



SV3: Linux Server

Die Shell Teil 2

Agenda

Die Bash

- Metazeichen, Dateinamenexpansion
- Reguläre Ausdrücke, Quoting
- Umgebungsvariablen
- Shell-Aufruf, -Optionen

Metazeichen zur Expansion von Dateinamen

Um beim Suchen nach einem bestimmten Muster (Wort) nicht alle Dateinamen des aktuellen Directories eintippen zu müssen, können Sie ein weiteres Werkzeug der Shell einsetzen: ***Das Ersetzen von Dateinamen durch Metazeichen.***

Was sind aber Metazeichen?

Die Shell ersetzt Dateinamen, in denen Metazeichen vorkommen, durch alle Dateinamen des betreffenden Directories, auf die die Vorgabe passt.

Metazeichen zur Expansion von Dateinamen

Welche Metazeichen gibt es?

- `root# find [abc]`

Bei der Umleitung von Ein- und Ausgabe (`>`, `>>`, `<`, `2>`) erfolgt keine Dateinamenexpansion durch Metazeichen.

- Bei Ausgabe- oder Fehlerumleitung würden die Metazeichen mit als Dateinamen angelegt werden, was zu Fehlern führen würde

Metazeichen zur Expansion von Dateinamen

Metazeichen	Ersetzung
*	Das Sternchen steht für eine beliebige Zeichenfolge (oder kein Zeichen)
?	Das Fragezeichen steht für ein einzelnes beliebiges Zeichen (aber nicht für kein Zeichen)
[...]	Die Klammer mit wird ersetzt durch ein in der Klammer angegebenes Zeichen: [abc] wird ersetzt durch a, b oder c
[!...]	Die Klammer mit Ausrufezeichen wird ersetzt durch ein beliebiges Zeichen m das nicht in der Klammer angegeben ist: [!abc] wird ersetzt durch ein beliebiges Zeichen, aber nicht a, b oder c
\	Dieser Schrägstrich hebt den Ersetzungsmechanismus für das nachfolgende Metazeichen auf (Fluchtsymbol): \ Das Fragezeichen wird nicht durch ein beliebiges Zeichen ersetzt, sondern als Fragezeichen übernommen.

Metazeichen zur Expansion von Dateinamen

Beispiele zur Dateinamenexpansion durch Metazeichen

- Um festzustellen, in welcher Datei innerhalb des aktuellen Directories das Wort »mail« vorkommt, können Sie folgendes Kommando eingeben:

```
sb@sb-VirtualBox: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
sb@sb-VirtualBox:~$ grep "mail" *  
grep: Bilder: Ist ein Verzeichnis  
grep: Dokumente: Ist ein Verzeichnis  
grep: Downloads: Ist ein Verzeichnis  
grep: Musik: Ist ein Verzeichnis  
neu: Wie bei dem Kommando mail muss das  
grep: Öffentlich: Ist ein Verzeichnis  
grep: Schreibtisch: Ist ein Verzeichnis  
grep: Videos: Ist ein Verzeichnis
```

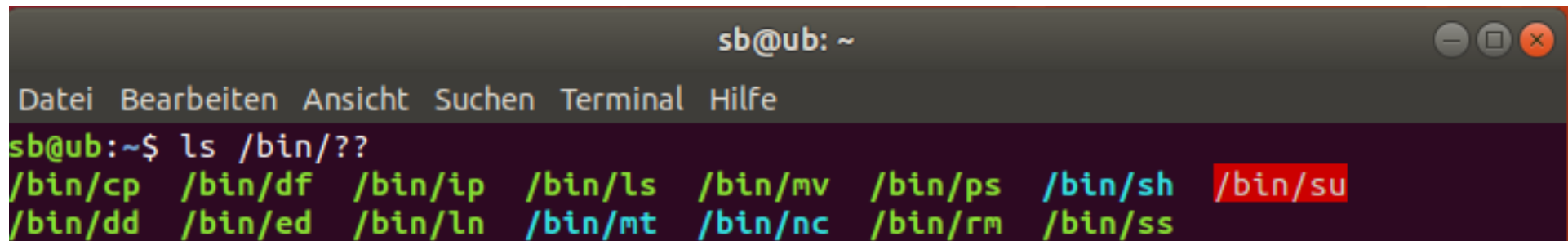
Die Shell ersetzt das *
Durch alle Dateinamen, die sie in dem
aktuellen Directory findet:
Datum befehle inhalt neu ueb1 projektA

↑ gefundene Zeile mit dem gesuchten Begriff
↑ Name der Datei, in der der gesuchte Begriff vorkommt

Metazeichen zur Expansion von Dateinamen

Beispiele zur Dateinamenexpansion durch Metazeichen

- Oder möchten Sie gerne wissen, welche zweistelligen Kommandos es unter dem Directory /bin gibt bzw. wie viele es sind?

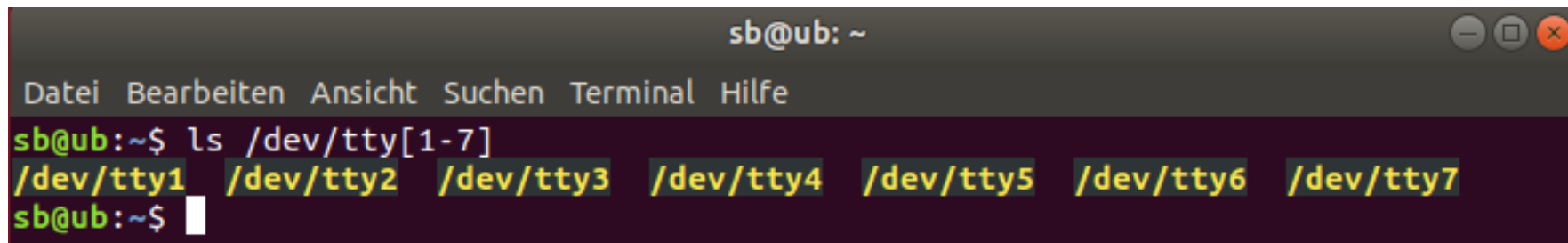
A terminal window titled 'sb@ub: ~' with standard window controls. The menu bar shows 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Terminal', and 'Hilfe'. The command prompt shows 'sb@ub:~\$ ls /bin/??'. The output is a two-line list of files: the first line contains /bin/cp, /bin/df, /bin/ip, /bin/ls, /bin/mv, /bin/ps, /bin/sh, and /bin/su (highlighted in red); the second line contains /bin/dd, /bin/ed, /bin/lm, /bin/mt, /bin/nc, /bin/rm, and /bin/ss.

```
sb@ub: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
sb@ub:~$ ls /bin/??
/bin/cp  /bin/df  /bin/ip  /bin/ls  /bin/mv  /bin/ps  /bin/sh  /bin/su
/bin/dd  /bin/ed  /bin/lm  /bin/mt  /bin/nc  /bin/rm  /bin/ss
```

Metazeichen zur Expansion von Dateinamen

Beispiel zur Dateinamenexpansion durch Metazeichen

- Gibt es die Terminalbezeichnungen tty1 – tty7 unter dem Directory /dev?
- Mit der eckigen Klammer lässt sich dies leicht abfragen.
- Beachten Sie aber bitte, dass jeweils nur ein Zeichen ersetzt werden kann.

A terminal window titled 'sb@ub: ~' with a menu bar containing 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Terminal', and 'Hilfe'. The terminal shows the command 'ls /dev/tty[1-7]' being executed. The output is a single line listing seven files: '/dev/tty1', '/dev/tty2', '/dev/tty3', '/dev/tty4', '/dev/tty5', '/dev/tty6', and '/dev/tty7'. The prompt 'sb@ub:~\$' is visible at the bottom with a cursor.

```
sb@ub: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
sb@ub:~$ ls /dev/tty[1-7]
/dev/tty1 /dev/tty2 /dev/tty3 /dev/tty4 /dev/tty5 /dev/tty6 /dev/tty7
sb@ub:~$
```

(weitere Beispiele im Handout)

Metazeichen zur Expansion von Dateinamen

Sie können sich mit Hilfe von Metazeichen viel Schreibarbeit ersparen.

- Wenn Sie sich den Inhalt von der Datei Datum ansehen wollen, genügt es, nur `cat D*` aufzurufen, vorausgesetzt, es gibt nur eine Datei, deren Namen mit »D« anfängt.
- Sollten mehrere Dateien mit »D« beginnen, würden alle der Reihe nach angezeigt werden.

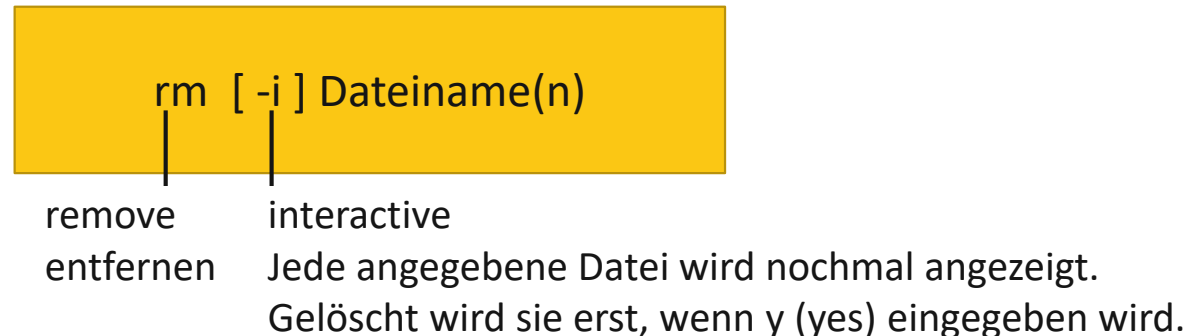
Was passiert wohl, wenn Sie das Datum an das Ende der Datei »neu« setzen möchten, und Sie geben bei der Umleitung der Ausgabe ebenfalls nur `D*` an?

- Bei der aktuellen Version in Ubuntu 16.04 funktioniert. Aber in der Vergangenheit hat es nicht funktioniert, also ich würde es vermeiden zu nutzen!

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

- Wie kann die fehlerhaft angelegte Datei wieder gelöscht werden?
- Hierzu gibt es das Kommando zum Löschen von Dateien `rm` (remove). Es ist als das gefährlichste Kommando unter Linux/Unix bekannt. Mit diesem Kommando werden Dateien unwiederbringlich gelöscht!



Linux verzeiht keine Fehler! Also überlegen sie sich gut wenn sie mit Root und rm arbeiten!

Metazeichen zur Expansion von Dateinamen

Was passiert, wenn `rm D*` eingegeben wird, um die in unserem Beispiel irrtümlich angelegte Datei »D*« zu löschen?

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

Um in unserem Beispiel die Datei `D*` zu löschen, können wir auch mit Hilfe des Fluchtsymbols (Aufhebung von Metazeichen) `»\«` gezielt nur die Datei `»D*«` angeben

- Ohne Fluchtzeichen (escape sequenz) und mit Fluchtzeichen:

```
sb@ub:~/test$ ls
befehle 'D*' Datum inhalt neu projektA ueb1
sb@ub:~/test$ rm -i D*
rm: Normale leere Datei 'D*' entfernen? n
rm: Normale leere Datei 'Datum' entfernen? n
sb@ub:~/test$ rm -i D\*
rm: Normale leere Datei 'D*' entfernen? n
sb@ub:~/test$ rm -i "D*"
rm: Normale leere Datei 'D*' entfernen? n
sb@ub:~/test$ rm -i 'D*'
rm: Normale leere Datei 'D*' entfernen? n
sb@ub:~/test$
```

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

Wenn Sie z.B. eine Kommandozeile eingeben und der Bildschirm ist nicht groß genug, um alle Angaben in einer Zeile zu schreiben, so können Sie mit dem Backslash die Bedeutung der Eingabetaste aufheben und auf der nächsten Zeile weiterschreiben.

```
sb@ub:~/test$ cat inhalt > inh\  
> alt2  
sb@ub:~/test$ cat inhalt2  
befehle  
D*  
Datum  
inhalt  
inhalt2  
neu  
projektA  
ueb1  
sb@ub:~/test$
```

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

- Um bestimmte Sonderzeichen (*, ? [], Leerzeichen usw.) zu übergeben, werden sie in Anführungszeichen gesetzt (d.h., sie sollen nicht von der Shell beim Kommandoaufruf durch Dateinamen ersetzt werden)



\" \"

Übergabe von bestimmten Sonderzeichen
Keine Ersetzung der Metazeichen: * ? []

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

In dem nachfolgenden Beispiel wurde aus dem Inhalt aller angegebenen Dateien nach den Wörtern Dont und Drink gesucht.

```
sb@ub:~/test$ cat ueb1 | grep "Dont drink"  
Dont drink and be root!!!  
sb@ub:~/test$
```

Beide Wörter wurden gefunden, da das Anführungszeichen das Leerzeichen auch als solches übergeben.

Metazeichen zur Expansion von Dateinamen

Anwendungsbeispiel

Lässt man das Anführungszeichen weg, wird das Zweite Wort „drink“ nicht mehr zum Muster zugeordnet, sondern als Option interpretiert:

```
sb@ub:~/test$ cat ueb1 | grep Dont drink
grep: drink: Datei oder Verzeichnis nicht gefunden
sb@ub:~/test$
```


Metazeichen zur Expansion von Dateinamen

Weitere Beispiele:

Kommando	Funktion
?	Genau ein beliebiges Zeichen
*	Beliebig viele (auch null) beliebige Zeichen (aber keine <code>.*</code> -Dateien!)
**	Alle Dateien und Verzeichnisse, auch aus allen Unterverzeichnissen (ab bash 4.0 mit shopt <code>–s globstar</code>)
[abc]	Eines der angegebenen Zeichen
[a-f]	Ein Zeichen aus dem angegebenen Zeichen
[!abc]	Keines der angegebenen Zeichen
[^abc]	Wie oben
~	Abkürzung für das Heimatverzeichnis

Metazeichen zur Expansion von Dateinamen

Weitere Beispiele:

Kommando	Funktion
.	Aktuelles Verzeichnis
..	Übergeordnetes Verzeichnis
ab{1, 2, 3}	Liefert ab1 ab2 ab3
a{1..4}	Liefert a1 a2 a3 a4
\$(3*4)	Arithmetische Berechnungen
`kommando`	Ersetzt das Kommando durch sein Ergebnis.
\$(kommando)	Wie oben, alternative Schreibweise
Kommando „Zeichen“	Verhindert die Auswertung aller Sonderzeichen außer \$
Kommando ‚Zeichen‘	Wie oben aber ohne restriktiver (keine Variablensubstitution)

Reguläre Ausdrücke

reguläre Ausdrücke (engl. regular expressions)

Diese Ausdrücke – es gibt übrigens ganze Bücher zum Thema – dienen in der Shell zum Filtern von Zeichenketten (Strings) aus einer Eingabe, etwa einer Textdatei.

- Wie Sie sehen, wurden tatsächlich nur die Zeilen ausgegeben, in denen das Zeichen »a« vorkam. Dies können Sie mit jedem Zeichen und sogar mit ganzen Strings.

```
$ cat Standorte
Augsburg
Bremen
Friedrichshafen
Aschersleben
Bernburg
Berlin
Halle
Essen
Furtwangen
Kehlen
Krumbach
Osnabrueck
Kempten

// Nun werden alle Orte, die ein 'a' enthalten gefiltert:

$ grep a Standorte
Friedrichshafen
Halle
Furtwangen
Krumbach
Osnabrueck
```

Reguläre Ausdrücke

Allerdings können mithilfe dieser regulären Ausdrücke nicht nur explizit angegebene Strings, wie etwa »hafen«, gefiltert werden.

- So können Sie angeben, dass »hafen« am Zeilenende oder -anfang vorkommen kann, dass das zweite Zeichen ein »a«, aber auch ein »x« sein kann
- dass das letzte Zeichen entweder klein- oder großgeschrieben werden darf und so weiter.
- Sollen beispielsweise alle Zeilen, die auf »n« oder »g« enden, ausgegeben werden, kann der reguläre Ausdruck `[ng]$` verwendet werden:

```
$ grep "[ng]$" Standorte
Augsburg
Bremen
Friedrichshafen
Aschersleben
Bernburg
Berlin
Essen
Furtwangen
Kehlen
Kempten
```

Reguläre Ausdrücke

Lesen würde man den Ausdruck so:

- Das letzte Zeichen der Zeile (\$) kann entweder ein »n« oder ein »g« sein ([ng]). Reguläre Ausdrücke können sich aus mehreren solcher Muster zusammensetzen.
- Sie können beispielsweise das Zeichen, das vor dem letzten Zeichen einer Zeile steht, auch noch festlegen und so weiter

Weitere Beispiele und Möglichkeiten der Kombination im Handout!

Shell-Variablen

Was sind Shell-Variablen?

Variablen, wie der Name bereits aussagt, können variabel sein, d.h. veränderliche Werte beinhalten.

- Unter der Shell können Sie Variablen, die durch einen Namen identifiziert angesprochen werden, einen Wert zuweisen.
- Diesen Wert können Sie zu einem späteren Zeitpunkt abfragen.

Name=Wert	Definition einer Variablen durch Zuweisung eines Wertes
Eingabe ohne Leerzeichen!	
\$Name	Ersetzung der Variablen „Name“ durch den zugewiesenen Wert
= - Zuweisung von Variablen unter der Shell	
Beispiel: k=kursteilnehmer	

Was sind Shell-Variablen?

Mit Variablen können Sie sich eine Reihe von Abkürzungen schaffen. Um den Wert einer Variablen zu bekommen, wird als Kennung ein Dollarzeichen »\$« vor den Namen gesetzt.

- Geben Sie »echo hallo \$k« ein, erkennt die Shell, dass es sich nicht um den Buchstaben »k« handelt, sondern um die Variable »k« und ersetzt »\$k« durch den zugewiesenen Wert »hallo Kursteilnehmer«

```
sb@ub:~$ k=kursteilnehmer
sb@ub:~$ echo "Hallo $k"
Hallo kursteilnehmer
sb@ub:~$
```

Variablen sind temporär und bleiben so lange erhalten, bis die Shell beendet wird (Beenden der Sitzung – neues Login).

Shell-Variablen

Verwenden Sie den Wert der Variablen, um z.B. neue Dateinamen zu erstellen, wobei die Ersetzung nur ein Teil des Namens ist, dann wird der Name der Variablen durch geschweifte Klammern abgegrenzt:

`${Name}`

Möchten Sie z.B. Dateien erstellen mit dem jeweiligen Login-Namen desjenigen, der die Prozedur aufgerufen hat, und zusätzlich noch eine Nummer anfügen, dann geben Sie ein: `> ${LOGNAME}1` bzw. `touch ${LOGNAME}1`. Ähnlich funktioniert auch die Variable `$USER`.

Shell-Variablen

Variable können durch die Zuweisung eines neuen Wertes verändert werden.

Shell-Variable sind nur für die aktuelle Shell gültig, sollen sie auch für Unterprogramme (Sohnprozesse) gelten, so sind sie zu exportieren.

```
export Name der Variablen
```

Shell-Variablen

Vordefinierte Shell-Variable

Um den PATH zu erweitern, sollten Sie sich erst vergewissern, welche Directories in dem für Sie vorgegebenen Suchpfad eingetragen sind.

- Das Kommando `echo $PATH` gibt Ihnen hierüber Auskunft:

```
sb@ub:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
sb@ub:~$
```

Vordefinierte Shell-Variable

Die Variable **PATH** wird von der Shell verwendet, um Kommandos zu suchen. Sollen die von Ihnen erstellten Kommandos Vorrang vor den Linux/Unix-Kommandos erhalten, dann müssen Sie die Variable PATH so belegen, dass das vorrangig zu behandelnde Directory zu Beginn steht, z.B.:

- `PATH=~ /Befehle:$PATH`; export PATH oder
- `export PATH=~ /Befehle:$PATH`

Vordefinierte Shell-Variable

Mit dem Kommando `set` werden die in dem aktuellen Programm gesetzten Variablen

- angezeigt, mit `env` die Variablen, die exportiert wurden.

set

env

set - Kommando, um definierte Variablen der aktuellen Shell anzuzeigen

env – Kommando um definierte und exportierte Variablen anzuzeigen

Vordefinierte Shell-Variable

- Die Variablen gelten immer nur für die Dauer des Programms.
- Die vordefinierten Variablen werden beim Starten Ihrer Shell zugewiesen (beim login).
- Sobald Sie sich abmelden bzw. ein neues Terminal mit eigener Shell öffnen, verlieren sie ihre Gültigkeit. Die Erweiterung des PATH müssten Sie also jeweils neu vornehmen.
- Um Ihre Shell-Umgebung automatisch mit den Variablen zu besetzen, die für Sie Gültigkeit haben sollen, können Sie die Datei .profile in Ihrem Home-Directory ändern.

Shell-Variablen

Vordefinierte Shell-Variable

- Fügen Sie z.B. mit dem Editor **nano** folgenden Inhalt in die Datei `bash.bashrc` ein:

```
echo "Guten Morgen, heute ist der $(date)"  
echo "Es arbeiten bereits folgende Kollegen:"  
who
```

Shell-Variablen

Vordefinierte Shell-Variable

- Nach dem Sie nun ein neues Terminal öffnen folgendes:

```
sb@ub: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
Guten Morgen, heute ist der Do 16. Apr 13:45:16 CEST 2020  
Es arbeiten bereits folgende Kollegen:  
sb      :1      2020-04-16 13:16 (:1)  
hans    pts/1    2020-04-16 13:44 (127.0.0.1)  
tux     pts/4     2020-04-16 13:45 (127.0.0.1)  
sb@ub:~$
```

Ersetzung durch das Ergebnis eines Kommandos

Mit Hilfe der Sonderzeichen `$(cmd)` oder mit der veralteten Schreibweise (Accent Grave) können Sie das Ergebnis von Kommandos als Parameter übergeben (Kommandosubstitution).

```
sb@ub:~$ echo "Guten Morgen, heute ist der $(date)"
Guten Morgen, heute ist der Do 16. Apr 13:46:48 CEST 2020
sb@ub:~$
```


Shell-Variablen

Ersetzung durch das Ergebnis eines Kommandos

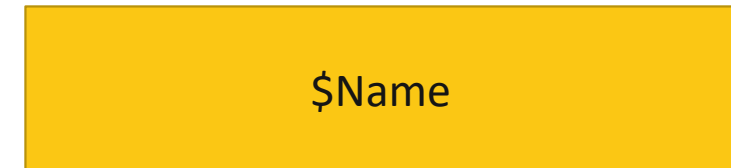
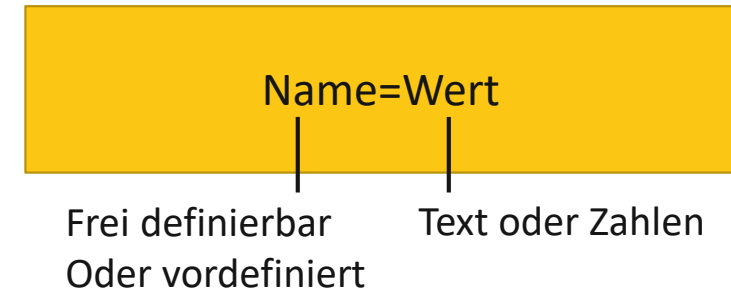
Beispiel Accent Grave:

```
sb@ub:~$ echo "Guten Morgen, heute ist der `date`"  
Guten Morgen, heute ist der Do 16. Apr 13:47:56 CEST 2020  
sb@ub:~$
```

Shell-Variablen

Fassen wir das Wesentliche über die Shell-Variablen zusammen:

- Shell-Variablen werden gebildet durch eine Wertzuweisung
- Kommen in einem Wert Leerzeichen vor, so muss der gesamte Ausdruck in Anführungszeichen gesetzt werden, z.b. PS1=„Alles klar“ . Zwischen dem Namen, dem Gleichheitszeichen und dem Wert dürfen keine Leerzeichen sein.
- \$Name wird ersetzt durch den Wert, der der Variablen Name zugewiesen wurde



Shell-Aufruf, -Optionen

Aufruf einer Shell

Prinzipiell kann eine Shell im laufenden Betrieb aufgerufen werden. Allerdings gilt der Aufruf nur bis zum nächsten Neustart.

- Aufruf der "sh" Shell z.B .

- root# sh
- root# bash
- root# zsh

```
sb@ub:~$ sh
$ pwd
/home/sb
$ exit
sb@ub:~$
```

Shell-Aufruf, -Optionen

Permanenter Wechsel einer Shell

Um auch bei einem Neustart eine andere Shell zur Verfügung zu haben, kann man das Kommando **chsh** nutzen. Die gewählte Shell wird dann bei jeder neuen Anmeldung ausgeführt.

```
root@ub:/home/sb# cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
root@ub:/home/sb#
root@ub:/home/sb# chsh
Login-Shell für root wird geändert.
Geben Sie einen neuen Wert an oder drücken Sie ENTER für den Standardwert
      Login-Shell [/bin/bash]: /bin/sh
root@ub:/home/sb#
```

Bash-Aufruf Optionen

Die Shell, in unserem Fall die "Bash", kann in verschiedenen Modi gestartet werden. Einen haben Sie schon kennen gelernt, den Start als "Login-Shell". Es gibt weitere, hier die wichtigsten.

-c	Übergabe eines Kommandos/Skripts
-i	Interaktive Shell
-l	Login Shell
-v	Zeigt die ausgeführten Befehle so wie sie von der "Bash" entgegen genommen werden
-r	Restricted Mode
-x	Zeigt die Befehle wie sie ausgeführt werden

Shell-Aufruf, -Optionen

Übergabe eines Kommandos/Skripts

- Sie können der "Bash" auch gleich einen Befehl übergeben.
- Die "Bash" wird gestartet und führt nur den Befehl aus.
- Es wird nur für den ausgeführten Befehl in die "Bash" verzweigt.
- Nach der Ausführung befinden Sie sich wieder in der aufrufenden Shell.

```
sb@ub:~$ bash -c 'ls -ld Videos'
drwxr-xr-x 2 sb sb 4096 Apr 15 16:22 Videos
sb@ub:~$
```

Interaktive Shell

Die "interaktive Shell" verbindet sich mit einem Terminal.

- Ist Sie mit einem "Terminal/Konsole", in unserem Fall einer "virtuellen Konsole" verbunden, kann sie Eingaben entgegen nehmen.

Shell-Aufruf, -Optionen

Restricted Mode

Im "restriktiven Modus" wird die "Bash" eingeschränkt.

- Es ist zum Beispiel nicht möglich einen Ordner zu wechseln.

Shell-Aufruf, -Optionen

shopt

bietet weitere Möglichkeiten der Einstellungen für die "Bash-Optionen".

```
sb@ub:~$ shopt --help
shopt: shopt [-pqsu] [-o] [Optionsname ...]
        Setzt oder löscht Shell Optionen.

        Ändert die in 'Optionsname' genannten Shell Optionen. Ohne
        Argumente wird eine Liste der Shell Optionen und deren Status
        ausgegeben.

        Optionen:
        -o      Beschränkt die Optionsnamen auf die, welche mit
                  'set -o' definiert werden müssen.
        -p      Gibt alle Shelloptionen und deren Status aus.
        -q      Unterdrückt Ausgaben.
        -s      Setzt jede Option in 'Optionsname.'
        -u      Deaktiviert jede Option in 'Optionsname'.
```



**VIELEN DANK
FÜR IHRE
AUFMERKSAMKEIT!**