KLEINE BASH-TRICKS

Tübix 2026

Tobias Heine | @toheine@social.tchncs.de | 04.07.2026

VORBEMERKUNGEN

MOTIVATION

- Vorab: Vortrag von CLT 2025 (in Teilen überarbeitet)
- Eigene Umgebung für sich selbst optimieren
- Möglichkeiten der Bash nutzen
- Nahe am Standard bleiben

VERWENDETES SYSTEM

- Ausgangspunkt ist ein Debian 13
- Andere Distributionen verhalten sich weitgehend ähnlich
- Empfohlene Tools aus den regulären Debian-Paketquellen

ZIEL

- Prompt für die eigenen Bedürfnisse anpassen
- History optimieren
- Ein paar Kniffe bei der Eingabe
- Eine Auswahl an Zusatztools

RELEVANTE KONFIGURATIONSDATEIEN

Für diesen Vortrag relevant:

- `/etc/bash.bashrc` → Systemweite Initialisierung (per interactive Shell)
- `~/.bashrc` → indiviudelle Initialisierung (per interactive Shell)
- `~/.bash_history` → History-Datei eines Users
- `/etc/skel/.bashrc` → Vorlage der `~/.bashrc` für neue User
- `~/.bash_aliases` → Aliase eines Users

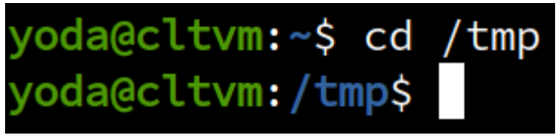
PROMPT ANPASSUNGEN

STANDARD-PROMPT

Aufbau eines Bash-Prompts üblicherweise:

```
username@<hostname>:<Pfad>#|$
```

Demo:



Hilfreich um zu sehen:

- welcher User / welche Userin
- welches Gerät
- an welchem Ort im Dateibaum

WEITERER AUSBAU DENKBAR:

Anpassungen je nach System:

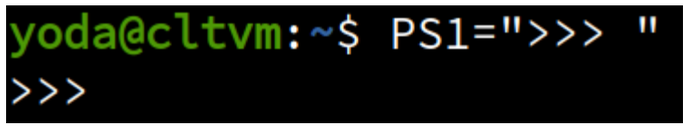
- Uhrzeit
- Farbliche Hervorhebung eigener Arbeitsgeräte
- Laborumgebungen von produktiven Umgebungen besser unterscheiden
- Git-Branch darstellen
- Zeilenumbruch, um lange Pfade vom eigentlichen Kommando zu trennen
- u.v.m.

EINFACHE (FARBLOSE) UMSETZUNG I

Anpassung der Variablen `PS1` ändert den Prompt sofort. Beispiel:

```
PS1=">>> "
```

Demo:



EINFACHE (FARBLOSE) UMSETZUNG II

Verwendung von **Special Characters** bei der Zuweisung bieten Dynamik:

- `\u` → Username
- `\h` → Hostname
- `\w` → Aktuelles Arbeitsverzeichnis
- `\$` → Indikator für einen eingeschränkten User

Beispiel:

```
PS1="\u@\h:\w\$ "
```

Demo:

```
>>> PS1="\u@\h:\w\$ "
yoda@cltvm:~$
```

EINFACHE (FARBLOSE) UMSETZUNG III

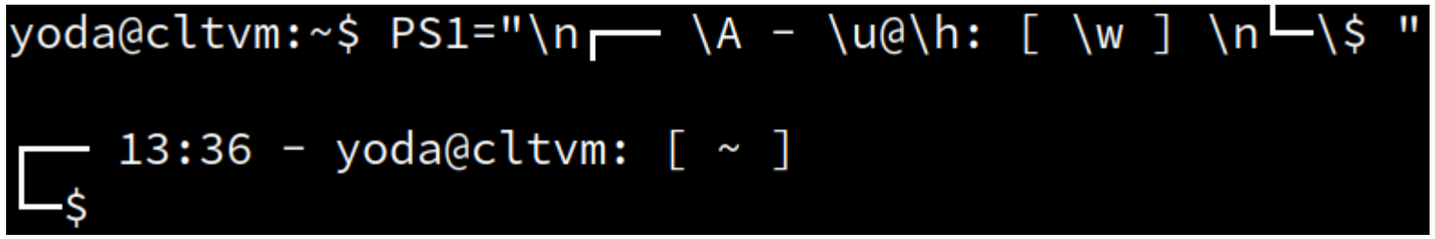
Eigene Optimierung mit Uhrzeit und Umbruch. Zusätzliche **Special Characters**:

- `\n` → Zeilenumbruch
- `\A` → Uhrzeit im 24-Stundenformat

Beispiel:

```
PS1="\n┌── \A - \u@\h: [ \w ] \n└─\$ "
```

Demo:



FARBE REIN BRINGEN I

Bestandteile des Prompts mit **Escape-Sequenzen** farblich hervorheben. Auszug:

- `\[\e[33m\]` → braun
- `\[\e[37m\]` → weiss
- `\[\e[32m\]` → grün
- `\[\e[00m\]` → defaults

Beispiel:

```
PS1="\n┌── \[\e[33m\]\A \[\e[m\]- \[\e[37m\]\u@\h\[\e[m\]: \[\e[33m\][ \w ]\[\e[m\] \n└─\$ "
```

FARBE REIN BRINGEN II

Bestandteile des Prompts mit **Escape-Sequenzen** farblich hervorheben. Auszug:

- `\[\e[33m\]` → braun
- `\[\e[37m\]` → weiss
- `\[\e[32m\]` → grün
- `\[\e[00m\]` → defaults

Demo:

```
┌── 13:36 - yoda@cltvm: [ ~ ]
└─$ PS1="\n┌── \[\e[33m\]\A \[\e[m\]- \[\e[37m\]\u@\h\[\e[m\]: \[\e[33m\][ \w ]\[\e[m\] \n└─\$ "

┌── 13:52 - yoda@cltvm: [ ~ ]
└─$
```

PROMPT HINTERLEGEN

- Sobald der gewünschte Prompt den eigenen Wünschen entspricht
- ... diese Variablenzuweisung in der `~/.bashrc` unten anfügen
- Prompt wird nun bei Login bzw. bei öffnen einer neuen `Bash` wie gewünscht dargestellt
- Die Änderung kann mit `source ~/.bashrc` direkt durchgeführt werden

GIT-BRANCH IM PROMPT DARSTELLEN I

Hierzu in `~/.bashrc` folgende drei Änderungen durchführen:

1. Funktion hinterlegen:

```
function check_branch {
    if git status > /dev/null 2>&1; then
        branch="<<< $(printf '\uE0A0') $(git branch --show current) >>>"
    else
        branch=""
    fi
}
```

2. Variable `PS1` anpassen (Zur besseren Lesbarkeit auf zwei Zuweisungen aufgesplittet):

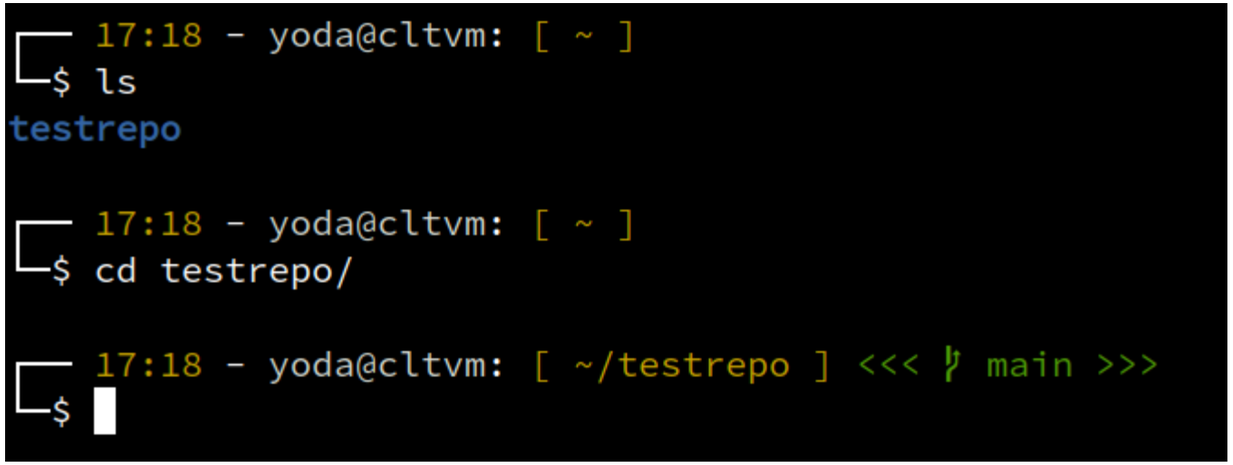
```
PS1="\n┌── \[\e[33m\]\A \[\e[m\]- \[\e[37m\]\u@\h\[\e[m\]: \[\e[33m\][ \w ]\[\e[m\] \[\e[32m\]"
PS1+="\$branch \[\e[m\] \n└─\$ "
```

GIT-BRANCH IM PROMPT DARSTELLEN II

3. Prompt-Command anpassen:

```
PROMPT_COMMAND="check_branch"
```

Demo:



ÜBER MEINE INHALTE HINAUS

- Infos zu **Special Characters**: `man bash` im Abschnitt **PROMPTING**
- Infos zu Farbsequenzen: `man console_codes`
- zahlreiche Bash-Generatoren im Netz. Beispiel: https://bash-prompt-generator.org/
- Fertige Lösungen gibt es ebenfalls. Beispiel: https://ohmyposh.dev/

HISTORY

HISTORY AUFRUFEN

```
┌─ 14:17 - yoda@cltvm: [ ~ ]
└─$ history
    1  PS1=">>> "
    2  PS1="\u@\h:\w\$ "
    3  PS1="\n┌─ \A - \u@\h: [ \w ] \n└─\$ "
    4  PS1="\n┌─ \[\e[33m\]\A \[\e[m\]- \[\e[37m\]\u@\h\[\e[m\]: \[\e[33m\][ \w ]\[\e[m\] \n└─\$ "
    5  man console_codes
    6  man bash
    7  history
    8  vim .bash_history
    9  exit
   10  history
```

- Ausgabe der letzten Befehle mit einer ID
- Liegt im RAM
- Beim Logout werden Befehle in die Datei `~/.bash_history` geschrieben

HISTORY ANPASSEN (BEISPIEL) I

In der `.bashrc` folgende Variablen nach eigenen Vorlieben anpassen:

```
HISTCONTROL=ignoreboth           # entspricht ignoredups + ignorespace
HISTSIZE=30000                   # Default: 1000
HISTFILESIZE=90000               # Default: 2000
HISTTIMEFORMAT="(%d.%m.%y)  "
HISTIGNORE="?:??:???:bash:clear:exit:man*:*--help"
PROMPT_COMMAND="history -a;check_branch"
```

HISTORY ANPASSEN (BEISPIEL) II

Ergebnis - Aufbau der `~/.bash_history`:

```
#1742304872
echo "Hallo"
#1742304880
whoami
```

Ausgabe des Kommandos `history`:

```
   19  (18.03.25)  echo "Hallo"
   20  (18.03.25)  whoami
   21  (18.03.25)  ls -l
   22  (18.03.25)  history
```

Ein Filtern nach Datum ist nun einfach möglich z. B. via

```
history | grep 18.03.25
```

ALIASE

DIE IDEE

- Häufig verwendete, lange Kommandos abkürzen
- Je nach Distribution sind schon einige Aliase gesetzt. Typische Beispiele:

```
alias ll='ls -lh'
alias la='ls -A'
alias ls='ls --color=auto'
```

- Aliase anzeigen: `alias`
- Einen bestimmten Alias anzeigen `alias ls`
- Alias entfernen: `unalias ls`

EIGENE ALIASE

- Eigene Bedürfnisse je nach Bedarf (z. B. Auszug meiner eigenen Aliase)

```
alias sb='source ~/.bashrc'
alias lsd='ls -l --group-directories-first'
alias pid='echo Meine Personalnummer lautet HAPPY_TUEBIX_26'
alias kto='echo Meine IBAN lautet DE..PFFT..26'
```

- Das wird schnell viel (zu viel)

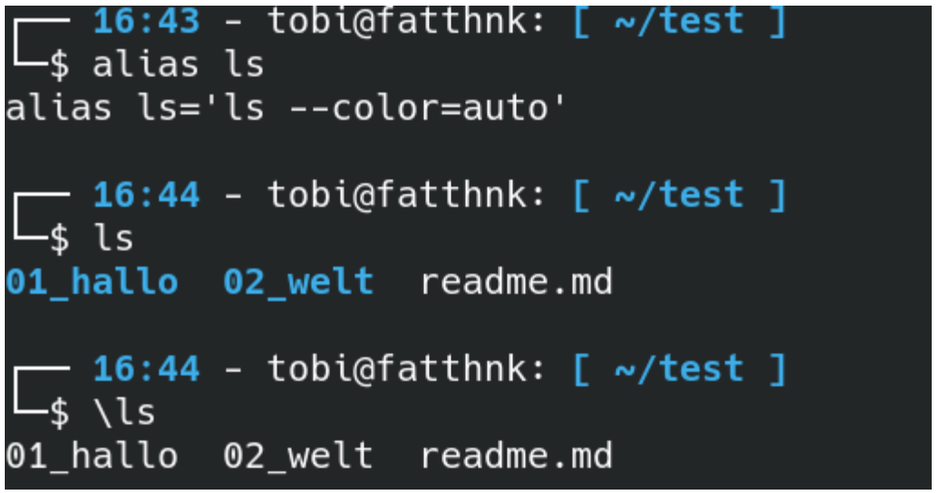
DEN ÜBERBLICK BEHALTEN

- Gesetzte Aliase sind immer nur in einer Shell gültig
- Sollen diese immer zu Verfügung stehen, können die Aliase in der `~/.bashrc` definiert sein
- Noch besser (IMHO): Aliase in `~/.bash_aliases` auslagern. Dazu in `~/.bashrc` folgende Zeilen hinterlegen:

```
# Include aliases
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

WENN EIN KOMMANDO AUCH EIN ALIAS IST

- Ein Kommando kann auch gleichzeitig ein Alias sein
- Wenn beides vorhanden ist, wird der Alias aufgerufen
- Mit einem führenden `\` vor dem Kommando, wird das Programm und nicht der Alias aufgerufen. Beispiel:



KLEINE KONSOLENKNIFFE

AUTOVERVOLLSTÄNDIGUNG

- Die Wundervolle [**TAB**]-Taste:
  - Autovervollständigung
  - Mehrdeutige Möglichkeiten ausgeben für Dateien ...
  - ... Kommandos ...
  - ... Aliase ...
- Letztes Argument hinter neuen Befehl: [**STRG**] + [**ALT**] + **.**

SCHNELLER ZUGRIFF AUF DIE HISTORY

- Bestimmter Befehl aus der History aufrufen → `!<id>`
- Den letzten Befehl mit `!-1`, vorletzen Befehl mit `!-2`, ... aufrufen
- Den letzten Befehl mit `!!` aufrufen
- Den letzten Befehl mit voran gestellten sudo aufrufen → `sudo !!`
- Rekursive Suche → [**STRG**] + [**R**] → `<zeichenkette>`

WIE WAR DAS NOCHMAL? APROPOS

- Ich weiß den Befehl nicht: `apropos <Suchbegriff>`
- Suche mit `apropos -a` (and) eingrenzen.

Beispiel: Ich suche einen Befehl um sicher auf ein Remote-Ziel zu kopieren:

```
apropos secure          # liefert 84 Treffer
apropos copy            # liefert 19 Treffer
apropos -a secure copy  # liefert 2 Treffer
```

WAS ICH SONST NOCH SO BRAUCHE

- Mit `cd -` ins vorgehende Arbeitsverzeichnis wechseln (`oldpwd`)
- Die letzten 10 Zeilen einer Log-Datei anzeigen → `tail <dateiname>`
- Die Ausgabe live verfolgen → `tail -f <dateiname>`
- Geht auch mit Systemd-Journal → `journalctl -fu <unitname>`
- Programm periodisch ausführen → `watch <kommando>`
- Programm mit Pipe periodisch ausführen → `watch "<kommando> | <kommando>"`

FAZIT

(M)eine total subjektive Einschätzung der Inhalte hier

- Investition in die Anpassung der eigenen Umgebung lohnt sich
  - Schneller orientieren
  - effizienter Arbeiten
  - Missgeschicke vermeiden
- IMHO mehr als nur eine Spielerei
- Es gibt so viel mehr zu entdecken

Hinweis: Gegenüber dem Vortrag der CLT 2025, sind die Tools rausgefallen. Eine Liste von ergänzenden Werkzeugen habe ich hier bereitgestellt: https://toheine.net/toolbox

Schau, schau ... dieses Video: Die Drei Klammern [] {} () und die BASH