatei*
$ *Unixkommando* 2> *Fehlerdatei*

`$ ls -lF >` *xxx* — Die erzeugte Liste wird in *xxx* abgelegt.

`$ mkdir` *test*

`$ rm` *test* `2>` *fehler* — In der Datei *fehler* steht jetzt die Fehlermeldung, daß *test* ein Verzeichnis ist und folglich nicht mit `rm` gelöscht werden kann.

Pipes:




```
$ ls -lF /dev | more
$ cat /etc/passwd | more
$ cat /etc/passwd | grep ddegen
```

Prozesse

Prozeß: Folge von Aktivitäten, die zwischen dem Aufruf eines Programms und dessen Ende stattfinden. Von jedem Kommando wird ein neuer Prozeß gestartet.

Prozeßabbruch:

- Vordergrundprozeß: i. d. R. mit `CTRL-C`
- nichtreagierende Prozesse: `$ kill` *pid*
- wenn gar nichts mehr hilft: `$ kill -9` *pid*

Liste der Prozesse inklusive der *Prozeßidentifikationsnummer PID*:

`$ ps`	process status
`$ ps -axu`	ausführliche Liste unter BSD (z. B. cipsuns)
`$ ps -ef`	ausführliche Liste unter System V (z. B. IBM-Cluster, RZ-Suns)

Beispiel:

`$ coffee`	Endlosschleife im Vordergrund $\longrightarrow$ mit CTRL-C abzubrechen
`$ coffee &`	Endlosschleife im Hintergrund $\longrightarrow$ CTRL-C funktioniert nicht
`$ ps -aux | more`	In anderer Shell: Welche PID hat `coffee` ?
`$ kill` *pid*	

Peripherie

Drucker

Für UNIX stellt das Drucken einfach nur das Kopieren einer Datei in eine andere Datei, nämlich den Drucker, dar.

`$ lpr -P`*druckername druckfile*	(BSD)
`$ lp -d`*druckername druckfile*	(System V)

Welche Drucker gibt es im System?

```
$ lpstat -p
```

Beispiel Cipsun-Pool:
zwei virtuelle Drucker namens `ascii` und `ps`

`$ lpr -Pascii` *ASCII-Datei*
`$ lpr -Pps` *Postscriptdatei*

Druckaufträge werden in eine Warteschlange (queue) gestellt und vom *line printer daemon* `lpd` verwaltet.

Ansehen der Warteschlange:

`$ lpq -P`*druckername*

Entfernen eines Druckauftrags aus der Warteschlange:

`$ lprm -P`*drucker auftragsnummer*	(BSD)
`$ cancel` *auftragsnummer drucker*	(System V)

Diskettenlaufwerke

- Auswerfen einer Diskette mit `eject`
- keine einheitliche Handhabung
- Zugriff auf DOS-Disketten häufig mit den Mtools: `mdir`, `mcd`, `mmd`, `mdel`, `mcopy` etc.

`$ mcopy` *unixfile* `a:`*DOS-Pfad/dosfile*

Eigenschaften der Kornshell

Startdateien

`/etc/profile` : nur vom Systemadministrator zu modifizieren

`.profile` im Homeverzeichnis des Benutzers: wird bei jedem Login abgearbeitet

`.kshrc` im Homeverzeichnis des Benutzers: wird mit jeder neuen Shell aufgerufen

Anzeigen versteckter Dateien

(e. g. `.profile`, `.kshrc`):

`$ ls -al` listet *a*lle Dateien im *l*ong format auf

Umgebungsvariablen

- Beispiele:

  PATH . . . Suchpfad für ausführbare Programme
  HOME . . . absoluter Pfad des Homeverzeichnisses

- Konvention: Großschreibung

- Anzeigen *aller* definierten Umgebungsvariablen:

  `$ env`

- Anzeigen *einzelner* Umgebungsvariablen:

  `$ echo $`*umgebungsvariable*

  Beispiel: `$ echo $PATH`

- Neudefinition:

  `$ export PATH=$HOME/bin:$PATH`

Aliase

Definition eigener Befehle mit `alias`:

```
$ alias dir='ls -al | more'
```

Benutzerdefinierte Aliase und Umgebungsvariablen gelten nur für die Lebensdauer der aktuellen Shell. Sollen sie auch für zukünftige Sessions gelten, setzt man sie in `.profile` oder `.kshrc`.

Dateinamenexpansion mit Wildcards

`*`	steht für eine beliebige Zeichenkette
`?`	steht für ein einzelnes Zeichen
`[`*Zeichenmenge*`]`	steht für einen bestimmten Zeichensatz

Beispiele:

`$ cat p*`	zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` beginnen
`$ cat p?`	zeigt den Inhalt aller zweibuchstabigen Dateien an, die mit einem kleinen `p` beginnen
`$ cat [pP]*`	zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` oder einem großen `P` beginnen
`$ cat p*[1-3]`	zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` beginnen und mit einer `1`,`2` oder `3` enden

Komplettierung von Dateinamen

Vervollständigen angefangener Dateinamen durch zweimaliges Drücken der `ESC`–Taste.

Kommandowiederholung

Mit Hilfe der *Pfeiltasten* ↓ und ↑ kann man früher abgeschickte Kommandos „wiederverwenden".

Anmelden an entfernten Unix-Rechnern

Voraussetzung: UserID auf Remotemaschine

`$ telnet` *rechnername*
`$ ssh` *rechnername*

rechnername: symbolischer Internetname oder entsprechende IP-Nummer des Remoterechners.

Beispiel:

```
$ telnet cipsun0
$ ssh cipsun0.ruf.uni-freiburg.de
$ telnet 132.230.1.80
```

`ssh`: Angabe der UserID auf dem Remoterechner mit der Option `-l`

Beispiel:

`$ ssh -l` *UserID* `sun2`

Dateitransfer auf entfernte Unix-Rechner

Verschlüsselte (sichere) Datenübertragung mit `scp` („Secure Copy“):

$ `scp` *filename remote_UserID@Remote-Rechner* : *Pfad/und/Dateiname*

Wenn man auf beiden Rechnern dieselbe UserID hat, kann man sie auch weglassen:

$ `scp` *filename Remote-Rechner* : *Pfad/und/Dateiname*