

Terminals, uxterm und screen

Ingo Blechschmidt
<iblech@web.de>

LUGA

7. September 2005

Inhalt

- 1 Terminals
 - Geschichte
 - Interna
 - Terminal-Emulatoren
- 2 (u)xterm
 - Geschichte
 - Features
 - Konfiguration
- 3 screen
 - Features
 - Funktionsweise
 - screen-nodestatus

Terminals

- Terminal: Einheit von Aus- und Eingabegerät
- „Dumb Terminals“ (frühe 1970er):
Revolution des Umgangs mit Computern durch einfache Ein- und Ausgabe
- „Intelligent Terminals“ (späte 1970er):
Interpretation von Escape-Sequenzen (Cursor-Positionierung, Schreiben auf beliebigen Positionen, Farbauswahl, etc.)
- Heute: Weitgehender Ersatz von echten Hardware-Terminals durch Terminal-Emulatoren (xterm, rxvt, DOS mit geladenem `ansi.sys`, ...)

Probleme

Inkompatibilitäten zwischen den einzelnen Terminal-Emulatoren:

- Unterschiedliche Interpretation von Escape-Sequenzen
- Unterschiedliche Unterstützung von Farben
- Unterschiedliche Handhabung von nicht-ASCII
- Unterschiedliche Handhabung von Sondertasten (Funktionstasten, Cursorbewegungstasten, Modifiers (Shift, Ctrl, Num-Lock, etc.))
- Unterschiedliches Verhalten in Grenzfällen (Cursor in der letzten Spalte der letzten Zeile, etc.)
- ...

Abhilfe: \$TERM, termcap, terminfo

- Setzen der Umgebungsvariablen TERM durch jeden Terminal-Emulator
- Datenbank über alle Terminal-Typen:
 - „Unterstützt du Farben?“
 - „Welche Escape-Sequenz muss ich dir schicken, um den Cursor auf den Zeilenanfang zu positionieren?“
 - „Wenn \$USER F1 drückt – welche Sequenz schickst du mir dann?“
 - „Wenn ich was in die untere rechte Ecke schreibe – tickst du dann aus?“
- Siehe auch: `/usr/share/terminfo/?/*`

Cool. Problem beseitigt, nicht?

- Leider nein.
- Problem: Aktualität der installierten terminfo-Datenbank
- Nutzen von `shinynewterm`
- SSHen zu `$oldbox`
- Probleme beim Nutzen bildschirmorientierter Anwendungen
- „Lösung“: Manuelles Setzen von `$TERM` auf einen von beiden Seiten verstandenen Standard (z.B. `vt100`)
- Lösung: `~/.terminfo/`

Dann ist jetzt alles ok?

- Leider nein.
- Problem: Lügen vieler Terminal-Emulatoren – Ausgeben von konsole, gnome-terminal, PuTTY und vieler anderer als xterm
- Das *wäre* kein Problem – *wenn* sie alle Features von xterm unterstützten und auch alle Besonderheiten von xterm übernehmen.
- In der Realität: Implementierung nur einer Untermenge von xterms Features durch die meisten Nachahmer

Workarounds

- Massive Beschneidung des `xterm`-terminfo-Eintrags auf die kleine Schnittmenge aller implementierten Features der am häufigsten verwendeten Terminal-Emulatoren
- Notwendigkeit für zusätzliche Checks in den Applikationen

SIGWINCH

- Problem: Möglichkeit, Terminals *während der Laufzeit* in der Größe zu verändern
- Früher: Fehlen einer Möglichkeit des Mitteilens dieser Information den laufenden Programmen
- Heute: Senden des Signals SIGWINCH bei Veränderung der Größe (und anderen Eigenschaften)

„Terminal-Viren“

- „Viren“ bei alten textbasierten BBS:
Nutzen von ANSI-Sequenzen zur Farbverstellung,
Tastaturumbelegung, etc.
- Unterstützung für gefährlichere Sequenzen durch heutige
Terminal-Emulatoren
- Niemals direkte Ausgabe von fraglichen Dateien per `cat`;
Stattdessen `less` oder `$EDITOR`
- Siehe auch: `'locate ctlseqs'`

Möglichkeiten zur Wiederherstellung

- Alternativer Zeichensatz? `^O`
- Kein lokales Echo? `stty sane`
- Vollständiger Reset nötig? `reset`

Terminal-Emulatoren

- xterm, uxterm (viele Features)
- konsole, gnome-terminal
(Integration in die jeweilige Desktop-Umgebung)
- rxvt, rxvt-unicode, aterm, wterm, ...
(geringer Ressourcenverbrauch)

(u)xterm

- xterm älter als X!
- Viele Versionen und Varianten
- Viele Features

Features

- Weniger Ressourcenverbrauch im Vergleich zu Terminal-Emulatoren großer Desktop-Umgebungen
- Exzellente Konfigurationsmöglichkeiten, auch zur Laufzeit
- **u**xterm: exzellente Unicode- und UTF-8-Unterstützung

Konfiguration: X-Resources

- Konfigurationseinstellungen über die X-Resources-Datenbank
- Siehe auch: xterm(1), xrdb(1)

Beispiel-~/Xdefaults

```
! Farben
UXTerm*background:  #000000
UXTerm*foreground:  #b2b2b2
! Statt fetter Schrift lieber hellere
UXTerm*colorBDMode: true
! Gut leserliche dicktengleiche Schrift
UXTerm*faceName:    Andale Mono
UXTerm*faceSize:    12
! Scrollbar nimmt nur Platz weg
UXTerm*scrollBar:   false
UXTerm*saveLines:   10000
```

Konfiguration: Kommandozeilen-Optionen

- `-geometry` *Standard-Geometrie-Beschreibung*
- `-e` *Programm*
- `-xrm` *zusätzlicher xrbdb-Eintrag*
- Siehe auch: `xterm -h`

Konfiguration: Maus-Menüs

- `<Ctrl> + <Button 1>`: Generelle Optionen
 - Senden von Signalen (u.a. SIGTERM, SIGKILL)
 - Neuzeichnen des Fensters
 - „Secure Keyboard“
- `<Ctrl> + <Button 3>`: Terminal-Optionen
 - Reverse Video
 - Visual Bell
 - Margin Bell
 - Reset
- `<Ctrl> + <Button 2>`: Auswahl der Schriftgröße

screen

- Fenstermanager für die Konsole mit vielen weiteren Features
- Erste Version 1987
- Oft genutztes Programm
(wenig Fehler, viel Dokumentation im Netz)
- Nach Konfiguration enorme Arbeitserleichterung!

Mehrere Fenster

- `^A c` (create): Erzeugen eines neuen Fensters
- `^A K` (kill): Abschießen des aktuellen Fensters
- `^A w` (windows): Anzeige des Fensterliste in der Statuszeile
- `^A "`: Anzeige des Fensterliste als Menü

- `^A n` (next): Wechsel zum nächsten Fenster
- `^A p` (prev): Wechsel zum vorherigen Fenster
- `^A a`: Wechsel zum zuletzt ausgewählten Fenster
- `^A Nummer`: Wechsel zu Fenster *Nummer*

- `^A M` (monitor): Überwachen des aktuellen Fensters auf Aktivität
- `^A _:` Überwachen des aktuellen Fensters auf Stille

Scrollback-Buffer

- `^A [`: Wechsel in Scrollback-Buffer
- `^A .:` Vorzeitige Rückkehr in normalen Betriebsmodus

- `^A <Space>`: Setzen einer Markierung

- `^A]`: Einfügen der Markierung

Statuszeile

Beispiel-~/ .screenrc

```
# Statuszeile immer anzeigen
hardstatus alwayslastline

# Datum ganz rechts anzeigen
hardstatus string "%h%= %D, %d.%m. %c"
# %h:           Normaler Status-Text
# %=:           Folgenden Text rechts ausrichten
# %D, %d.%m. %c: Di, 06.09. 18:21
```

„Abdocken“

- `^A ^D` (detach): Beenden von screen (aber nicht der Programme!)
- `^A *` (display): Anzeige aller „angedockten“ screens
- `$ screen -RD`: Andocken an eine frühere Sitzung oder Erstellen einer neuen

Beispiel-~/`.screenrc`

```
# Automatisch abdocken wenn das echte Terminal  
# stirbt (Absturz des X-Servers, Tod einer SSH-  
# Verbindung, etc.)  
autodetach on
```

Viele Konfigurationsmöglichkeiten

- Vielseitige Konfigurationsmöglichkeiten
- `~/ .screenrc`, Kommandozeilen-Parameter

- Umbelegung von Tastenbindungen
- Vorbelegung einiger Fenster für bestimmte Programme
- Detaillierte Anpassung der Statuszeile
- Ab-/Einschalten von UTF-8 für einzelne Fenster
- ...

Funktionsweise

- Erzeugung eines eigenen virtuellen Terminals (`TERM=screen`) für jedes Fenster
- Verstecken des echten Terminals vor den Programmen in der `screen`-Session
- Übersetzen der für `TERM=screen` passenden Escape-Codes in die Codes des unterliegenden Terminals (`TERM=xterm`, `TERM=linux`, ...)
- Damit auch möglich: Starten einer `screen`-Session in (z.B.) `xterm` und dann späteres Andocken in PuTTY!

screen-nodestatus

- Problem:
 - „Habe ich das Gateway eingeschaltet?“
 - „Ist mein Server online?“
 - „Spielt AugustaKOM mal wieder Ping-Pong mit meinen Paketen?“
- Lösung: Anzeige des Erreichbarkeitsstatus beliebiger Rechner in screens Statuszeile durch screen-nodestatus
- Platz in der Statuszeile kostbar –
 - Anzeige nur der Anfangsbuchstaben der Namen der überwachten Rechner
 - Großer Anfangsbuchstabe? Rechner erreichbar
 - Kleiner Anfangsbuchstabe? Rechner nicht erreichbar

screen-nodestatus: Funktionsweise

Beispiel-~/ .screenrc

```
# Anzeige des jeweils letzten Zeile des Ausgabe  
# von backtick-Befehl #40 in der Statuszeile  
hardstatus string "...%40'..."  
  
# Definition des backtick-Befehls #40  
backtick 40 0 0 programm  
# 40: ID  
# 0 0: Einmaliges Starten des Programms,  
# Anzeige der jeweils letzten Zeile der Ausgabe
```

Siehe auch

<http://www.catb.org/~esr/terminfo/>,
term(7), terminfo(5), termcap(5), tset(1), stty(1):

Informationen über die terminfo-Datenbank und \$TERM

<http://thread.gmane.org/gmane.linux.gentoo.devel/30624>:
„Fixing the TERM mess“

<http://dickey.his.com/xterm/xterm.faq.html>, xterm(1):
xterms Geschichte und Konfigurationsmöglichkeiten

<http://www.x.org/pub/unsupported/doc/papers/tutorials/resources.txt>:
Umgang mit der X-Resources-Datenbank

<http://www4.informatik.uni-erlangen.de/~jnweiger/screen-faq.html>, screen(1):
Lösungen zu Problemen mit screen, Konfigurationsmöglichkeiten

<http://svn.openfoundry.org/pugs/examples/network/screen-nodestatus.p6>:
screen-nodestatus (in Perl 6)