1.5 Anwendung/Umgebungsvariablen

1.5.1 Was sind Variablen?

Beim Installieren von Programmen wird man manchmal nach den Einstellungen für die `PATH`-Variable gefragt. Was hat es damit auf sich?

Bei Windows und Linux und allen anderen Systemen gibt es dafür das gleiche Konzept; nur die Befehle sind unterschiedlich. Das Konzept nennt sich *Variablen*. Variablen sind Platzhalter, die einen Inhalt haben, vergleichbar mit dem Platzhalter x in mathematischen Aufgaben. Wenn man von der Variablen x spricht, meint man:

- den Namen x, unter der diese Variable geführt ist. Der Name ist frei wählbar.
- den Inhalt von x, z.B. x=3.
- den Speicherort, an dem man x findet. Um den Speicherort muss man sich hier nicht kümmern.

1.5.2 Variablen auf der Linux- und auf der Windows-Konsole

Zuerst kann man eine Variable auf einen bestimmten Wert setzen, zunächst unter Linux:

Terminal
```
schueler@debian964:~$ export x=3
```

Und so geht es unter Windows:

Terminal
```
C:\> set x=3
```

Dann kann man die Variable benutzen. Das bedeutet, man gibt ihren Namen an und erhält dafür den Inhalt. Mit dem Befehl `echo` kann man den Inhalt dann ausgeben, zunächst wieder unter Linux:

Terminal
```
schueler@debian964:~$ echo "$x"
3
```

Und unter Windows:

Terminal
```
C:\> echo %x%
3
```

Man kann die Variable auch per Tastatur auf einen Wert setzen:

Terminal
```
schueler@debian964:~$ echo -n "Bitte x eingeben:"; read x
Bitte x eingeben: 4
```

Und unter Windows:

Terminal
```
C:\> set /P x="Bitte x eingeben:"
Bitte x eingeben: 4
```

Will man die Inhalte aller Variablen ausgeben, so benutzt man den Befehl `set`

Terminal
```
schueler@debian964:~$ set
...
x=4
```

1.5.3 Spezielle Variablen

Einige Variablen sind mit speziellen Bedeutungen vorbelegt.

Dazu gehört beispielsweise die Variable `PATH` für die Suchpfadliste. `PATH` enthält eine Liste von Pfadnamen von Verzeichnissen. Sobald man einen Befehl eingibt, z.B. `cal`, dann sucht die Shell in allen diesen Verzeichnissen nach dem Programm `cal`. Schließlich findet die Shell das Programm – es liegt in `/usr/bin` – und führt es aus. Man braucht also nicht einzutippen:

Terminal
```
schueler@debian964:~$ /usr/bin/cal
```

Man kann sich den Inhalt der Suchpfadliste wie jede anderen Variableninhalt ausgeben lassen:

Terminal
```
schueler@debian964:~$ echo $PATH
C:\> echo %PATH%
```

In Linux sind die Pfade durch Doppelpunkt getrennt, in Windows durch Semikolon. Man kann die Suchpfadliste auch ergänzen:

Terminal
```
schueler@debian964:~$ PATH=$PATH:neuerPfad
C:\> set PATH=%PATH%;neuerPfad
```

Eine andere Variable liefert den *Fehlerwert* des letzten Programms, also die Information, ob das Programm erfolgreich war. Der Wert 0 bedeutet, dass das Programm erfolgreich war. Jeder andere Wert weist auf einen Fehler hin. Zunächst unter Linux:

Terminal
```
schueler@debian964:~$ cal 10000
cal: year 10000 not in range 1..9999
schueler@debian964:~$ echo $?
64
```

Und wieder unter Windows:

Terminal
```
C:\> format t:
Das angegebene Laufwerk ist nicht vorhanden.
C:\> echo %ERRORLEVEL%
1
```

1.5.4 Variable vorbesetzen durch `.bashrc`

Bei der Linux-Shell ist es vorgesehen, dass man schon zum Start der Konsole einige Variable vorbesetzt. Dazu ist die versteckte Datei `.bashrc` vorgesehen, die beim Start der Shell einmal eingelesen wird. Wenn man sie ergänzt, stehen die neuen Inhalte ab dem nächsten Start zur Verfügung.

Wenn man häufig eigene kleine Programme schreibt, ist es sinnvoll, mit Hilfe eines Editors die folgende Zeile am Ende der Datei anzufügen:

```
1 PATH="$PATH:. "
```

Mit dieser Zeile fügt man das jeweils aktuelle Verzeichnis (durch einen Punkt abgekürzt) an das Ende der Suchpfadliste an. Wenn man nun ein Programm schreibt, z.B. mit dem Namen `a.out`, so muss man dann zum Starten des Programms nicht mehr eingeben:

Terminal
```
schueler@debian964:~$ ./a.out
```

Stattdessen reicht aus:

Terminal
```
schueler@debian964:~$ a.out
```