

SelfLinux-0.13.0



## Bunte Shells - Benutzung der Shellfarben



Autor: Nico Golde ([nico@ngolde.de](mailto:nico@ngolde.de))  
Formatierung: Matthias Hagedorn ([matthias.hagedorn@selflinux.org](mailto:matthias.hagedorn@selflinux.org))  
Lizenz: GPL

## Inhaltsverzeichnis

### 1 Allgemeines

- 1.1 Konfiguration der Shell
- 1.2 Escape-Sequenzen
- 1.3 Farben der Shell benutzen
- 1.4 Eigenschaften der Schrift

### 2 Eine andere Form der Anwendung

- 2.1 Beispiele in C
- 2.2 Beispiel für die init-Scripte

### 3 Feedback

## 1 Allgemeines

Mit der [Einführung in die Bourne Again Shell \(bash\)](#) kommt praktisch jeder Linux-Benutzer irgendwann in Berührung. Auf den ersten Blick sieht diese sehr langweilig aus, doch man hat viele Möglichkeiten, seine Shell individuell zu gestalten. Farbliche Hervorhebungen der Eingabeaufforderung (des Shell-Promptes) machen die Shell zum Einen individuell und zum Anderen gut lesbar.

Dieses Dokument bezieht sich auf die bash-Shell. Die Escape-Sequenzen (► [siehe unten](#)) können sich von Terminal zu Terminal unterscheiden, der Autor bezieht sich in diesem Text auf ein ANSI Terminal. Ein ANSI Terminal ist ein Terminal, das zu dem Zeichensatzstandard ANSI kompatibel ist. Es gibt noch andere Zeichensätze, der bekannteste ist ASCII.

Dieses Dokument kann nur einen kleinen Überblick über die Funktionalität der Shell in Bezug auf ihre Eigenschaften geben. Viele Themen werden nur angerissen oder überhaupt nicht erwähnt. Grundlegende Informationen zur **bash** sind in einem eigenen Kapitel [Einführung in die Bourne Again Shell \(bash\)](#) zu finden.

Der Autor  [Nico Golde](#) freut sich natürlich über Kommentare und Anregungen zu diesem Text.

### 1.1 Konfiguration der Shell

Bei der **bash** ist die zentrale Konfigurationsdatei, die auch bei der Konfiguration der Shellfarbe zum Einsatz kommt, die Datei `~/.bashrc` oder die globale bash-Konfigurationsdatei `/etc/bashrc`. Mittels der `PS1` Variable in der `bashrc` werden die Einstellungen für das Aussehen der Shell verändert.

Normalerweise hat dieser Eintrag die folgende Form:

```
PS1="\s-\v\$ "
```

Dabei steht `\s` für den Namen und `\v` für die Version der Shell. Am Ende des Promptes wird dann ein `$` gesetzt. Das Ganze sieht in der Shell so aus (ohne Anführungszeichen):

```
"bash-2.05b$ "
```

Da dies ein wenig langweilig ist, kann man stattdessen folgenden Eintrag verwenden, den die meisten Linux-Distributionen als Standardeinstellung haben:

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

Hier steht `\u` für den Benutzernamen, `\h` für den Hostnamen und `\w` für das aktuelle Verzeichnis. Als Prompt bekommt man jetzt (wieder ohne Anführungszeichen):

```
"user@myhost /home $ "
```

Dies ist der normale Shell-Prompt, den die meisten Linux-Benutzer gewöhnt sein werden.

### 1.2 Escape-Sequenzen

Um diesem Prompt mittels verschiedener Farben eine persönliche Note zu verleihen, verwendet man Escape-Sequenzen (engl: flüchten). Eine Escape-Sequenz ist ein Steuerzeichen, das die Shell veranlasst, etwas Bestimmtes auszuführen, sozusagen aus ihrem normalen Modus zu flüchten. Escape-Sequenzen werden auch Steuerzeichen genannt und sind keine normalen druckbaren Zeichen. Vielmehr dienen sie dazu, Bildschirmausgaben eines Programms zu manipulieren (steuern). Eine Escape-Sequenz wird normalerweise mittels `^[` eingeleitet. Diese Schreibweise ist etwas gewöhnungsbedürftig. `\033` hat denselben Effekt.

In der Shell kann man, statt die Sequenz zu tippen, allerdings auch `Strg + V ESC` drücken.

### 1.3 Farben der Shell benutzen

Die Farben der Shell werden zunächst an einem Beispiel-Prompt erklärt.

```
PS1="\[\033[0;32;40m\u@\h:\w\$ \]"
```

Dies stellt den kompletten Prompt in grün dar. `\033` leitet die Escape Sequenz ein, `[` eröffnet die Farbangabe. Die anschließende `0` gibt an, dass eine Normaldarstellung benutzt wird. Welche anderen Möglichkeiten man an dieser Stelle hat, wird später erwähnt. Die gesamte Sequenz ist in `\[` und `\]` eingeschlossen, damit sie nicht in die Ausgabe mit hinein kommen und Platz auf der Shell wegnehmen.

Als nächstes wird die Vordergrundfarbe gewählt (in diesem Fall `32`, das entspricht Grün). Die Hintergrundfarbe `40` steht für die Farbe Schwarz. Möchte man in unserem Beispiel nicht, dass die Schrift nach dem Prompt auch grün ist, so fügt man an den Schluss die Escape Sequenz `\033[0m` an. Dies ist die Voreinstellung für die Shellfarbe. Sowohl für den Vordergrund als auch für den Hintergrund stehen 8 Farben zur Verfügung.

Auswahl: schwarz, rot, grün, gelb, blau, magenta, cyan und weiß. Die Zahlen dafür sind: 30, 31, 32, 33, 34, 35, 36, 37.

Das Setzen der Hintergrundfarbe verläuft genauso, allerdings statt 3 mit 4. Also 40, 41, 42, 43, 44, 45, 46, 47.

Beispiel:

```
PS1="\[\033[0;37;44m\u@\033[0;32;43m\h:\033[0;33;41m\w$\033[0m\]"
```

Am besten testet man diese Einstellungen per `export PS1="string"` in der Kommandozeile und kann die Einstellungen später in die `bashrc` eintragen. Der aktuelle Prompt des Autors sieht z.B. so aus:

```
PS1="\[\033[1;34;40m[\033[1;31;40m\u@\h:\w\033[1;34;40m]\033[1;37;40m $ \033[0;37;0m\]"
```

### 1.4 Eigenschaften der Schrift

Wie oben erwähnt ist die `0` direkt nach der ersten Escape Sequenz die Voreinstellung für die Schrift des Shell-Promptes. Für die Schrifteigenschaft sind die folgenden Werte sinnvoll: **0**, **1**, **22**, **4**, **24**, **5**, **25**, **7**, **27**. Sie bedeuten: **Standard**, **dick**, **nicht dick**, **unterstrichen**, **nicht unterstrichen**, **blinkend**, **nicht blinkend**, **invers**, **nicht invers**.

Mit dem folgenden kleinen Shell-Script kann man sich die Farbkombinationen ansehen:

```
#!/bin/sh
#####
# Nico Golde <nico@ngolde.de> Homepage: http://www.ngolde.de
# Letzte Änderung: Mon Feb 16 16:24:41 CET 2004
#####

for attr in 0 1 4 5 7 ; do
  echo "-----"
  printf "ESC[%s;Vordergrund;Hintergrundm - \n" $attr
  for fore in 30 31 32 33 34 35 36 37; do
    for back in 40 41 42 43 44 45 46 47; do
      printf '\033[%s;%s;%sm %02s;%02s ' $attr $fore $back $fore $back
    done
  done
  printf '\n'
done
printf '\033[0m'
done
```

## 2 Eine andere Form der Anwendung

Die Fähigkeit der Farbsetzung in der Shell macht nicht nur Sinn, wenn man einen schöneren Shell-Prompt haben möchte, sondern kann auch in der Programmierung eines Programmes für die Konsole sinnvoll sein.

So müsste man bei jeder Benutzung von Farben auf Bibliotheken wie `slang` oder `ncurses` zurückgreifen, was die Größe der Binärdatei beachtlich steigen ließe. `Ncurses` hat allerdings den Vorteil, dass es mehr oder weniger unabhängig vom Terminal-Typ ist.

### 2.1 Beispiele in C

Ein Hello World mit grüner Schrift:

Hello World
<pre>#include &lt;stdio.h&gt; int main(void){     const char *const green = "\033[0;40;32m";     const char *const normal = "\033[0m";     printf("%sHello World%s\n", green, normal);     return 0; }</pre>

Eine weitere nützliche Escape Sequenz ist `printf("\033[2J");` Sie hat denselben Effekt wie `system(clear);` allerdings kann man damit auf die Headerdatei `unistd.h` verzichten.

Mit `printf("\033[1K");` lässt sich eine Zeile löschen.

### 2.2 Beispiel für die init-Scripte

Möchte man nicht, dass beim Starten der init-Scripte in `/etc/init.d` nur ein `!` beim erfolgreichen Starten des Prozesses angezeigt wird, sondern statt dessen eine farbliche, besser lesbare Ausgabe, so ist dies auch mittels einer Escape-Sequenz lösbar.

Hier ein Auszug aus einem `Cron` init-Script:

Cron init-Script
<pre>#!/bin/sh # Start/stop the cron daemon. test -f /usr/sbin/cron    exit 0  case "\$1" in     start) echo -n "Starting periodic command scheduler: cron"            start-stop-daemon --start --quiet --exec /usr/sbin/cron            echo "."     ;; </pre>

Beim erfolgreichen Starten von `Cron` wird dann ein Punkt ausgegeben. Farblich könnte man das auch durch `[Ok]` gestalten, indem man den echo String ändert. Z.B. so:

## Cron init-Script 2

```
#!/bin/sh
# Start/stop the cron daemon.
test -f /usr/sbin/cron || exit 0
case "$1" in
start) echo -n "Starting periodic command scheduler: cron"
      start-stop-daemon --start --quiet --exec /usr/sbin/cron
echo "\[ \033[1;34;40m[ \033[1;32;40mOk \033[1;34;40m]\033[0m\"
; ;
```

Möchte man diese Einstellungen allerdings für alle init-Scripte vornehmen, ist das sehr zeitaufwändig, es sei denn, man benutzt als Escape Sequenz `\033`, da `Strg + v ESC` nicht als Zeichen interpretiert wird.

Es kann sich allerdings ein weiteres Problem ergeben: Bei einer System-Aktualisierung, wie z.B. der Einspielung von Sicherheitsaktualisierungen können veränderte init-Scripte und andere Konfigurationsdateien durch Distributionsstandards überschrieben werden. Dadurch könnte das farbige [ OK ] schneller verschwinden, als es eingerichtet wurde.

### 3 Feedback

Feedback, Kritik, Bugs etc. bitte per Mail an ✉️ [nico@ngolde.de](mailto:nico@ngolde.de) schicken.  
Viel Spaß...