UNIX – Grundkurs

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.

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.
- 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 ...

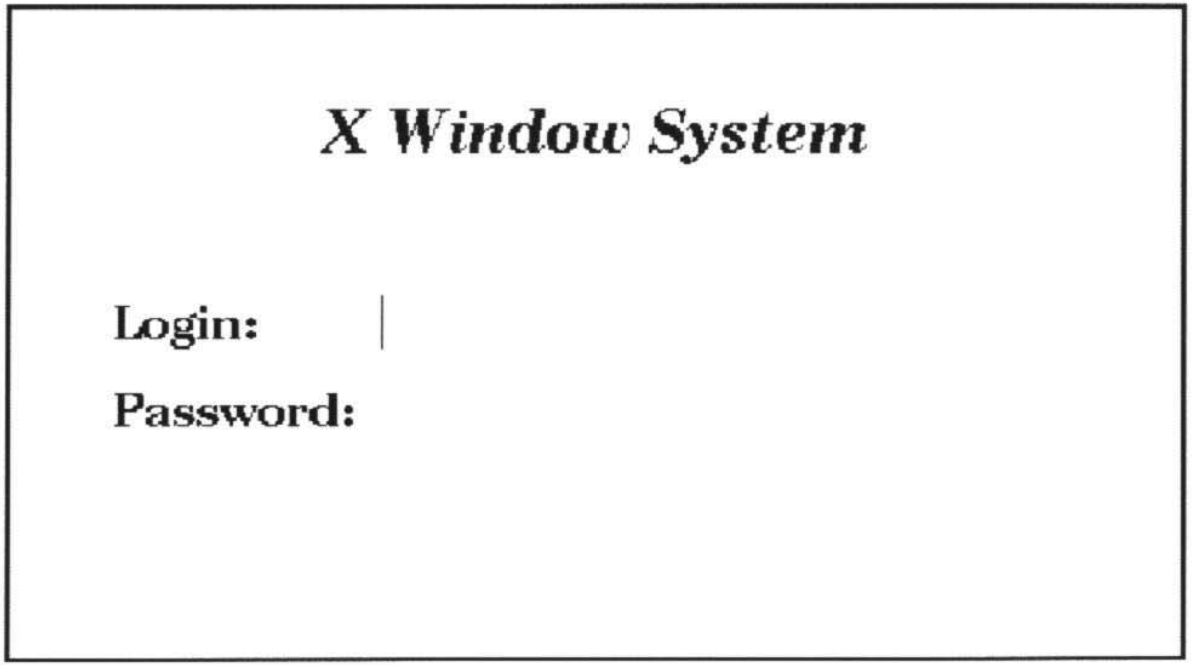
Anmeldung am System:

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

Zwei Arten der Anmeldung:

- grafisch
- textorientiert (Konsole)

Grafische Anmeldung:



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 sogennantes 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**.

Manpages

Eine wichtige Funktion sind die sogenannten Manpages. Durch Eingabe von

```
man <Befehl>
```

kann man sich zu jedem Befehl ein englischsprachige Hilfefunktion aufrufen. 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

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

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]	brif.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 speicherrestistent 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 **Id**entification 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 entpricht 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 (sogenannte 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% cachefsd
```

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

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>
```

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.

Verzeichniswechsel

```
cd [Zielverzeichnis]
```

Ausgabe des aktuellen Pfades

```
pwd
```

Verzeichnisinhalt

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

Optionen:

`-l` Listenartige Ausgabe mit zusatzinformationen.

`-a` Es werden alle Datein (auch versteckte) angezeigt.

Löschen

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

Die Shell

- Die Shell ist ein sogenannter Kommandointerpreter.
- Sie liefert den Prompt und ist auch an dessen Darstellung erkennbar.
- Sie ist die eigentliche Schnittstelle zum System auf der Kommandozeile.
- Alle bisherigen Befehle wurden bereits mittels der Korn-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	`unics:/home/zzzzheis:!$`	Der Rechnername `unics` und das aktuelle Verzeichnis wird angezeigt.
C-Shell	`unics%`	Lediglich der aktuelle Rechner wird angezeigt.
Korn-Shell	`unics::138$`	`unics` ist der Rechnername und die Nummer gibt die aktuelle Anzahl der Kommandozeilen aus, die man bisher abgearbeitet hat. Löscht man den Inhalt der Datei `.sh_history`, so fängt die Nummerierung wieder bei 1 an.
Bourne-again-Shell	`unics:/home/zzzzheis:!$`	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 Korn-Shell 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.**

Drei 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 **zweimaliges** Drücken der ESC-Taste 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 am System gesetzt. Soll sie auch bei der nächsten Anmeldung einen bestimmten Wert besitzen, so muss die Datei

`.myprofile`

im eigenen Homeverzeichnis entsprechend geändert werden. Dort muss einfach eine Zeile hintzugefügt 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 eigenen Homeverzeichnis gesetzt, ihr Inhalt sei also

```
/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.

...