D Einführung in UNIX

D.1 Benutzerumgebung

- die voreingestellte Benutzerumgebung umfaßt folgende Punkte:
  - ➤ Benutzername
  - ➤ Identifikation **(User-Id und Group-Ids)**
  - ➤ Home-Directory
  - ➤ Shell

D.2 Sonderzeichen

- einige Zeichen haben unter UNIX besondere Bedeutung
- Funktionen:
  - ➤ Korrektur von Tippfehlern
  - ➤ Steuerung der Bildschirm-Ausgabe
  - ➤ Einwirkung auf den Ablauf von Programmen

D.2 Sonderzeichen (2)

- die Zuordnung der Zeichen zu den Sonderfunktionen kann durch ein UNIX-Kommando (***stty(1)***) verändert werden
- die Vorbelegung der Sonderzeichen ist in den verschiedenen UNIX-Systemen leider nicht einheitlich
- Übersicht:

<BACKSPACE>	letztes Zeichen löschen (häufig auch **<DELETE>**)
<DELETE>	alle Zeichen der Zeile löschen (häufig auch **<CTRL>U** oder **<CTRL> X**)
<CTRL>C	Interrupt - Programm wird abgebrochen
<CTRL>\	Quit - Programm wird abgebrochen + core-dump
<CTRL>Z	Stop - Programm wird gestoppt (nur in csh)
<CTRL>D	End-of-File
<CTRL>S	Ausgabe am Bildschirm wird angehalten
<CTRL>Q	Ausgabe am Bildschirm läuft weiter

D.3 Dateien und Dateisystem

1 Allgemeines

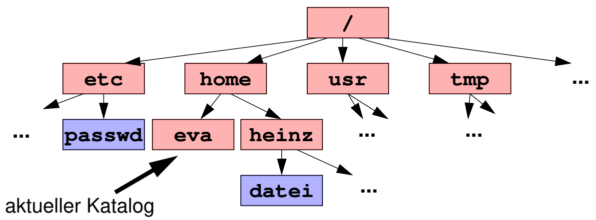
- Dateien sind abstrakte Gebilde zur Speicherung von Daten auf einem Hintergrundspeicher (Festplatte, Diskette)
  - ◆ Verbergen die reale Struktur (Sektoren, Zylinder, etc.)
  - ◆ Haben einen Namen (Zeichenkette)
  - ◆ UNIX verwaltet Zugriffsrechte (welcher Benutzer darf lesen, schreiben oder ein Programm in der Datei ausführen)

2 Struktur des Dateisystems

- Hierarchische Strukturierung des Dateisystems
  - ◆ reguläre Dateien
    - – Können beliebige Daten speichern (Texte, Zahlen, Programme, ...)
    - – Einfache Bedienung (öffnen, lesen, schreiben)
  - ◆ Dateikataloge *(Directories)*
    - – Enthalten reguläre Dateien oder weitere Kataloge
  - ➥ baumförmige Struktur des Dateisystems
  - ◆ spezielle Dateien *(special files)*
    - – verweisen i. a. auf Peripheriegeräte - können aber meist wie reguläre Dateien verwendet werden
  - ◆ symbolic links
    - – verweisen auf eine (beliebige) andere Datei
  - ◆ Dateien werden durch Aneinanderreihung der Katalognamen und des Dateinamens eindeutig benannt

3 Pfade im Dateisystem

- Baumstruktur & Pfade im Dateibaum



◆ z. B. **/home/heinz/datei** **/tmp** **../heinz/datei**

◆ **/** ist Trennsymbol *(Slash)* zwischen Pfadkomponenten
Pfade, die mit **/** beginnen, starten im Wurzelkatalog *(root)*
sonst Beginn implizit im aktuellem Katalog

◆ jeder Katalog enthält Verweise auf sich selbst **(.)** und auf den darüberliegenden Katalog **(..)** — in Pfadnamen verwendbar!

◆ aktueller Katalog kann mit Kommando **cd** gewechselt werden
z. B. **cd ../heinz**

4 Zugriffsrechte

- jede Datei ist mit individuellen Zugriffsrechten versehen, die für
  - ➤ den **Eigentümer** der Datei ***(user)***
  - ➤ alle Benutzer aus der **Gruppe**, zu der die Datei gehört ***(group)***
  - ➤ alle übrigen Benutzer ***(others)***

getrennt gesetzt werden können.

- Zugriffsrechte werden unterteilt in:

r	Leserecht	
w	Schreibrecht	
x	bei regulären Dateien:	Ausführungsrecht = Kommando
	bei Directories:	Zugriff durch Directory über einen Pfad

- Zugriffsrechte können vom Eigentümer einer Datei mit Hilfe des Kommandos ***chmod(1***) verändert werden

D.4 UNIX-Kommandointerpreter - Shell

auf den meisten Rechnern stehen verschiedene Shells zur Verfügung:

sh	**Bourne-Shell** - erster UNIX-Kommandointerpreter (vor allem für Kommandoprozeduren geeignet)
ksh	**Korn-Shell** - ähnlich wie Bourne-Shell, aber mit eingebautem Zeileneditor (vi- oder emacs-Modus)
csh	**C-Shell** (stammt aus der Berkeley-UNIX-Linie) - vor allem für interaktive Benutzung geeignet
tcsh	**erweiterte C-Shell** - enthält zusätzliche Edier-Funktionen, ähnlich wie Korn-Shell
bash	Shell der GNU-Distribution

1 Aufbau eines UNIX-Kommandos

UNIX-Kommandos bestehen aus:

- **Kommandonamen**
  (der Name einer Datei in der ein ausführbares Programm oder eine Kommandoprozedur für die Shell abgelegt ist)
- einer Reihe von **Optionen** und **Argumenten**
- Kommandoname, Optionen und Argumente werden durch Leerzeichen oder Tabulatoren voneinander getrennt
- Optionen sind meist einzelne Zeichen, denen ein **–** vorangestellt ist
- Argumente sind häufig Namen von Dateien, die von dem Kommando bearbeitet werden

Nach dem Kommando wird automatisch in allen Directories gesucht, die in der *Environment-Variablen* **$PATH** aufgelistet sind.

2 Vordergrund- / Hintergrundprozeß

- die Shell meldet mit einem Promptsymbol (z. B. `faui09%`), daß sie ein Kommando entgegennehmen kann
- die Beendigung des Kommandos wird abgewartet, bevor ein neues Promptsymbol ausgegeben wird - **Vordergrundprozeß**
- wird am Ende eines Kommandos ein **&**-Zeichen angehängt, erscheint sofort ein neues Promptsymbol - das Kommando wird im Hintergrund bearbeitet - **Hintergrundprozeß**

2 Vordergrund- / Hintergrundprozeß (2)

- **Jobcontrol**:
  - ➤ durch **<CTRL>Z** kann die Ausführung eines Kommandos *(Job)* angehalten werden - es erscheint ein neues Promptsymbol
  - ➤ funktioniert nicht in der *Bourne-Shell* und nicht in allen UNIX-Versionen
- die Shell (*csh*, *tcsh*, *ksh*) stellt einige Kommandos zur Kontrolle von Hintergrundjobs und gestoppten Jobs zur Verfügung:

jobs	Liste aller existierenden Jobs
bg %n	setze Job **n** im Hintergrund fort
fg %n	hole Job **n** in den Vordergrund
stop %n	stoppe Hintergrundjob **n**
kill %n	beende Job **n**

3 Ein- und Ausgabe eines Kommandos

- jedes Programm wird beim Aufruf von der Shell mit 3 E/A-Kanälen versehen:

stdin	Standard-Eingabe (Vorbelegung = Tastatur)
stdout	Standard-Ausgabe (Vorbelegung = Terminal)
stderr	Fehler-Ausgabe (Vorbelegung = Terminal)

- diese E/A-Kanäle können auf Dateien umgeleitet werden oder auch mit denen anderer Kommandos verknüpft werden ***(Pipes)***

4 Umlenkung der E/A-Kanäle auf Dateien

- die Standard-E/A-Kanäle eines Programms können von der Shell aus umgeleitet werden
  (z. B. auf reguläre Dateien oder auf andere Terminals)
- die Umleitung eines E/A-Kanals erfolgt in einem Kommando (am Ende) durch die Zeichen **<** und **>**, gefolgt von einem Dateinamen
- durch **>** wird die Datei ab Dateianfang überschrieben, wird statt dessen **>>** verwendet, wird die Kommandoausgabe an die Datei angehängt
- Syntax-Übersicht

<datei1	legt den Standard-Eingabekanal auf **datei1**, d. h. das Kommando liest von dort
>datei2	legt den Standard-Ausgabekanal auf **datei2**
>&datei3	*(csh, tcsh)* legt Standard- und Fehler-Ausgabe auf **datei3**
2>datei4	*(sh, ksh)* legt den Fehler-Ausgabekanal auf **datei4**
2>&1	*(sh, ksh)* verknüpft Fehler- mit Standard-Ausgabekanal (Unterschied zu **">datei 2>datei"** !!!)

5 Pipes

- durch eine **Pipe** kann der Standard-Ausgabekanal eines Programms mit dem Eingabekanal eines anderen verknüpft werden
- die Kommandos für beide Programme werden hintereinander angegeben und durch | getrennt
- Beispiel:

```
ls -al | wc
```

  ➤ das Kommando **wc** (Wörter zählen), liest die Ausgabe des Kommandos **ls** und gibt die Anzahl der Wörter (Zeichen und Zeilen) aus

- *Csh* und *tcsh* erlauben die Verknüpfung von Standard-Ausgabe und Fehler-Ausgabe in einer Pipe:
  ➤ Syntax: |**&** statt |

6 Kommandoausgabe als Argumente

- die Standard-Ausgabe eines Kommandos kann einem anderen Kommando als Argument gegeben werden, wenn der Kommandoaufruf durch ` ` geklammert wird
- Beispiel:

```
rm `grep -l XXX *`
```

◆ das Kommando **grep -l XXX** liefert die Namen aller Dateien, die die Zeichenkette **XXX** enthalten auf seinem Standard-Ausgabekanal

➥ es werden alle Dateien gelöscht, die die Zeichenkette **XXX** enthalten

7 Quoting

Wenn eines der Zeichen mit Sonderbedeutung (wie **<**, **>**, **&**) als Argument an das aufzurufende Programm übergeben werden muß, gibt es folgende Möglichkeiten dem Zeichen seine Sonderbedeutung zu nehmen:

- Voranstellen von \ nimmt genau einem Zeichen die Sonderbedeutung
  \ selbst wird durch \\ eingegeben
- Klammern des gesamten Arguments durch **" "**,
  " selbst wird durch **"** angegeben
- Klammern des gesamten Arguments durch **' '**,
  ' selbst wird durch **'** angegeben

8 Environment

- Das *Environment* eines Benutzers besteht aus einer Reihe von Text-Variablen, die an alle aufgerufenen Programme übergeben werden und von diesen abgefragt werden können
- Mit den Kommandos ***env(1)*** (SystemV) bzw. ***printenv(1)*** (BSD) können die Werte der Environment-Variablen abgefragt werden:

```
% env
EXINIT=se aw ai sm
HOME=/home/jklein
LOGNAME=jklein
MANPATH=/local/man:/usr/man
PATH=/home/jklein/.bin:/local/bin:/usr/ucb:/bin:/usr/bin:
SHELL=/bin/sh
TERM=vt100
TTY=/dev/ttyp0
USER=jklein
HOST=faui43d
```

8 Environment (2)

- Mit dem Kommando ***env(1)*** kann das Environment auch nur für ein Kommando gezielt verändert werden
- Auf Environment-Variablen kann – wie auf normale Shell-Variablen auch – durch **$Variablenname** in Kommandos zugegriffen werden
- Mit dem Kommando ***setenv(1)*** (C-Shell) bzw. ***set*** und ***export*** (Shell) können Environment-Variablen verändert und neu erzeugt werden:

```
% setenv PATH "$HOME/.bin.sun4:$PATH"

$ set PATH="$HOME/.bin.sun4:$PATH"; export PATH
```


8 Environment (*2*)

- Überblick über einige wichtige Environment-Variablen

$USER	Benutzername (BSD)
$LOGNAME	Benutzername (SystemV)
$HOME	Homedirectory
$TTY	Dateiname des Login-Geräts (Bildschirm)
$TERM	Terminaltyp (für bildschirmorientierte Programme, z. B. ***vi(1)***)
$PATH	Liste von Directories, in denen nach Kommandos gesucht wird
$MANPATH	Liste von Directories, in denen nach Manual-Seiten gesucht wird (für Kommando ***man(1)***)
$SHELL	Dateiname des Kommandointerpreters (wird teilweise verwendet, wenn aus Programmen heraus eine Shell gestartet wird)

D.5 Auswahl einiger wichtiger Kommandos

ls		Directory auflisten wichtige Optionen:
	-l	langes Ausgabeformat
	-a	auch mit **.** beginnende Dateien werden aufgeführt
cat		Datei(en) hintereinander ausgeben
more, less		Dateien bildschirmweise ausgeben
head		Anfang einer Datei ausgeben (Vorbel. 10 Zeilen)
tail		Ende einer Datei ausgeben (Vorbel. 10 Zeilen)
cp		Datei(en) kopieren
rcp		Datei(en) auf anderen Rechner kopieren
mv		Datei(en) verlagern
ln		Datei linken (weiteren Verweis auf gleiche Datei erzeug.)
ln -s		Symbolic link erzeugen
rm		Datei(en) löschen

D.5 Auswahl einiger wichtiger Kommandos (2)

mkdir	Directory erzeugen
rmdir	Directory löschen (muß leer sein!!!)
pr, lp, lpr	Datei ausdrucken
chmod	Zugriffsrechte einer Datei verändern
wc	Zeilen, Wörter und Zeichen zählen
grep, fgrep	nach bestimmten Mustern bzw. Zeichenketten suchen
sed	Stream-Editor
tr	Zeichen abbilden
awk	pattern-scanner
sort	sortieren
who, finger	Liste der gerade am Rechner aktiven Benutzer
rwho	wie who, aber über alle Rechner im lokalen Netz
rlogin, slogin	login an anderem Rechner