2.2.F Skripte/Variablen – Ergänzungen und Bilder

2.2.F.1 Variablen-Substitution und Default-Werte

Durch kleine Erweiterungen kann man sehr einfach neue Möglichkeiten für die Variablen-Substitution bekommen.

Wenn eine Variable leer ist, kann mit der `:-`Zeichenkette einen Default-Wert einsetzen:

Terminal
```
schueler@debian964:~$ unset BILD            #ist leer
schueler@debian964:~$ echo ${BILD:-default.png}
default.png                 #dann nimmt man eben das Default-Bild
schueler@debian964:~$ echo $BILD
                            #ist immer noch leer
schueler@debian964:~$ BILD="landschaft.png" #jetzt Dateiname eintragen
schueler@debian964:~$ echo ${BILD:-default.png}
landschaft.png              #und nun dieses Bild benutzen
```

Mit der `:+`Zeichenkette kann dieses Verhalten genau umgedreht werden (wozu auch immer das gut sein soll). Mit der der `:=`Zeichenkette kann man (wie mit der `:-`Zeichenkette) einen Default-Wert einsetzen; falls die Variable noch nicht gesetzt war, wird dies auch gleichzeitig der neue Variablen-Inhalt:

Terminal
```
schueler@debian964:~$ unset LIED                    # ist leer
schueler@debian964:~$ echo ${LIED:=default.mp3}
default.mp3                         # man nimmt das Default-Lied
schueler@debian964:~$ echo $LIED
default.mp3                         # Inhalt ist jetzt Default (!)
schueler@debian964:~$ LIED="alle_meine_entchen.mp3" # Dateiname eintr.
schueler@debian964:~$ echo ${LIED:=default.mp3}
alle_meine_entchen.mp3              # und nun dieses Lied benutzen
```

Mit dem `?`-Zeichen wird für den Fall, dass eine Variable leer ist, eine Fehlermeldung ausgegeben und ggf. die aktuelle Shell beendet:

Terminal
```
schueler@debian964:~$ unset BUCH            # ist leer
schueler@debian964:~$ echo ${BUCH?Variable nicht gesetzt.}
bash: BUCH: Variable nicht gesetzt.
schueler@debian964:~$ echo $?
1                     # bei interaktiver Shell Fehlercode statt Ende
```

2.2.F.2 Variablen-Substitution und Text-Ersetzung

Mit dem #-Zeichen kann man den Anfang eines Variablen-Inhalts entfernen:

Terminal
```
schueler@debian964:~$ NAME="Dr. Meier"
schueler@debian964:~$ echo ${NAME#Dr. }
Meier
```

Mit dem `%`-Zeichen kann man das Ende eines Variablen-Inhalts entfernen. Bei einem Dateinamen kann das z. B. die Dateiendung sein:

Terminal
```
schueler@debian964:~$ DATEI="landschaft.jpeg"
schueler@debian964:~$ echo ${DATEI%.jpeg}
landschaft
schueler@debian964:~$ convert $DATEI ${DATEI%.jpeg}.png
```

Mit dem :-Zeichen kann man ein Stück aus einem Variablen-Inhalt herausschneiden:

Terminal
```
schueler@debian964:~$ KLASSE=IFS6A
schueler@debian964:~$ echo ${KLASSE:3}
6A                # Am Anfang 3 Zeichen weglassen
schueler@debian964:~$ echo ${KLASSE:1:2}
FS                # Am Anfang 1 Zeichen weglassen, 2 Zeichen nehmen
```

Mit dem /-Zeichen kann man Teilstrings im Innern ersetzen:

Terminal
```
schueler@debian964:~$ WORT="Zitronensaft"
schueler@debian964:~$ echo ${WORT/Zitrone/Tomate}
Tomatensaft
```

Es wird damit allerdings nur einmal pro Variable substituiert. Möchte man alle Vorkommen substituieren, benutzt man zwei /-Zeichen:

Terminal
```
schueler@debian964:~$ WORT="Aus Zitronen macht man Zitronensaft"
schueler@debian964:~$ echo ${WORT/Zitrone/Tomate}
Aus Tomaten macht man Zitronensaft      # interessant
schueler@debian964:~$ echo ${WORT//Zitrone/Tomate}
Aus Tomaten macht man Tomatensaft       # so sollte es sein
```

Mit dem ^-Zeichen kann man Großschreibung erzwingen:

Terminal
```
schueler@debian964:~$ TATSACHE="heute ist ein schöner Tag"
schueler@debian964:~$ echo ${TATSACHE^[a-n]}
Heute ist ein schöner Tag      # 1 Zeichen ersetzt, falls a-n
schueler@debian964:~$ echo ${TATSACHE^^[a-n]}
HEutE Ist EIN sCHöNEr TAG      # alle Zeichen ersetzt, falls a-n
schueler@debian964:~$ echo ${TATSACHE^^[a-zäöü]}
HEUTE IST EIN SCHÖNER TAG  # alle Zeichen, auch Umlaute
```

Genauso kann man mit dem ,-Zeichen Kleinschreibung erzwingen.

2.2.F.3 Weitere Möglichkeiten für die Variablen-Substitution

Mit dem #-Zeichen *vor* dem Variablennamen bekommt man die Länge des Inhalts:

Terminal
```
schueler@debian964:~$ STRING=Verl
schueler@debian964:~$ echo ${#STRING}
4
```

In der Shell kann man auch Arrays von Strings anlegen. Mit Hilfe eckiger Klammern kommt man an die Inhalte heran:

Terminal
```
schueler@debian964:~$ GETREIDE=("Roggen" "Gerste" "Hafer") #Nr.0,1,2
schueler@debian964:~$ echo "${GETREIDE[1]}"
Gerste
schueler@debian964:~$ GETREIDE[4]=Weizen          # Nr. 4
schueler@debian964:~$ echo "${GETREIDE[4]}"
Weizen
schueler@debian964:~$ echo "${GETREIDE[@]}"
Roggen Gerste Hafer Weizen       # alle Inhalte
schueler@debian964:~$ echo "${!GETREIDE[@]}"
0 1 2 4                          # alle Indizes mit Inhalten
```

Mit einer ähnlichen Konstruktion kann man sich auch die Namen aller aktuell gesetzten Variablen ausgeben lassen, deren Namen mit einem bestimmten Teilstring beginnen (die Namen, nicht die Inhalte!):

Terminal
```
schueler@debian964:~$ KLASSE="IFS2B"
schueler@debian964:~$ KLIMA="Nasskalt"
schueler@debian964:~$ echo "${!KL@}"
KLASSE KLIMA
```

2.2.F.4 Gleitkomma-Arithmetik mit `bc` und `dc`

Die arithmetische Substitution kann nur mit ganzen Zahlen umgehen. Möchte man auf Shell-Ebene auch mit gebrochenen Zahlen (Kommazahlen) umgehen, empfehlen sich die Programme `bc` und `dc`.

Bei `bc` handelt es sich um einen normalen Taschenrechner:

Terminal
```
schueler@debian964:~$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017
Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
scale=40
(1+2)*3^4
243
100/97
1.0309278350515463917525773195876288659793
quit
```

Mit dem Befehl `scale=40` wird festgelegt, dass die Berechnung auf 40 Dezimalstellen genau erfolgt.

Bei `dc` handelt es sich um einen Taschenrechner mit der so genannten *Umgekehrten Polnischen Notation* (UPN). Dabei werden Werte nacheinander auf einen Stack gelegt und anschließend die jeweils obersten Werte mit einer Rechenoperation verknüpft:

Terminal
```
schueler@debian964:~$ dc
40 k
1 2 + 3 4 ^ * p
243
100 97 / p
1.0309278350515463917525773195876288659793
q
```

Hier wird mit `40 k` die Anzahl der Dezimalstellen festgelegt. Mit `p` erfolgt die Ausgabe des letzten Wertes auf dem Stapel, mit `q` das Ende.

Und nun ein Beispiel, wie man `bc` benutzen kann, um in Z das Ergebnis von X/Y zu erhalten:

Terminal
```
schueler@debian964:~$ X=1
schueler@debian964:~$ Y=7
schueler@debian964:~$ bc <<< "scale=20
> $X/$Y" > tmp.txt
schueler@debian964:~$ read Z < tmp.txt
schueler@debian964:~$ echo "$Z"
.14285714285714285714
```

Hier war es nach dem `scale`-Befehl nötig, einen Zeilenumbruch auszuführen.

Zum Schluss das gleiche Beispiel mit `dc`:

Terminal

```
schueler@debian964:~$ X=1
schueler@debian964:~$ Y=7
schueler@debian964:~$ dc <<< "20 k $X $Y / p" > tmp.txt
schueler@debian964:~$ read Z < tmp.txt
schueler@debian964:~$ echo "$Z"
.14285714285714285714
```