1.103.1 Command Line

www.lpi.org

Shell-Variablen

Variablen, welche der Shell (bash, Bourne Again Shell) zur Verfügung stehen und diese Steuern können.

Anzeigen einer einzelnen Variablen
Beispiel:

```
root@abc:~# echo $PS1
\u@\h:\w\$
root@abc:~#
```

$PS1 bestimmt das Auss“ehen des Promts
\u = User, \h=Hostname, \w bzw. \W=Working-Dir,
\$=$ für user und # für root

Setzen einer Variablen:

```
root@abc:~# PS1=“\u@\H:\W\\$“
```

Achtung: Variable hier ohne $!

Anzeige aller Shell-Variablen mit *set*
Entfernen einer Shell-Variablen mit *unset*
Anzeigen des aktuellen Pfades mit *pwd*

Umgebungs-Variablen

Wird ein Programm von der Shell aus aufgerufen, so stehen diesem die Shell-Variablen **nicht** zur Verfügung.
=> Variable muss exportiert werden (Verwendung einer Kopie!)

```
root@abc:~# export dummy
root@abc:~# export test=123
```

Mit *env* kann man alle exportierten Variablen betrachten, oder ein Programm aufrufen, wobei die Variablen gesetzt werden können

Wichtige Umgebungsvariable

$PATH In den dort angegebenen Pfaden wird nach Befehlen gesucht (Trennung der Pfade durch :)
Befehle, welche nicht in einem dieser Verzeichnisse liegen, müssen mit dem Pfad verwendet werden.

Viele Variablen werden in */etc/profile* bzw. in *~/.profile* gesetzt.

Command History

Die Befehle die in der Vergangenheit ausgeführt wurden werden gespeichert.

- Mit ↑ (bzw. Ctrl-p) kann man die alten Befehle wieder erreichen
- Mit Ctrl-r kann man nach alten Befehlen suchen

Beim Abmelden merkt sich die bash in der Datei ~/.bash_history die History (-> Befehl: *history*)

Vervollständigung

- Wurde nur ein Teil eines Befehls (oder eines Dateinamens) eingegeben, dieser jedoch eindeutig, so vervollständigt TAB den Befehl (bzw. den Dateiname)
- Ist der eingegebene Teil mehrdeutig, wird bei der Eingabe eines weiteren TAB alle Möglichkeiten angezeigt

Befehlsaufruf

- Wird ein Befehl aufgerufen wird die bash **nicht beendet**
- Wird ein Befehl mit exec aufgerufen, wird die bash **beendet**

Kommandoausführung

Mehrere Befehle können auch in einer Zeile eingegeben werden

- Kommando1 ; Kommando2
  - Kommando2 wir nach Beendigung von Kommando1 ausgeführt
- Kommando1 & Kommando2
  - Beide Kommandos werden parallel ausgeführt (Kommando2 im Vordergrund, Kommando1 im Hintergrund)
- Kommando1 && Kommando2
  - Kommando2 wird nur ausgeführt, wenn Kommando1 fehlerfrei ausgeführt wurde (Fehlercode 0)
- Kommando1 || Kommando2
  - Kommando2 wird nur ausgeführt, wenn in Kommando1 ein Fehler auftrat (Fehlercode ungleich 0)

Kommandosubstitution

Das Ergebnis (genauer die Ausgabe) eines Befehls, kann als Teil eines anderen Befehls verwendet werden.

Beispiel:

```
root@berkely:~# pwd
/root
root@abc:~# echo Wir sind im Verzeichnis `pwd`
Wir sind im Verzeichnis /root
root@berkely:~# echo Wir sind im Verzeichnis $(pwd)
Wir sind im Verzeichnis /root
root@berkely:~#
```

pwd gibt als Befehl das aktuelle Verzeichnis an. `pwd` bzw. $(pwd) wird zuerst von der bash ausgeführt und das Ergebnis dann in den echo-Befehl eingesetzt.

` -> Backtick

Aufgaben

- Ändern Sie das Prompt auf
  root@abc.xyz.de Time: 16:15 root$
  (Hinweis: Suchen Sie in der Manualseite der *bash* nach PS1)
- Welche Shell-Variablen sind gesetzt?
- Welche Umgebungs-Variablen sind gesetzt?
- Was ist der Unterschied zwischen einer Shell-Variablen und einer Umgebungsvariablen?
- Welche Pfade sind gesetzt?
- Speichern Sie in der Umgebungsvariablen *$LINE* die Anzahl der Zeilen der Datei *~/.bash_history* ab.
  (Hinweis: Verwenden Sie den Befehl *wc*)