Kurze Einweisung in die Shell

- Betriebssystem Unix
- primäre Kommunikation mit dem System ist über Kommandozeile, graphische Systeme sind nur Aufsatz
- C ist speziell mit/für Unix-Betriebsysteme entwickelt worden
- das Programm, welches die Kommandozeile implementiert, heisst `bash` (bourne again shell)

Einzugebende Kommandos werden in der folgenden Form dargestellt:

```
# command
```

Dabei symbolisiert das `#` die Kommandozeile und darf nicht mit eingegeben werden. Bildschirmausgaben werden ebenfalls in dieser Schriftart dargestellt.

1 Hilfe

Anzeige von Informationen mit dem Kommando `man`:

```
# man bash
```

```
# man man
```

Optionen zu den Kommandos werden als `-o` oder auch als `--option` angegeben

Andere Informationsquelle:

```
# info bash
```

2 Dateien und Verzeichnisse

- zusammengehörige Informationen werden in einer Datei organisiert
- mehrere Dateien liegen in einem Verzeichnis
- Verzeichnisse können auch weitere Verzeichnisse enthalten (Unterverzeichnisse)
- Es gibt ein Wurzelverzeichnis `/`
- Unterteilung ein Unterverzeichnisse durch weitere `/`

Dateien und Verzeichnis sind durch ihren absoluten Pfad eindeutig identifiziert.

```
/usr/bin/bash
```

Bei vielen Verzeichnisebenen ist die Angabe von absoluten Pfaden mühsam, daher gibt es die Möglichkeit mit relativen Pfaden zu arbeiten.

Angenommen, man befindet sich um Verzeichnis `/usr`. Dann ist die Datei `/usr/bin/bash` auch über

```
./bin/bash
```

erreichbar. Der Punkt `.` steht für das aktuelle Verzeichnis (*working directory*).

Die Information `./` kann auch weggelassen werden.

Der Doppelpunkt `..` bezeichnet das übergeordnete Verzeichnis.

Befindet man sich im Verzeichis `/usr`, ist die Datei `/usr/bin/bash` auch ueber `../usr/bin/bash` erreichbar.

Bei Datei- und Verzeichnisnamen wird zwischen Gross und Kleinschreibung unterschieden, `testdatei` und `Testdatei` bezeichnen zwei verschiedene Dateien.

2.1 Erzeugen und Löschen von Verzeichnissen

Mit den Befehlen `mkdir` und `rmdir` können Verzeichniss erzeugt und gelöscht werden

```
# mkdir testdir
```

```
# rmdir testdir
```

Verzeichnisse koennen mit `rmdir` nur dann gelöscht werden, wenn sie komplett leer sind.

2.2 Anzeige von Dateien

Das Kommando `ls` dient zur Anzeige des Inhaltes von Verzeichnissen:

```
# ls /usr
```

```
# ls
```

Dateien, die mit einem Punkt beginnen, werden nicht anzeigt, sondern nur, wenn die Option `-a` angeben wird:

```
# ls -a
```

Die Option `-l` gibt zusätzliche Informationen aus:

```
# ls -l
drwxr-xr-x   2 cbecker  lsiii        512 May 18  2005 wls
```

Die 3. Spalte gibt den Besitzer der Datei an, die 4. Spalte die Zugehöigkeit zu

einer Gruppe. Es folgen die Größe der Datei und der Zeitpunkt der Erstellung bzw. letzten Modifikation. Zum Schluß erscheint der Name der Datei.

Der erste Eintrag gibt die Zugriffsrechte und die Art der Datei an. Dabei bedeuten:

`Auuugggooo`

A: Art der Datei, `-` für Datei, `d` für Verzeichnis

uuu: Zugriffsrechte auf Benutzerebene für Lesen (r), Schreiben (w) und Ausführen (x), z.B. bedeutet `r--` ausschliesslichen Lesezugriff, während `rwx` Vollzugriff bedeutet.

ggg: Zugriffsrechte auf Gruppenebene

ooo: Zugriffsrechte für alle, die weder Benutzer noch Gruppenangehörige sind.

`-rw-r-----` definiert für eine Datei Schreib- und Lesezugriff für den Benutzer, Lesezugriff für die Gruppe und keinen Zugriff für den Rest der Welt

Das `x` kennzeichnet eine Datei als ausführbar, d.h. sie kann direkt vom Betriebssystem gestartet werden:

```
# ls -la /usr/bin/ls
-r-xr-xr-x   1 root     bin        27400 Jan 24  2007 /usr/bin/ls
```

Ausführbar können übersetzte Programme, als auch sogenannte Shellskripte, d.h. in einer Datei hintereinander geschriebene Shell-Kommandos, sein.

Ausführbar im Zusammenhang mit einem Verzeichnis bedeutet, dass der Inhalt des Verzeichnisses angezeigt werden kann.

```
# mkdir testdir
```

```
# ls -lad testdir
drwxr-xr-x   2 cbecker  lsiii        512 Mar  7 12:06 testdir
```

Die Option `-d` bedeutet, dass das Verzeichnis selbst und nicht der Inhalt angezeigt werden soll, vgl.

```
# ls -la testdir
total 40
drwxr-xr-x   2 cbecker  lsiii        512 Mar  7 12:06 .
drwxr-xr-x 238 cbecker  lsiii      18944 Mar  7 12:07 ..
```

Das Verzeichnis ist noch leer, es werden nur die Metaverzeichnisse "aktuelles Verzeichnis" `.` und "übergeordnetes Verzeichnis" `..` angezeigt.

Die Rechte einer Datei oder eines Verzeichnisses können mit Hilfe des `chmod`-Befehls geändert werden. Der Befehlt hat die Syntax

```
# chmod [ugoa][+-][rwx] Datei
```

Dabei bedeutet:

u: ändere Benutzerrechte
g: ändere Gruppenrechte
o: ändere die Rechte für die anderen
a: Alias für `ogu`

+: füge Rechte hinzu
-: nehme Rechte weg

r: Leserecht
w: Schreibrecht
x: Ausführungsrecht

```
# ls -lad testdir
drwxr-xr-x   2 cbecker  lsiii        512 Mar  7 12:06 testdir
```

```
# chmod go-rx testdir
# ls -lad testdir
drwx------   2 cbecker  lsiii        512 Mar  7 12:29 testdir
```

```
# chmod a+r testdir
# ls -lad testdir
drwxr--r--   2 cbecker  lsiii        512 Mar  7 12:29 testdir
```

Durch den Befehl `cd` kann das Verzeichnis gewechselt werden, der Befehl `pwd` gibt das aktuelle Verzeichnis aus.

```
# pwd
/home/user/cbecker

# cd testdir
# pwd
/home/user/cbecker/testdir

# cd ..
# pwd
/home/user/cbecker
```

Durch den Befehl `rm` können einzelne Dateien oder ganze Verzeichnisse gelöscht werden:

```
# rm Datei
```

Soll ein Verzeichnis gelöscht werden, muss die Option `-r` angegeben werden:

```
# rm -r testdir
```

Der Befehl `touch` gibt einer Datei den aktuellen Zeitstempel und erzeugt die Datei neu, falls sie noch nicht existiert. Die Datei hat dann die Länge 0.

```
# touch leeredatei
# ls -la leeredatei
-rw-r--r--   1 cbecker  lsiii          0 Mar  7 12:41 leeredatei
```

Die Befehle `cp` und `mv` dienen zum Kopieren bzw. Verschieben von Dateien und Verzeichnissen.

```
# touch testdatei
# cp testdatei testdatei1
# ls -la testdatei testdatei1
-rw-r--r--   1 cbecker  lsiii          0 Mar  7 12:42 testdatei
-rw-r--r--   1 cbecker  lsiii          0 Mar  7 12:42 testdatei1

# mv testdatei1 testdatei2
# ls -la testdatei testdatei1 testdatei2
testdatei1: No such file or directory
-rw-r--r--   1 cbecker  lsiii          0 Mar  7 12:42 testdatei
-rw-r--r--   1 cbecker  lsiii          0 Mar  7 12:42 testdatei2
```

Jeder Benutzer hat ein sogenanntes Homeverzeichnis, welches durch die Tilde ~ angesprochen werden kann.

```
# cd ~
```

3 Wichtige Shell-Befehle

`echo` gibt den Inhalt des übergebenen Strings aus:

```
# echo "hallo"
hallo
```

Die Shell bietet die Möglichkeit, Variablen zu definieren mit dem Format NAME=WERT, also z.B.

```
# test="Hallo"
```

Mittels `echo` kann der Wert einer Variablen ausgegeben werden.

```
# echo $test
Hallo
```

Der Befehl `cat` dient zur Ausgabe einer Datei. Dabei sollte es sich um eine Text- und nicht um eine Binärdatei handeln, weil ansonsten auch haufenweise Steuerzeichen mit ausgegeben werden würden, die keinen Sinn machen.

```
# cat /etc/motd
```

Der Befehl `wc` zählt Zeichen, Worte und Zeilen.

```
# wc /etc/motd
      25     140    1638 /etc/motd
```

Die Datei `/etc/motd` enthält 25 Zeilen, 140 Worte und 1638 Zeichen.

4 Zugriffsmasken

Neben der Möglichkeit, Dateinamen direkt anzugeben, existiert auch die Möglichkeit, mit Suchmasken zu arbeiten.

Will man eine beliebige Anzahl an Zeichen offen lassen, wird das `*`-Zeichen verwendet.

```
# ls *.c
```

Ein einzelnes Zeichen lässt sich mit Hilfe des `?` maskieren, dabei ist zu beachten, dass `?` nicht ein leeres Zeichen maskiert, ein `test?.c` maskiert nicht `test.c`.

```
# touch test1.c
# touch test2.c
# ls test?.c
test1.c  test2.c
```

Einzelne Zeichen lassen sich durch `[..]` angeben, dabei wird pro Klammer ein Zeichen maskiert.

```
# ls test[12].c
```

Es lassen sich auch Bereich angeben:

```
# ls test[0-9].c
```

Es lassen sich auch negierte Masken angeben:

```
# ls test[!a].c
test1.c  test2.c
```

Es werden alle Dateinamen ausgegen, die mit `test` anfangen, als nächstes Zeichen kein `a` haben und mit `.c` aufhoeren.

5 & und ;

Normalerweise ist die Shell solange blockiert, wie die Ausführung des eingegeben Befehls dauert, z.B. gibt man in einer Shell den Befehl

```
# acroread
```

ein, kann man keine neuen Kommandos eintippen, solange Acroread laeuft.

Mit Hilfe des Operators `&` kann man jedoch Programme im Hintergrund starten. Die Eingabe von

```
# acroread &
```

führt dazu, dass Acroread gestartet wird, der Kommandozeilenprompt jedoch sofort wieder für neue Kommandos zur Verfügung steht.

Der ; Operator ermöglicht es, mehrere Kommandos in einer Zeile zu schreiben:

```
# ls ; echo "Hallo"
```

6 Ein-/Ausgabeumleitung

Ein sehr mächtiges Werkzeug ist die Möglichkeit, die Ein- und Ausgabe von Befehlen umzuleiten:

```
# ls -l > ausgabe
```

Die Ausgabe, die normalerweise auf dem Bildschirm angezeigt werden würde, landet in der Datei `ausgabe`.

Den Inhalt dieses Datei lässt sich mittels `cat` anzeigen.

```
# cat ausgabe
```

In der Variante `>>` wird die Ausgabedatei nicht überschrieben, sondern die Ausgabe wird an die Datei angehängt.

```
# echo "test" > ausgabe
# cat ausgabe
test
# echo "2.test" >> ausgabe
# cat ausgabe
test
2.test
```

Die Eingabe eines Programmes lässt sich ebenfalls umlenken:

```
# wc < ausgabe
     612    5530   41299
```

Neben der Umleitung in und von Dateien lassen sich die Ein-/Ausgabeströme von Kommandos direkt über eine sogenannte Pipe verbinden, die Ausgabe des ersten Kommandos wird direkt mit der Eingabe des zweiten Kommandos verbunden:

```
# cat ausgabe | wc
     612    5530   41299
```

Neben der normalen Ausgabe existiert noch ein weiterer Ausgabekanal, den Kommandos zur Ausgabe von Fehlermeldungen nutzen, Normalerweise erscheinen beide Kanäle auf dem Bildschirm, bei der Umlenkung wird nur der Standardausgabekanal umgelenkt:

```
# ls -al dateinichtda > ausgabe
dateinichtda: No such file or directory
```

Die Fehlermeldung erscheint auf dem Bildschirm. Um auch den Fehlerkanal umzulenken, ist folgendes zu benutzen

```
# ls -al dateinichtda 2> ausgabe
```

7 Wichtige Shell-Kommandos Teil II

7.1 Dateien suchen

Der Befehl `find` sucht nach Dateien im Verzeichnisbaum. Dem Befehl wird das Startverzeichnis, sowie eine Reihe von Optionen übergeben, die die Suche einschränken können.

```
# find . -name "*.c"
```

Dieser Befehl sucht im aktuellen Verzeichnis nach allen Dateien und Verzeichnissen, die auf `.c` enden.

```
# find . -name "ab*" -type d
```

Dieser Befehl sucht nach allen Verzeichnissen ab dem aktuellen Verzeichnis, die mit `ab` anfangen.

Ferner gibt es die Möglichkeit, auf das Ergebnis der Suchoperation direkt einen Befehl anwenden zu lassen:

```
# find . -name "ab*" -type d -exec ls -lad {} \;
```

Für jede gefundene Datei wird der Befehl, der zwischen `-exec` und `\;` steht aufgerufen, dabei wird `{}` durch das gerade aktuelle Suchergebnis ersetzt.

7.2 Zeichenketten finden

Durch den Befehl grep kann in Datein ach dem Vorkommen eines Strings gesucht werden.

```
# echo "Hallo" > suchdatei
# grep Hallo *
suchdatei:Hallo
```

7.3 Zeichenketten ersetzen

Häufig werden Ersetzungen von Strings benötigt, dies lässt sich mit dem `sed`-Befehl erreichen, in seiner gebräuchlichsten Form lautet der Befehl:

```
sed 's/STRING1/STRING2/g' quelle
```

Dabei werden alle Vorkommen von STRING1 in der Datei quelle durch STRING2 ersetzt und am Bildschirm ausgegeben. Selbstverständlich arbeitet `sed` auch als Teil einer Pipe.

```
# echo "Testlauf" | sed 's/Test/2.Test/g'
2.Testlauf
```

Ein weiteres mächtiges Unix-Werkzeug ist `awk`, welches oft dazu verwendet wird, Teile eines Strings auszugeben:

```
# echo "Teil1 Teil2 Teil3" | awk '{print $3}'
Teil3
```

8 Shell-Skripte

.

Die bislang erläuterten Konstrukte lassen sich nicht nur auf der Kommandozeile benutzen, sondern auch in eigenen Dateien zusammenfassen und gemeinsam ausführen.

In der Datei `helloworld.sh` soll folgender Inhalt stehen:

```
#!/bin/bash

echo "Hello World"
```

Die erste Zeile kennzeichnet die Datei als Bash-Skript.

Bevor die Datei ausgefürht werden kann, muss noch das x-Flag gesetzt werden:

```
# chmod a+x helloworld.sh
```

Das Skript kann jetzt ausgefuehrt werden

```
# ./helloworld.sh
Hello World
```

Shellskripten koennen Parameter übergeben werden, die innerhalb des Skriptes mit `$1, $2` usw. abgefragt werden können.

Inhalt von Datei `helloworld.sh`:

```
#!/bin/bash

echo "Hello World "$1
```

```
# ./helloworld.sh Dortmund
Hello World Dortmund
```