Textverarbeitung in der Unix-Shell

Peter Keel,
Király András

1. Februar 2003

Inhaltsverzeichnis

1 Die Shell **3**
1.1 Prompt . . . 3
1.2 Verzeichnisse . . . 3
1.3 Home-Verzeichnis . . . 3
1.4 Versteckte Dateien . . . 4
1.5 Parameter . . . 4
1.6 Wildcards . . . 4
1.7 Gross/Kleinschreibung . . . 4
1.8 Umleitungen . . . 4
1.9 Pipe . . . 4
1.10 Abbrechen und Unterbechen . . . 5
1.11 Commandline-Completion . . . 5

2 Kommandos – Navigation im System **5**
2.1 pwd . . . 5
2.2 mkdir . . . 5
2.3 cd . . . 5
2.4 ls . . . 6
2.5 rmdir . . . 8
2.6 touch . . . 8
2.7 cp . . . 9
2.8 rm . . . 9
2.9 mv . . . 10

3 Kommandos – Hilfe **10**
3.1 man . . . 10

4 Kommandos – Textverarbeitung **10**
4.1 cat . . . 10
4.2 wc . . . 11
4.3 more . . . 11
4.4 less . . . 11
4.5 head . . . 11
4.6 tail . . . 12
4.7 sort . . . 12
4.8 uniq . . . 12
4.9 grep . . . 13
4.10 cut . . . 13
4.11 paste . . . 13

4.12 tr 14
4.13 comm 14
4.14 join 15
4.15 diff 15
4.16 split 15
4.17 expand/unexpand 15
4.18 fold 15
4.19 pr 15
4.20 fmt 16

5 Kommandos – Systeminformation 16
5.1 date 16
5.2 df 16

6 Glossar 16
6.1 Gleichwertige Begriffe 16
6.2 Tasten und Zeichen 16
6.3 Kommandoübersicht 17

1 Die Shell

Die Shell ermöglicht die Eingabe von Befehlen via Texteingabe (Kommandozeile). Unter Unix existieren ganze Mengen dieser Shells, je nach Unix wird die eine oder andere bevorzugt: `csh, tcsh, ksh, bash` und `zsh`. Unter Linux normalerweise `bash`, unter MacOS X die `tcsh`. Die Shells unter Unix sind das Äquivalent zur command.com respektive cmd unter Windows, jedoch ungleich mächtiger.

1.1 Prompt

Das erste was man von der Shell sieht ist ein Fenster mit einem Prompt, welches anzeigt dass man nun Befehle eingeben kann. Dieses sieht vielleicht so aus:

```
bash$
```

Man kann es anpassen. Der Superuser (root) hat normalerweise ein Octothorp (#) statt eines Dollarzeichens ($), und auch sonst kann es beliebig anpassen, dazu später mehr.
Beispiele:

```
hostname:~/verzeichnis$
```

1.2 Verzeichnisse

Unter Unix sind Verzeichnisse (Directories, Folders) durch einen Slash (/) getrennt. Die Schreibweise von Verzeichnissen in URLs (http://www/verzeichnis/) korrespondiert auch damit, weil das WWW unter Unix erfunden wurde. Es gibt keine Laufwerksbuchstaben, stattdessen werden alle Geräte und Disks irgendwo in den Verzeichnisbaum eingehängt. Die Wurzel des Baums ist dabei das root-Verzeichnis "/".

1.3 Home-Verzeichnis

Jeder Benutzer besitzt ein Home-Verzeichnis, welches dasjenige Verzeichnis ist, welches er nach dem einloggen oder dem Aufruf der Shell zuerst zu sehen bekommt. Es ist auch das einzige Verzeichnis in dem er Schreibrechte besitzt (von Ausnahmen wie `/tmp` abgesehen). Es kann auch durch die Tilde (~) angezeigt werden.

1.4 Versteckte Dateien

Dateien die mit einem Punkt (".") beginnen sind grundsätzlich versteckte Dateien. Dies ist nicht vor allem aus Sicherheitsgründen so, sondern der Übersichtlichkeit halber, da diese im Normalfall uninteressant sind.

1.5 Parameter

Unix-Befehle (Kommandos,Programme) kennen viele Parameter, mit denen sich einstellen lässt wie ein Programm reagieren soll, oder auf was es reagieren soll. Dabei gilt dass diese immer mit einem Leerschlag (Space) vom Programmnamen getrennt sein müssen. Optionen die das Programm selber kennt, werden mit dem Dash ("$-$") oder dem doppelten Dash ("$--$") vom Kommando abgetrennt. Ein typischer Parameter ist "$-$h" oder "$--$help" um Hilfe zum Kommando zu bekommen.

1.6 Wildcards

Wildcards sind Zeichen, die für beliebige andere stehen können. Es sind dies das Fragezeichen ("?") welches für EIN beliebiges anderes Zeichen steht, und der Stern ("*") welcher für beliebig viele andere Zeichen steht. Sie werden typischerweise verwendet um einem Programm ganz viele Dateinamen als Parameter zu übergeben.

1.7 Gross/Kleinschreibung

Diese ist unter Unix wichtig. Eine Datei namens "Dateiïst durchaus nicht dieselbe wie die Datei namens "datei" oder die andere namens "DATEI", oder "DaTeiöder "DatEi" usf..

1.8 Umleitungen

Mit "$>$" lässt sich die Ausgabe eines Programms in eine Datei umleiten, respektive mit "$<$" die Eingabe. Die doppelte Umleitung "$>>$" fügt die Ausgabe eines Programms an einer Datei an, ohne diese zu überschreiben.

1.9 Pipe

Die Pipe ("$|$") ermöglicht es die Ausgabe eines Kommandos direkt einem anderen Kommando zu übergeben. Derselbe Effekt kann durch die Umlei-

tung in eine Datei und das Lesen aus dieser erreicht werden, doch mit der Pipe geht das ohne Umweg.

1.10 Abbrechen und Unterbechen

Kommandos lassen sich mit Control-C abbrechen, und mit Control-Z unterbrechen (und in den Hintergrund schieben).

1.11 Commandline-Completion

Wenn man den Anfang eines Kommandos oder Dateinamens weis, kann man durch Eingabe der ersten paar Buchstaben und die Tabulator-Taste den Namen automatisch komplettieren lassen. Dies geht in den meisten modernen shells. ”tou<tab>” wird also automatisch zu ”touch” expandiert.

2 Kommandos – Navigation im System

2.1 pwd

Print Working Directory
Dieses Kommando zeigt uns an wo im Verzeichnisbaum wir uns befinden.

```
hostname:~$ pwd
/home/user
```

2.2 mkdir

Make Directory
Damit können wir ein Verzeichnis erstellen.

```
hostname:~$ mkdir verzeichnis
```

2.3 cd

Change Directory
Damit können wir nun das Verzeichnis wechseln. Der Punkt designiert dabei das momentane Verzeichnis, der zweifache Punkt das Vorangehende. cd ohne Parameter wechselt in das home-Verzeichnis des Benutzers.

```
hostname:~$ cd verzeichnis
hostname:~/verzeichnis$

hostname:~/verzeichnis$ cd ..
hostname:~$
```

oder

```
hostname:~/verzeichnis$ cd
hostname:~$
```

2.4 ls

List
Zeigt uns die Verzeichnisse und Dateien in einem Verzeichnis an. Der wichtigste Parameter ist ”-l”, welcher ein langes Listing anzeigt welches mehr informationen enthält. Ein weiterer wichtiger Parameter ist ”-a”, damit werden auch die Versteckten Dateien angezeigt. Man kann die Parameter auch zusammenfassen (”-al”).

```
hostname:~$ ls
verzeichnis
hostname:~$ ls -l
total 4
tdrwxr-xr-x 2 user user 4096 2003-01-10 13:31 verzeichnis

hostname:~$ ls -a
. .. .alias .bash_logout .bash_profile .bashrc verzeichnis

hostname:~$ ls -al
total 36
t drwxr-xr-x 3 user user 4096 2003-01-10 13:31 .
tdrwxr-xr-x 197 root root 12288 2003-01-10 13:29 ..
t-rw-r-r- 1 user user 266 1999-07-08 22:53 .alias
t-rw-r-r- 1 user user 175 2000-10-02 17:09 .bash_logout
t-rw-r-r- 1 user user 556 2002-09-12 00:51 .bash_profile
t-rw-r-r- 1 user user 1263 2002-11-04 10:02 .bashrc
tdrwxr-xr-x 2 user user 4096 2003-01-10 13:31 verzeichnis
```

In der langen Darstellung sehen wir auch diverse merkwürdige Nummern und Namen. Der Reihe nach sind dies Filetyp und Permissions (die Rechte) auf die Dateien, die Grösse in Blöcken, der Besitzer, die Gruppe, die wirkliche Grösse in Bytes, das Datum der letzten Modifikation und den Dateinamen.

Am Beispiel:

```
hostname:~$ ls -l
total 4
drwxr-xr-x 2 user user 4096 2003-01-10 13:31 verzeichnis
```

Der erste Teil "`drwxr-xr-x`" sagt uns zuerst einmal den Dateityp ("`d`") nämlich ein Verzeichnis. Dahinter folgen drei 3er-Blöckchen mit den Rechten. Zuerst kommen die Rechte des Besitzers, dann der Gruppe, und dann des Rests der Welt. Jedes Blöckchen gibt an ob er die Datei lesen (read "`r`"), schreiben (write, "`w`") oder ausführen (execute, "`x`") darf. Hier darf also der Besitzer lesen, schreiben und ausführen, die Gruppe nur lesen und ausführen, und der Rest der Welt ebenfalls nur lesen und ausführen. Speziell hier ist noch, dass das ein Verzeichnis ist, und hier gilt dass Äusführen das Recht ist ins Verzeichnis zu wechseln.
Danach folgt die Grösse in Blöcken auf der Disk, hier also 2. Die Blöcke sind meist 4KB gross. Das braucht uns aber jetzt nicht zu interessieren.
Danach kommt der Besitzer ("`user`") und die Gruppe (ebenfalls "`user`") dem die Datei gehört.
Nun kommt die Grösse der Datei ("`4096`") in Bytes. In diesem Fall ist das ein Verzeichnis, und als solches hat es unter Unix tatsächlich eine Grösse, weil es Platz braucht die Verzeichniseinträge zu verwalten.
Zum Schluss kommt noch das Datum der letzten Modifikation und der Dateiname.

2.5 rmdir

Remove Directory
Damit können wir Verzeichnisse löschen.

```
hostname:~$ ls
verzeichnis
hostname:~$ rmdir verzeichnis
hostname:~$ ls
```

2.6 touch

Unix hat keinen spezifischen Befehl zum anlegen von Dateien, deshalb muss ein anderes Programm herhalten. touch setzt eigentlich das Datum der letzten Modifikation einer Datei auf den momentanen Zeitpunkt. Falls

die Datei aber nicht existiert, wird sie neu angelegt.

```
hostname:~$ touch Datei
hostname:~$ ls -l
-rw-rw-r- 1 user user 0 2003-01-10 13:55 Datei
```

2.7 cp

Copy
Damit können wir Dateien kopieren.

```
hostname:~$ ls
Datei
hostname:~$ cp Datei Datei2
hostname:~$ ls
Datei Datei2
```

2.8 rm

Remove
Dieser Befehl löscht eine Datei. Wichtige Parameter sind "`-f`" was "force" bedeutet und ohne nachfragen löscht, und "`-r`" was "rekursiv" bedeutet und dazu benutzt werden kann ganze Verzeichnisbäume zu löschen. Normalerweise ist es unmöglich eine Datei danach wieder herzustellen, der Befehl ist also gefährlich und ein "`rm -rf *`" könnte das ganze aktuelle Verzeichnis löschen.

```
hostname:~$ ls
Datei Datei2
hostname:~$ rm Datei2
hostname:~$ ls
Datei
```

2.9 mv

Move
Dieser Befehl dient zum verschieben oder umbenennen von Dateien.

```
hostname:~$ ls
Datei Datei2 verzeichnis
hostname:~$ mv Datei verzeichnis
hostname:~$ mv Datei2 Datei3
hostname:~$ ls
Datei3 verzeichnis
hostname:~$ ls verzeichnis
Datei
```

3 Kommandos – Hilfe

3.1 man

Manual
Unter Unix besitzt jedes Kommando ein zugehöriges Manual. Dieses kann man mit "`man <Kommando>`" abrufen. Vorausgesetzt man weiss zu welchem Kommando man das Manual möchte...

4 Kommandos – Textverarbeitung

4.1 cat

Concatenate
Mit `cat` kann man den Inhalt einer Datei anzeigen, oder mehrere Dateien aneinander Anfügen.

```
hostname:~$ cat > Test
1 a AA
2 b BB
1 a AA
3 c CC
```
CTRL-C

```
hostname:~$ cat Test
1 a AA
2 b BB
1 a AA
3 c CC
```

4.2 wc

Word Count
`wc` zählt ganz einfach wieviele Zeilen, Wörter und Zeichen ein Text enthält.

```
hostname:~$ wc Test
4 12 28 Test
```

4.3 more

Manchmal möchte man Dateien Seitenweise angezeigt haben (um dann jedesmal "`more`" bestätigen können wenn man die nächste Seite sehen möchte).

4.4 less

Eine bessere Version von "`more`", mit der man auch nach oben scrollen kann.

4.5 head

Head zeigt nochmals die Datei an, diesmal nur den Kopf. Ein Parameter "`-<Nummer>`" kann angeben werden um entsprechend viele Zeilen auszugeben; per default sind es 10.

```
hostname:~$ head -1 Test
1 a AA
```

4.6 tail

Tail ist das Gegenteil von "`head`" und zeigt demgemäss das Ende der Datei an, auch hier kann man mit "`-<nummer>`" angeben wieviele Zeilen man möchte.

```
hostname:~$ tail -1 Test
3 c CC
```

4.7 sort

Mit `sort` können wir Dateien sortieren. Der Parameter "`-r`" sortiert in umgekehrter Reihenfolge. Um das gewünschte Resultat zu erhalten bedarf es oft diverser weiterer Parameter.

```
hostname:~$ sort Test
1 a AA
1 a AA
2 b BB
3 c CC
```

4.8 uniq

Dieser Befehl stellt sicher, dass eine Zeile nur einmal vorkommt. Er funktioniert jedoch nur, wenn die doppelten Zeilen aufeinanderfolgend sind.

```
hostname:~$ uniq Test
1 a AA
2 b BB
1 a AA
3 c CC
hostname:~$ sort Test > Test2
hostname:~$ uniq Test2
1 a AA
2 b BB
3 c CC
```

4.9 grep

Mit `grep` lässt sich nach Dateinhalten suchen. Wichtige Parameter sind "`-v`" der nach Zeilen sucht in denen das gesuchte NICHT vorkommt, und "`-r`" welches rekursiv durch den ganzen Verzeichnisbaum sucht ("`-r`" wird aber nicht von allen Unix-Versionen unterstützt).

```
hostname:~$ grep BB Test2
2 b BB
hostname:~$ grep -v BB Test2
1 a AA
3 c CC
```

4.10 cut

`cut` schneidet Spalten aus. Das Spaltentrennzeichen muss dabei mit dem Parameter "-d" (Delimiter) angegeben werden (ansonsten wird Tabulator angenommen) und die Felder mit "-f <von>-<bis>". Alternativ können auch mit "-b <von>-<bis>" absolute Positionen in Zeichen angebenen werden.

```
hostname:~$ cut -d '' '' -f 2-3 Test2
a AA
b BB
c CC
hostname:~$ cut -b 3-5 Test2
a A
b B
c C
```

4.11 paste

Das Gegenteil zu `cut`. Fügt Dateien Spaltenweise zusammen:

```
hostname:~$ paste Test Test2
1 a AA 1 a AA
2 b BB 2 b BB
1 a AA 3 c CC
```

```
3 c CC
```

4.12 tr

Translate
Mit `tr` kann man in Textfiles Wörter oder Zeichenfolgen durch andere ersetzen. `tr` ist etwas tricky zu handhaben falls man ganze Wortketten durch andere ersetzen möchte, kann weder zeichen durch mehrere Zeichen ersetzen noch die Reihenfolge der Buchstaben verändern. `tr` liest von der Standardeingabe, d.h. man muss ihm die Daten per Umleitung oder Pipe übergeben.

```
hostname:~$ tr a b < Test2
1 b AA
2 b BB
3 c CC
hostname:~$ cat Test2 | tr [:alnum:] b
b b bb
b b bb
b b bb
```

4.13 comm

Compare
Vergleicht zwei Textfiles, Zeile für Zeile. Die Parameter "`-1`", "`-2`", und "`-3`" unterdrücken dabei jeweils die Ausgabe der Zeilen die nur im ersten File, nur im zweiten File oder nur in Beiden vorkommen.

```
hostname:~$ comm Test Test2
1 a AA
2 b BB
1 a AA
3 c CC
hostname:~$ comm -3 Test Test2
1 a AA
```

4.14 join

`join` ermöglicht das Zusammenfügen von Dateien, wobei defaultmässig der kleinste gemeinsame Nenner gilt.

```
hostname:~$ join Test Test2
1 a AA a AA
2 b BB b BB
3 c CC c CC
```

4.15 diff

Differences
Dies ist der grosse Bruder von `comm`. `diff` kann wesentlich mehr, ist aber auch viel komplizierter zu handhaben.

4.16 split

Mit split kann man Dateien in Teile zerlegen. Mit dem Parameter "`-l`" kann man angeben wieviele Zeilen es pro Teil sein sollen, mit dem Parameter "`-b`" wieviele Zeichen. Default ist 1000 Zeilen.

4.17 expand/unexpand

Expandiert Tabulatoren (8 Zeichen breit) in 8 Leerzeichen und umgekehrt. `unexpand` funktioniert nur pro 8 Leerzeichen.

4.18 fold

Dieses Programm bricht Zeilen um. Damit lassen sich überlange Zeilen auf eine auf Konsole oder Printer darstellbare Grösse bringen.

4.19 pr

Prepariert eine Datei zu Ausgabe auf ASCII-Printern. Erstellt Seitennumerierung etc.

4.20 fmt

Formatiert Texte neu, damit sie einfacher gelesen werden können.

5 Kommandos – Systeminformation

5.1 date

Gibt das aktuelle Datum und Zeit aus.

5.2 df

Disk Free
Zeigt den freien Speicherplatz auf den Disks an.

6 Glossar

6.1 Gleichwertige Begriffe

Oftmals werden verschiedene deutsche und englische Begriffe gleichwertig verwendet. Es sind dies:

- Directory Verzeichnis, Folder, Ordner
- File, Datei
- Permissions, Rechte
- Space, Leerschlag
- Programm, Kommando, Befehl, Executable
- Terminal,Konsole

6.2 Tasten und Zeichen

Die Namen von Zeichen und Tasten variieren im Sprachgebrauch auch sehr, deshalb hier diejenigen die hier verwendet werden:

$\sim$	Tilde
/	Slash
\	Backslash
-	Dash oder Strich
$	Dollar
#	Octothorp oder Hash
?	Fragezeichen
$*$	Stern
|	Pipe
$>$	Umleitung Nach oder Grösser
$<$	Umleitung Von oder Kleiner
@	At
&	Ampersand
’	Tick
‘	Backtick
”	Anführungszeichen
$wedge$	Caret
()	Klammern
[]	Eckige Klammern
{}	Geschweifte Klammern

6.3 Kommandoübersicht

Eine Kurze Kommandoübersicht:

pwd (print working directory)	Aktuelles Verzeichnis anzeigen
mkdir (make directory)	Verzeichnis erstellen
cd (change directory)	Verzeichnis wechseln
ls (list)	Dateien auflisten
rmdir (remove directory)	Verzeichnis löschen
touch	Datei anlegen
cp (copy)	Datei kopieren
rm (remove)	Datei löschen
mv (move)	Datei verschieben
man (manual)	Manual ansehen
cat (concatenate)	Datei anzeigen oder zusammenfügen
wc (word count)	Wörter zählen
more	Datei seitenweise anzeigen
less	Datei seitenweise anzeigen
head	Kopf der Datei Anzeigen
tail	Fuss der Datei Anzeigen
sort	Sortieren
uniq	Duplikate aussortieren
grep	Finden
cut	Spalten ausschneiden
paste	Spalten einfügen
tr (translate)	Einzelne Zeichen ersetzen
comm (compare)	Dateien vergleichen
join	Dateien zusammenfügen
split	Dateien zerstückeln
diff	Dateien vergleichen
expand	Tabulatoren in Spaces umwandeln
unexpand	Spaces in Tabulatoren umwandeln
fold	Zeilen umbrechen
pr	Seitenlayout umwandeln
fmt	Seitenlayout umwandeln
date	Datum und Zeit anzeigen
df	Freien Diskplatz anzeigen