

Dauer: 3 Tage

Dozent: Mark Heisterkamp

Email: heisterkamp@rrzn.uni-hannover.de

Vortrag: 9.15 Uhr - 13.00 Uhr

Übung: 13.30 Uhr - 17.00 Uhr

Dieser Kursus richtet sich an Unix- und damit auch an Linux-Anfänger. Ziel ist es, neben grundlegenden Unix-Kenntnissen die für die Benutzung von Anwendersoftware auf dem Workstationpool des RRZN notwendigen Kenntnisse zu vermitteln. Folgende Themen werden behandelt: Aufbau des Dateiensystems, Bearbeiten und Editieren von Dateien, Ein-Ausgabe-Umlenkung, Unix-Shells, verschiedene Hilfemöglichkeiten. Ferner werden die Unix-Programme zur Kommunikation mit anderen Rechnern (SSH, FTP, News, E-Mail) sowie die Benutzung der grafischen Oberfläche der Sun-Workstations vorgestellt. Voraussetzung ist die Kenntnis des Stoffumfangs des Unix-Vorkursus.

Literatur:

- RRZN-Broschüre „UNIX - Eine Einführung in die Benutzung“ (7.00DM)
- Skript zum Kurs

Das Skript zum Kurs liegt als PDF-Datei auf einem Server des RRZN unter

`http://www.rrzn.uni-hannover.de/Kurse/Materialien/index.html`

Themen 1. Teil:

- Rechner und Betriebssystem
- Erste Schritte ...
- Dateiverwaltung
- Prozessverwaltung
- Die Shell
- Eingabeumlenkung
- Nützliche Befehle

Rechner und Betriebssystem

- Das *BIOS* (**B**asic **I**nput **O**utput **S**ystem) hält einige grundlegende Informationen über die Rechnerkomponenten vor.
- Der Rechner „weiß“ bis auf die BIOS-Informationen nichts über sich.
- Das Betriebssystem kann die Komponenten und den Rechner interaktiv (d. h. unter Einflussnahme des Benutzers) verwalten.
- In der Form der Verwaltung dieser Rechnerkomponenten unterscheiden sich Betriebssysteme voneinander.

Entwicklung

- **60er-70er:** Großrechner im *Batch-Betrieb*, d. h. alle Programme wurden nacheinander ausgeführt (Stapelverarbeitung), ohne den Benutzer in den Programmablauf zu integrieren.
- **70er:** Workstations wurden entwickelt, und der Bedarf nach dialogorientierten Betriebssystemen nahm zu. Die Entwicklung von UNIX nahm damit ihren Anfang.
- **80er:** Der PC erschien mit dem Betriebssystem DOS, das der niedrigen Leistungsfähigkeit des PCs Rechnung trug. Für ein UNIX oder ein ähnlich mächtiges Betriebssystem war der PC zu klein.
- **Ende 80er:** Der PC wurde leistungsfähig genug, um eine grafische Benutzeroberfläche (Windows) verwalten zu können. UNIX- und Apple-Rechner verfügten bereits über eine solche Oberfläche.
- **90er:** Die PCs wurden so leistungsfähig, dass sie ein vollwertiges UNIX verkraften konnten. Die Entwicklung von Linux begann.
- **Heute:** PCs auf Linux-Basis konkurrieren mit „echten“ UNIXen und werden auch in empfindlichen EDV-Bereichen eingesetzt (Netzwerke, Server ...).

Geschichte von UNIX

- 1969 begann Ken Thompson an den Bell Laboratories die Entwicklung von UNIX. Er wollte die Bedienung von Rechnern benutzerfreundlicher gestalten, gleichzeitig sollte das Betriebssystem relativ kompakt sein, so dass es auch auf einer Workstation laufen konnte. Außerdem sollte das Betriebssystem unabhängig von der Rechnerarchitektur sein.
- Die erste Variante war noch in Maschinsprache geschrieben. Wegen der Portierung auf andere Rechner erfand er die Programmiersprache B.
- Dennis Ritchie entwickelte B zur Programmiersprache C weiter.
- 1971 wurde UNIX zum ersten mal komplett (bis auf den Kernel) in C geschrieben. Die Quellcodes dieser Variante waren frei erhältlich, und jeder konnte somit das Betriebssystem nach Belieben weiterentwickeln.
- Es entwickelten sich viele Dialekte, die teilweise inkompatibel waren.
- Die Entwicklung war schnell und der Bedarf nach UNIX hoch. Eine Standardisierung wurde notwendig.
- 1991 wurde der Quasi-Standard *Unix System V, Release 4* (SVR4) entworfen, an dem sich alle kommerziellen UNIXe anpassen sollten.
- Zwar unterschieden sich die UNIXe noch immer, aber nur geringfügig, und der Umstieg von einer Variante zur anderen wurde problemlos.

Abgrenzung zu anderen Betriebssystemen

Gemeinsamkeiten:

- Grafische Benutzeroberfläche mit Maussteuerung.
- Multitasking

Unterschiede:

- UNIX ist multiuserfähig.
- UNIX besitzt eine sehr zuverlässige Prozesskontrolle (wichtig für das Multitasking). Dementsprechend absturzsicher ist das System!
- Wegen seiner wissenschaftlichen Ausrichtung besitzt UNIX einen großen Umfang an Entwicklungs- und Programmierwerkzeugen.
- UNIX besteht aus vielen kleinen nützlichen Programmen, die nur eine spezielle Aufgabe sehr effektiv erledigen. Große „Alleskönner“ sind anderen Systemen vorbehalten.
- Mittels sogenannter Skriptsprachen kann man die kleinen Programme des vorherigen Punktes zu leistungsfähigen „großen“ Programmen zusammenbauen.
- Trotz der Beliebtheit der Maus werden noch immer viele Prozesse unter UNIX mittels Tastatur gesteuert.

Vor- und Nachteile von UNIX

Nachteile:

- Gewöhnungsbedürftige Bedienung.
- Komplexe Installation.
- Unter Umständen komplexe Nachinstallation von zusätzlichen Programmen.
- SVR4 ist kein 100%er Standard.
- Gewisser Mangel an multimedialen Fähigkeiten.

Vorteile:

- Sehr hohe Stabilität und Verfügbarkeit.
- Durch Multitasking und Multiuserbetrieb ist eine komfortable Nutzung und Administration des Systems möglich.
- Eine Standardinstallation beinhaltet bereits für die meisten Aufgaben eine Lösung.
- Die Entwicklungs- und Programmierwerkzeuge zählen zu den mächtigsten und leistungsfähigsten im EDV-Betrieb.
- In seiner Erscheinungsform als Linux ist UNIX kostenlos.

Erste Schritte ...

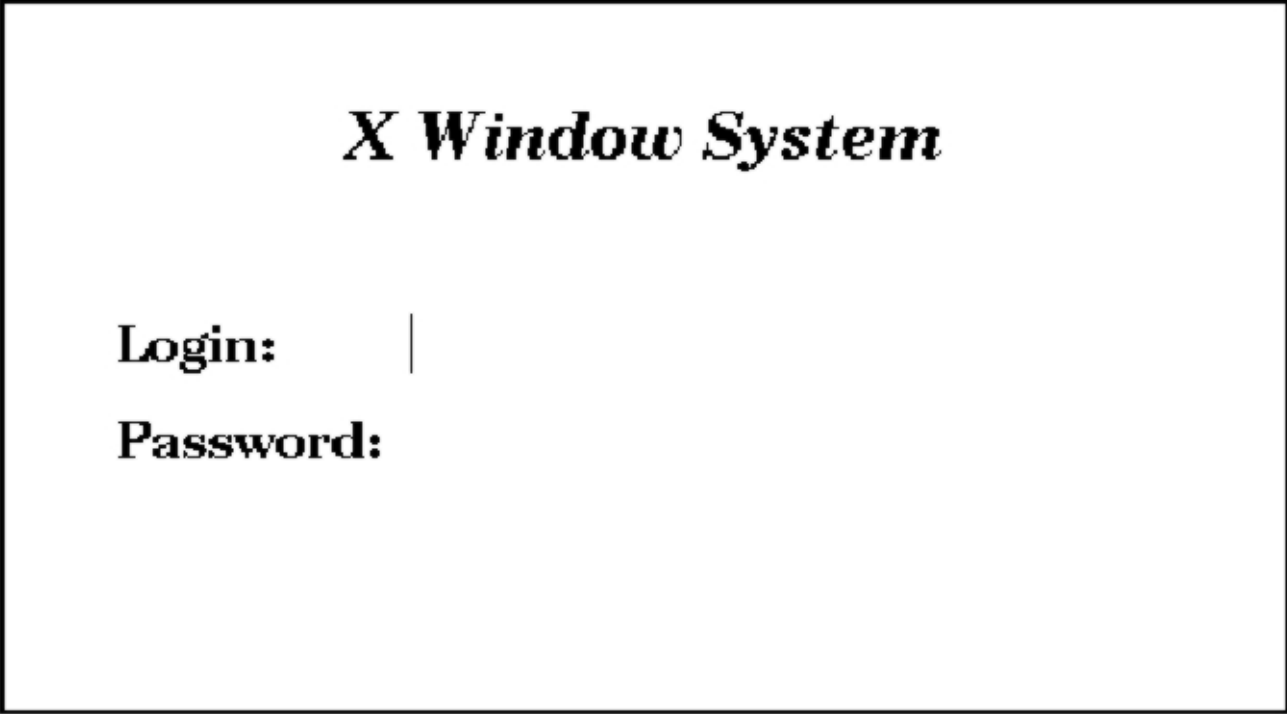
Anmeldung am System:

- Benutzername (bzw. User-ID, Login-Name)
- Passwort

Zwei Arten der Anmeldung:

- grafisch
- textorientiert (Konsole)

Grafische Anmeldung:



X Window System

Login: |

Password:

A rectangular window with a black border. Inside, the text 'X Window System' is centered and italicized. Below it, on the left, are the labels 'Login:' and 'Password:'. To the right of 'Login:' is a vertical line, and to the right of 'Password:' is a larger rectangular input field.

Textorientierte Anmeldung (Konsole):

```
sun225h console login: mheiste  
Password: |
```

Eingabe des Benutzernamens und des Passwortes:

- Auf Groß- und Kleinschreibung achten!
- Bei Eingabe des Passwortes wird unter Umständen keinerlei Bildschirmausgabe erzeugt.

Erste erfolgreiche Anmeldung

- Man befindet sich entweder auf der grafischen Benutzeroberfläche mit Maussteuerung oder auf der Konsole.
- Um im System arbeiten zu können braucht man einen Prompt, d. h. die Möglichkeit, Befehle per Tastatur an den Rechner geben zu können.
- Auf der Konsole hat man bereits einen Prompt.
- Auf der grafischen Oberfläche muss noch ein sogenanntes XTerminalfenster (X-Terminal, XTerm) geöffnet werden. In der Regel ist bei der Anmeldung bereits eines geöffnet. Ist kein Fenster zur Befehlseingabe vorhanden, so klickt man mit der linken Maustaste einmal auf den Bildschirmhintergrund und klickt einmal auf die Option „Terminal“ und wählt dann ein mögliches Terminalfenster aus.

Befehlseingabe

Befehl Optionen Parameter

Bestimmte Optionen bzw. Parameter können in Abhängigkeit vom Befehl optional oder obligatorisch sein. In der Darstellung wird das durch folgende Schreibweise erreicht:

Befehl [optionale Angaben] <obligatorische Angaben>

Dabei werden die Klammern nicht mit eingegeben! Beispielsweise bedeutet die Angabe von

`mkdir [Option] <Verzeichnis>`

dass dem Befehl `mkdir` einige optionalen Angaben folgen **können**, aber die Angabe eines Verzeichnisses folgen **muss**.

Ändern des Passwortes

Mit dem Befehl

`passwd`

kann man sein Passwort ändern. Nach einmaliger Eingabe des (noch) aktuellen Passwortes wird man zweimal nach der Eingabe des neuen Passwortes gefragt.

Manpages

Eine wichtige Funktion sind die sogenannten Manpages. Durch Eingabe von

```
man <Befehl>
```

kann man sich zu jedem Befehl eine englischsprachige Hilfefunktion aufrufen (unter Linux teilweise auch auf deutsch). Der Hilfetext kann durch die 'b', Leer- und die Returntaste rauf- und runtergescrollt werden. Beendet wird die Hilfefunktion durch Druck auf die Taste 'Q'.

Ein paar nützliche Befehle

<code>date</code>	Ausgabe des aktuellen Datums und der Uhrzeit.
<code>who</code>	Ausgabe aller am System eingeloggten Benutzer.
<code>whoami</code>	Ausgabe des eigenen Benutzernamens.
<code>echo</code>	Der nach <code>echo</code> folgende Text wird am Bildschirm ausgegeben.
<code>more</code>	Der Inhalt der nach <code>more</code> genannten Datei wird auf den Bildschirm ausgegeben.

Dateiverwaltung

- hierarchisch
- Baumstruktur, ausgehend von einer Wurzel '/'.
- Jedes Verzeichnis enthält mind. 2 Einträge: '.' und '..'.
- '.' steht für das Verzeichnis selbst.
- '..' steht für das übergeordnete Verzeichnis.
- Absolute und relative Pfade sind dadurch möglich.

Homeverzeichnis

Jeder Benutzer unter UNIX hat einen eigenen Raum auf der Festplatte, der ihm zugeteilt wurde, das sogenannte Homeverzeichnis.

Verzeichnisinhalt

```
ls [Optionen] [Zielverzeichnis]
```

Optionen:

- l Listenartige Ausgabe mit zusatzinformationen.
- a Es werden alle Dateien (auch versteckte) angezeigt.

Verzeichniswechsel

```
cd [Zielverzeichnis]
```

Ausgabe des aktuellen Pfades

`pwd`

Löschen

```
rm [Optionen] <Datei/Verzeichnis>
```

Anlegen eines Verzeichnisses

```
mkdir [Optionen] <Verzeichnis>
```

Optionen:

- m Zugriffsrechte werden mitgesetzt.
- p Es werden auch oberhalb liegende, möglicherweise noch nicht existente Verzeichnisse angelegt.

Metazeichen bzw. Wildcards

Metazeichen sind Platzhalter. Sie können überall dort eingesetzt werden, wo Datei- oder Verzeichnisnamen angegeben werden.

?	Steht für ein einzelnes beliebiges Zeichen.
*	Steht für eine beliebig lange Folge beliebiger Zeichen.
[...]	Ist eine Auswahlliste von einzelnen Zeichen, die an dieser Stelle stehen dürfen.
[!...]	Ist eine Auswahlliste von einzelnen Zeichen, die nicht an dieser Stelle auftauchen dürfen.

Beispiel für Metazeichen

Folgende Dateien liegen z. Bsp. in einem Verzeichnis:

myfile.txt	brief.txt	bericht.txt
protokoll-a.doc	protokoll-b.doc	

Muster	Entsprechung
protokoll-?.doc	protokoll-a.doc protokoll-b.doc
*.txt	myfile.txt brief.txt bericht.txt
b*	brief.txt bericht.txt
*	myfile.txt brief.txt bericht.txt protokoll-a.doc protokoll-b.doc
[abc]* oder [a-c]*	brief.txt bericht.txt
[!p]*	myfile.txt brief.txt bericht.txt

Prozessverwaltung

- Prozesse sind praktisch gestartete Programme.
- Jeder Prozess hat einen Eigentümer.
- Jeder Prozess hat einen Mutterprozess und ist Kindprozess eines solchen Mutterprozesses.
- Prozesse können ruhen, aktiv sein, angehalten sein, auf die Festplatte ausgelagert oder speicherresistent sein, im Vordergrund oder im Hintergrund laufen, und sie können beendet sein, ohne den Mutterprozess über die Beendigung verständigt zu haben.
- Die Prozesse sind mittels der PID (**P**rocess **I**dentification Number) durchnummeriert.
- Die Prozesse sind in einer Baumhierarchie angeordnet.

Wichtige Prozesse

- Mutter **aller** Prozesse ist der Prozess `init` mit der PID 1. Wird er beendet, wird der Rechner runtergefahren.
- Die Anmeldung eines Benutzers entspricht dem Login-Prozess, der dem neuen Benutzer gehört, und der die Mutter aller von diesem Benutzer gestarteten Prozesse ist.
- Es gibt eine Reihe systemrelevanter Prozesse, die im Hintergrund laufen (so genannte Dämonen). Sie übernehmen Aufgaben wie die Netzwerkanbindung, die Druckausgabe, Darstellung der graphischen Benutzeroberfläche etc.

Das Kommando top

```

load averages:  0.07,  0.09,  0.08                                17:27:20
49 processes:  47 sleeping, 1 running, 1 on cpu
CPU states: 97.8% idle,  0.4% user,  1.8% kernel,  0.0% iowait,  0.0% swap
Memory: 128M real, 30M free, 33M swap in use, 1321M swap free

  PID USERNAME  THR PRI NICE   SIZE   RES STATE   TIME    CPU COMMAND
  570 mheiste    1  46   4  7208K  5640K sleep    0:01   3.85% emacs
  544 mheiste    1  58   0  2232K  2008K cpu       0:00   0.53% top5.7
  368 mheiste    1  59   0  3680K  2832K sleep    0:02   0.27% xterm
  551 mheiste    1  48   0  1808K  1360K sleep    0:00   0.20% ksh
  401 mheiste    1  59   0  3472K  2424K sleep    0:00   0.19% ctwm
  550 mheiste    1  58   0  3672K  2816K run      0:00   0.15% xterm
  272 root       11  58   0  2296K  1544K sleep    7:53   0.03% mibiisa
  221 root       11  58   0  3232K  2288K sleep    1:02   0.03% nscd
  356 mheiste    1  58   0  4008K  3240K sleep    0:00   0.02% xsession
  248 root        1  58   0  1000K   552K sleep    0:02   0.01% utmpd
 4125 root        1  30   0  1880K   960K sleep   10:28   0.00% sshd1
  184 root        5  58   0  3208K  2296K sleep    4:17   0.00% automountd
  138 root        3  58   0  2112K  1520K sleep    0:55   0.00% nis_cachemgr
  305 root        1  48   0  3120K  1152K sleep    0:47   0.00% xdm
  402 root        4  58   0  1992K  1456K sleep    0:37   0.00% cachedfsd

```

Wichtige Prozessdaten bei top

PID	Process Identification Number
USERNAME	Eigentümer des Prozesses
SIZE	Gesamtgröße des Prozesses
RES	Größe des Prozesses im Arbeitsspeicher
STATE	Prozesszustand
TIME	Laufzeit des Prozesses
CPU	Prozentuale Angabe der CPU-Auslastung durch den Prozess
COMMAND	Kommando, das den Prozess gestartet hat

Das Kommando ps

ps [Optionen]

ps bietet abhängig vom System sehr unterschiedliche Optionen, deren Nutzung man deshalb fast immer nur durch Studium der dazugehörigen Manpage herausfindet.

Löschen eines Prozesses

Ist ein Prozess / Programm hängengeblieben, so kann man dieses Programm notfalls von Hand beenden. Dazu muss man zunächst Eigentümer des zu beenden Prozesses sein, und man muss seine PID kennen. Das Kommando lautet:

```
kill [-Killsignal] <PID>
```

Wird das Killsignal weggelassen, so wird der Prozess aufgefordert, sich ordentlich zu beenden. Bleibt der Prozess dennoch in der Prozesstabelle erhalten, so kann man ihn zur „Aufgabe“ zwingen, indem man das Killsignal 9 verwendet:

```
kill -9 <PID>
```

Die Shell

- Die Shell ist ein sogenannter Kommandointerpreter.
- Sie liefert den Prompt und ist auch an dessen Darstellung möglicherweise erkennbar.
- Sie ist die eigentliche Schnittstelle zum System auf der Kommandozeile.
- Alle bisherigen Befehle wurden bereits mittels der so genannten Bash (Bourne Again Shell) eingegeben!
- Die Shell legt sich wie eine Muschel um den Systemkern, interpretiert die eingegebenen Befehle und reicht ihre Ergebnisse an den Systemkern weiter.
- Sie bietet eingebaute Kommandos, die die Kommandoeingabe teilweise extrem vereinfachen.

Die verschiedenen Shells

Name	Befehl	Eigenschaften
Bourne-Shell	sh	Älteste Shell unter UNIX. Benannt nach ihrem Entwickler Steven R. Bourne.
C-Shell	csh	Weiterentwicklung der Bourne-Shell mit größerem Funktionsumfang, aber teilweise inkompatibel.
Korn-Shell	ksh	Erweiterung der Bourne- und C-Shell bei voller Kompatibilität zu beiden. Benannt nach ihrem Entwickler David Korn. RRZN-Standard.
Bourne-Again-Shell	bash	Massive Erweiterung der drei oberen Shells bei voller Kompatibilität. Unter Linux sehr weit verbreitet.

Die Eingabe-Prompts der Shells

Bourne-Shell	<code>unics:/home/zzzzheis: !\$</code>	Der Rechnername <code>unics</code> und das aktuelle Verzeichnis wird angezeigt.
C-Shell	<code>unics%</code>	Lediglich der aktuelle Rechner wird angezeigt.
Korn-Shell	<code>unics::138\$</code>	<code>unics</code> ist der Rechnername, und die Nummer gibt die aktuelle Anzahl der Kommandozeilen aus, die man bisher abgearbeitet hat. Löscht man den Inhalt der Datei <code>.sh_history</code> , so fängt die Nummerierung wieder bei 1 an.
Bourne-again-Shell	<code>unics:/home/zzzzheis: !\$</code>	Die Darstellung entspricht der der Bourne-Shell.

Um eine Shell wieder zu verlassen, benutzt man das Kommando

`exit`

Gibt man dieses Kommando in der Login-Shell ein, so meldet man sich vom System wieder ab.

Jedes XTerminal, das geöffnet wird, startet bereits mit der Standard-Shell des Systems (im Falle des RRZN ist das die Korn-Shell). Gibt man hier `exit` ein, so schließt man dieses Fenster wieder.

Jede Shell hat einen ganzen Satz eingebauter Kommandos, die den Umgang mit dem System erleichtern sollen. Alle verfügbaren Kommandos kann man über die Manpage der jeweiligen Shell herausfinden.

Zwei sehr nützliche Funktionen der Bash (auch die Korn-Shell hat sie) kennen wir schon:

- **Die History-Funktion:** D. h. mittels der Cursortasten kann man bereits eingebene Kommandos wieder abrufen und ggf. modifizieren.
- **Die Wildcards bzw. Metazeichen.**

Weitere Funktionen sind:

- **Automatische Vervollständigung von Namen:** Wann immer man einen Befehl oder einen Datei- bzw. Verzeichnisnamen eintippt, kann man zu jedem beliebigen Zeitpunkt durch Druck auf die TAB-Taste (Bash) oder durch **zweimaliges** Drücken der ESC-Taste (Korn-Shell) versuchen, das begonnene Wort komplettieren zu lassen.
- **Scrollen im Terminalfenster:** Durch die bisherigen Ausgaben in einem Terminalfenster kann man durch **gleichzeitiges** Drücken der Shift- und der PgUp- bzw. PgDn-Taste rauf- und runterscrollen.
- **Mehrere Kommandos nacheinander ausführen:** Bei der Eingabe von Kommandos kann man mehrere Befehle nacheinander ausführen lassen, indem man diese Kommandos durch Semikolons trennt.
- **Selbstständige Prozesse:** Durch Eingabe des Zeichens '&' nach einem Befehl und mit einem Leerraum dazwischen, wird der Befehl unabhängig von der aufrufenden Shell. Sie kann also weitere Befehle ausführen.

Umgebungsvariablen der Shell

Die Umgebungsvariablen der Shell teilen ihr mit, wo sie bestimmte Informationen finden kann, welche Geräte sie nutzen kann und wie die Umgebung der Shell auf dem jeweiligen System definiert ist. Alle gesetzten Umgebungsvariablen kann man sich mit dem Kommando

`set`

ansehen.

Setzen einer Umgebungsvariable:

Die Syntax lautet in dem Fall:

```
export <UMGEBUNGSVARIABLE>=<NeuerWert>
```

Beispielsweise kann man die Umgebungsvariable PRINTER auf den Wert hp4_rz_b219 setzen, indem man eingibt:

```
export PRINTER=hp4_rz_b219
```

Permanentes Setzen einer Umgebungsvariable:

Mit der `export`-Funktion wird eine Umgebungsvariable nur für die Dauer der Anmeldung an der momentanen Shell gesetzt. Soll sie auch bei der nächsten Anmeldung und in allen geöffneten Terminalfenstern einen bestimmten Wert besitzen, so muss die Datei

`.myprofile`

im eigenen Homeverzeichnis entsprechend geändert werden. Dort muss einfach eine Zeile hinzugefügt werden, die die entsprechende `export`-Anweisung enthält.

Löschen einer Umgebungsvariable:

Um den Inhalt einer Umgebungsvariable zu löschen, benutzt man den Befehl

```
unset <UMGEBUNGSVARIABLE>
```

Inhalt einer Umgebungsvariable ansprechen:

Setzt man vor den Namen einer Umgebungsvariable das Dollarzeichen '\$', so referenziert man deren Inhalt. Angenommen die Umgebungsvariable PATH sei auf das eigene Homeverzeichnis gesetzt, ihr Inhalt sei also beispielsweise

```
/home/zzzzheis
```

Die Umgebungsvariable HOME kann man direkt auf den Wert von PATH setzen, indem man das Dollarzeichen benutzt:

```
export HOME=$PATH
```

Ausgabe einer Umgebungsvariable:

Mittels der Eingabe von

```
echo $<UMGEBUNGSVARIABLE>
```

kann man sich den Inhalt einer bestimmten Umgebungsvariable anzeigen lassen.

Wichtige Umgebungsvariablen

PATH	Durch einen Doppelpunkt getrennte Liste aller Pfade, die bei Aufruf eines Kommandos durchsucht werden.
HOSTNAME	Name des Rechners.
PRINTER	Name des Standarddruckers.
PS1	Prompt-String der Login-Shell.
HOME	Pfad zum Homeverzeichnis.
USER	Benutzername der angemeldeten Person.
...	

Ein- / Ausgabeumlenkung und Kommandoverknüpfung

Alle vier vorgestellten Shells unterstützen die Eingabeumlenkung und die Kommandoverknüpfung. In der Darstellung beziehen wir uns hier auf die Syntax der Korn-Shell und der Bash.

- **Ein- / Ausgabeumlenkung:** Man kann die Ausgabe eines Prozesses umlenken. D. h. die Ausgabe kann auf einem anderen Gerät stattfinden als es normalerweise der Fall ist. So kann man beispielsweise die Ausgabe des `ls`-Kommandos in eine Datei umlenken, so dass das Inhaltverzeichnis nicht auf dem Bildschirm angezeigt wird, sondern in einer Textdatei gespeichert ist. Ebenso könnte man die Ausgabe auch direkt auf einen Drucker umleiten.
- **Kommandoverknüpfung (sog. Pipes):** Man kann die Ausgabe eines Prozesses direkt als Eingabe in einen anderen Prozess weiterleiten. So könnte man beispielsweise die Ausgabe einer Suchfunktion direkt an einen Befehl weiterleiten, der die Ergebnisse der Suchfunktion nochmals nach bestimmten Merkmalen filtert.

Zur Eingabeumlenkung verwendete Zeichen:

- > Die Ausgabe des links stehenden Befehles wird nach rechts umgelenkt.
- >> Die Ausgabe des links stehenden Befehles wird rechts angehängt.

Beispiel für > und >>:

die Eingabe von

```
ls > ls-Datei.txt
```

leitet die Ausgabe des `ls`-Kommandos in die Datei `ls-Datei.txt` um. Falls die Datei noch nicht existiert, wird sie erzeugt, und falls sie existiert, wird sie überschrieben.

```
ls >> ls-Datei.txt
```

hängt die Ausgabe des `ls`-Kommandos an den Inhalt der Datei `ls-Datei.txt` an. Existiert `ls-Datei.txt` noch nicht, so wird sie erzeugt.

Kommandoverknüpfung (Pipe):

Auf die Ausgabe eines Kommandos kann man ein weiteres Kommando anwenden, das die Ausgabedaten des anderen Kommandos auswertet. Durch den Trennstrich '|' wird das syntaktisch dargestellt:

Kommando 1 | Kommando 2

Die Ausgabe von Kommando 1 wird als Eingabe (Parameter) von Kommando 2 genutzt.

Beispiel zur Pipe:

Das Kommando `grep` durchsucht eine angegebene Datei nach einer Zeichenkette und gibt bei Erfolg die Zeile(n) der Datei aus, die die gesuchte Zeichenkette enthält.

Durch Eingabe von

```
ls | grep pdf
```

erhält man als Ausgabe alle Zeilen des Inhaltverzeichnis, die die Zeichenkette 'pdf' enthalten.

Nützliche Befehle

- clear
- grep
- wc
- find

Das Kommando `clear`:

`clear` löscht den aktuellen Inhalt eines Terminalfensters.

Das Kommando grep:

Die Syntax lautet:

```
grep <Zeichenkette> <Datenquelle>
```

grep durchsucht die Datenquelle (in der Regel ein Dateiname) nach der Zeichenkette und gibt alle Zeilen aus, in denen Die Zeichenkette vorkommt.

Das Kommando `wc`:

Die Syntax lautet:

```
wc [Option] <Datenquelle>
```

`wc` zählt Worte, Zeilen oder Buchstaben in einer oder mehreren Datenquellen. Möchte man mehrere Datenquellen angeben, so trennt man ihre Angabe durch die Leertaste. Die Optionen sind:

- m Zeichen werden gezählt.
- C Zeichen werden gezählt.
- l Zeilen werden gezählt.
- w Worte werden gezählt.

Das Kommando `find`:

Die Syntax lautet:

```
find <StartVerzeichnis> [Option] <Suchbegriff>
```

Die wohl wichtigste Option ist `'-name'`. Wird der Suchbegriff (Wildcards sind erlaubt!) dann als Zeichenkette in Anführungsstriche gesetzt, so werden alle Dateien / Verzeichnisse ab dem angegebenen Startverzeichnis nach dem Suchbegriff durchsucht.