

1.7.F Anwendung/Expansion und Substitution – Ergänzungen und Bilder

1.7.F.1 Reihenfolge der Substitutionen durch die Shell

Die Substitutionen und Expansionen durch die Shell werden in einer genau festgelegten Reihenfolge ausgeführt (Merkwort: **KTASW**):

1. **K**lammern-Expansion von Listen
2. **T**ilden-Substitution
3. **A**lle Ersetzungen mit Dollar-Zeichen: Variablen-Substitution, Kommando-Substitution, Arithmetische Substitution
4. **S**plitten der Eingabezeile in Worte (*word splitting*)
5. **W**ildcard-Expansion von Pfadnamen

1.7.F.2 History-Substitution

Mit dem Begriff *History* meint man bei der Shell eine Liste der bisher durch diesen Benutzer eingegebenen Befehlszeilen. Sie wird gespeichert in der Datei `~/.bash_history`. Mit dem Befehl `history` kann man die Liste komplett ausgeben. Durch die Pfeiltasten kann man an der Eingabeaufforderung durch die Liste hindurchscrollen. Mit Hilfe von `[Alt] - [.]` kann man an der aktuellen Position das letzte Wort aus einer beliebigen Zeile der History einfügen.

Außerdem gibt es die Möglichkeit, mit Hilfe der History-Substitution gezielt einzelne Befehlszeilen wiederzubekommen:

```
Terminal
schueler@debian964:~$ ls # führe ls aus
..
schueler@debian964:~$ !! # führe letzten Befehl noch einmal aus
..
schueler@debian964:~$ echo !! # zeige letzten Befehl noch einmal
echo ls
ls
schueler@debian964:~$ echo !-2 # zeige vorletzten Befehl noch einmal
echo ls
ls
schueler@debian964:~$ echo !500 # zeige Befehl Nr. 500
echo cal
cal
schueler@debian964:~$ echo !l # zeige letzten Bef., der mit l beginnt
echo ls
ls
schueler@debian964:~$ echo !?l # zeige letzten Bef., der ein l enthält
echo echo ls
echo ls
```

Bei der History-Substitution wird aus Gründen der Übersichtlichkeit die neue Zeile (also die mit der Ersetzung) vor dem Ausführen angezeigt.

Mit dem Setzen der Variablen `$HISTTIMEFORMAT` kann man übrigens bestimmen, dass zu jedem Eintrag der History ein Zeitstempel gespeichert wird. Er wird dann mit dem in der Variablen festgelegten Format (wie in der `strftime()`-Funktion in C) ausgegeben:

```
Terminal
schueler@debian964:~$ HISTTIMEFORMAT="%d.%m.%Y-%H:%M:%S"
schueler@debian964:~$ history
```

```
...
2039 06.10.2017-12:58:54 history
```

1.7.F.3 Word Splitting

Hier ein Beispiel zum sogenannten *word splitting*: In der Variablen ORTE soll eine Liste von Ortsnamen liegen. Für jeden dieser Ortsnamen soll ein eigenes Verzeichnis angelegt werden.

```
Terminal
schueler@debian964:~$ set -x # alle Shell-Ersetzungen sichtbar machen
schueler@debian964:~$ ORTE="Detmold Lemgo Lage"
+ ORTE='Detmold Lemgo Lage'
schueler@debian964:~$ mkdir $ORTE
+ mkdir Detmold Lemgo Lage
schueler@debian964:~$ ls -l
+ ls --color=auto -l
Detmold
Lage
Lemgo
```

Zuerst wurde die Befehlszeile in zwei Worte zerteilt. Dieses Trennen fällt noch nicht unter den Begriff *word splitting*, sondern passiert bei jedem Befehl. Die entstandenen Worte sind hier in Hochkomma gesetzt:

```
'mkdir' '$ORTE'
```

Dann wurde die Variablen-Substitution angewandt:

```
'mkdir' 'Detmold Lemgo Lage'
```

Abschließend geschah das *word splitting*; die durch die Variablen-Substitution neu gebildeten Worte wurden nach den in der Variablen IFS aufgezählten Trennzeichen durchsucht und aufgeteilt:

```
'mkdir' 'Detmold' 'Lemgo' 'Lage'
```

Dieser Befehl wurde ausgeführt und ergab das gewünschte Ergebnis, nämlich drei Verzeichnisse.

Wenn man Ortsnamen mit Leerzeichen hat, kann man die Variable IFS zu Hilfe nehmen:

```
Terminal
schueler@debian964:~$ ORTE="Herford+Bad Salzuflen"
+ ORTE='Herford+Bad Salzuflen'
schueler@debian964:~$ IFS="+"
+ IFS=+
schueler@debian964:~$ mkdir $ORTE
+ mkdir Herford 'Bad Salzuflen'
schueler@debian964:~$ ls -l
+ ls --color=auto -l
Bad Salzuflen
Herford
schueler@debian964:~$ unset IFS # zur Vorsicht
+ unset IFS
```

Hier hat das *word splitting* genau an der Stelle getrennt, an der das gewünscht war.

Und hier ein Beispiel, in dem das *word splitting* ein unerwünschtes Ergebnis schafft. Es sollen für ein Auto (angegeben sind Marken- und Typnamen) zwei Verzeichnisse angelegt werden, nämlich eines für Reparaturen und eines für die Versicherung.

```
Terminal
schueler@debian964:~$ VERZEICHNIS="VW Golf"
+ VERZEICHNIS='VW Golf'
schueler@debian964:~$ mkdir $VERZEICHNIS.{Reparaturen,Versicherung}
+ mkdir VW Golf.Reparaturen VW Golf.Versicherung
mkdir: Verzeichnis „VW“ kann nicht angelegt werden: Datei existiert bereits
schueler@debian964:~$ ls -l
+ ls --color=auto -l
Golf.Reparaturen
Golf.Versicherung
VW
```

Was ist hier passiert? Offenbar hat hier das *word splitting* zugeschlagen. Die Reihenfolge war so: Zuerst hat die Shell die originale Befehlszeile in zwei Worte (hier in Hochkomma gesetzt):

```
Terminal
'mkdir' '$VERZEICHNIS.{Reparaturen,Versicherung}'
```

Anschließend wird das zweite Wort wegen seiner geschweiften Klammern in zwei Worte aufgeteilt:

```
Terminal
'mkdir' '$VERZEICHNIS.Reparaturen' '$VERZEICHNIS.Versicherung'
```

Nun findet die Variablen-Substitution statt:

```
Terminal
'mkdir' 'VW Golf.Reparaturen' 'VW Golf.Versicherung'
```

Der in diesem Fall letzte Schritt ist dann das *word splitting*:

```
Terminal
'mkdir' 'VW' 'Golf.Reparaturen' 'VW' 'Golf.Versicherung'
```

Richtig funktioniert es dann, wenn man die Variable in Anführungszeichen setzt:

```
Terminal
schueler@debian964:~$ mkdir "$VERZEICHNIS".{Reparaturen,Versicherung}
+ mkdir 'VW Golf.Reparaturen' 'VW Golf.Versicherung'
schueler@debian964:~$ ls -l
+ ls --color=auto -l
VW Golf.Reparaturen
VW Golf.Versicherung
```

1.7.F.4 Eingebaute Shell- bzw. Umgebungsvariablen bei Linux und Windows

Einige Shell- bzw. Umgebungsvariablen sind sowohl bei Linux als auch bei Windows vorhanden. Tabelle 1 zeigt eine Gegenüberstellung gleichartiger Variablen.

Linux	Windows	Bedeutung
\$HOSTNAME	%COMPUTERNAME%	Rechnername
\$HOME	%HOMEPATH%	Pfadname des persönlichen Verzeichnisses
\$PATH	%PATH%	Suchpfadliste für ausführbare Dateien
\$USER	%USERNAME%	Name des aktuellen Benutzers
\$LOGNAME	%USERNAME%	Loginname des aktuellen Benutzers
\$BASH	%CMDCMDLINE%	Pfadname des Kommandozeilen-Interpreters
\$SHELL	%COMSPEC%	Pfadname des Kommandozeilen-Interpreters
\$PWD	%CD%	Pfadname des aktuellen Verzeichnisses
\$PS1	%PROMPT%	Prompt (Eingabeaufforderung)
\$OSTYPE	%OS%	Name des Betriebssystems
\$?	%ERRORLEVEL%	Rückgabewert des letzten Befehls
\$RANDOM	%RANDOM%	Zufallszahl

Tabelle 1: Gleichartige Variablen bei Linux und Windows