tmux(1) für screen(1)-Affine

ZEDAT

Julius Plenz

2011-07-21

Ablauf

1. **Konzepte:** Was ist neu, was bleibt gleich?
2. **Converter's Config:** Den Umsteig erleichtern
3. **Windows/Panes** ... und wie man damit umgeht
4. **Features:** Was gibt's für Nettigkeiten?
5. **Was geht nicht?**

Konzepte: Was bleibt?

- Mehrere Programme in einem Terminal
- Client-Server-Prinzip (detach/reattach)
- Eigene Tastaturkürzel
- Anpassbarer Statusbalken
- Copy-Mode

Konzepte: Was ist neu?

- Es gibt *einen* tmux-Server
  - *Pro*: Clients können zwischen verschiedenen *Sessions* hin- und hergeschoben werden
  - *Contra*: Wenn der abstürzt, ist alles futsch
- Wildcards via `fnmatch(3)`
- Panes, Windows, Sessions, Clients
  - Syntax: `<session>:<window>.<pane>`
  - z.B.: `:4`, `comm:1`, `:.`, `:.+`

Konfiguration: Grundlagen

- Konsistentes Subkommando-Handling
  - `tmux <command> <args>`
  - Prefix-`: <command> <args>`
  - `bind <key> <command> <args>`
- Einfache Befehle
  - `set`, `select-window`, `kill-pane`
- Template- und Confirm-Befehle
  - `command-prompt "rename-session '%%'"`
  - `confirm-before -p "kill-window #W? (y/n)" kill-window`

Konfiguration: Optionen setzen

- `set` arbeitet auf einer Session
  - mit `-w` für das *Window*
  - mit `-g` global (kombinierbar mit `-w`)

Nützliche Optionen

```
set -g base-index 1
set -g default-shell /bin/zsh
set -g prefix <prefix-key>
set -w automatic-rename on
set -w -g automatic-rename on
```

Konfiguration: Tasten wie in `screen(1)`!

- Default-Prefix ist Ctrl-B, warum auch immer
- `C-a a` soll `C-a` senden
- `C-a C-a` soll zum vorherigen Fenster wechseln

screen(1)-artige Keybinds

```
unbind C-b
set -g prefix C-a
bind a send-prefix
bind C-a last-window
```

Konfiguration: Tasten wie in `screen(1)`! (cont)

- Space und Backspace zum "cyclen"
- Feature: *repeatable key binds*

screen(1)-artige Keybinds

```
bind -r Space next-window
bind -r C-Space next-window
bind -r C-h previous-window
bind -r C-?  previous-window
```

Konfiguration: Statusbalken

- Schritt für Schritt konfigurierbar via Variablen
- `status-{left,right}`
  - Text wird durch `strftime(3)` geschickt (`%H:%M`)
  - Escape-Sequenzen von tmux, z. B. `#H` = Hostname
  - `#[bg=red,nobold]` für Farb- und Modusanpassungen
  - `#(uptime)` für Shell-Kommandos

Status-Konfiguration

```
set -g status-left "#[fg=red]#S %H:%M "
set -g status-right "#H: \
    #(cut -d' ' -f1-3 /proc/loadavg)"
set -w -g window-status-current-bg red
set -w -g window-status-current-fg yellow
```

Attach und Detach

- Detach aus einem Client heraus: `C-a d`
- `tmux has-session -t <name>`
- `tmux new-session [-d] -s <name> [<command>]`
- `tmux attach-session [-dr] -t <name>`

Auto-Reattach

```
if tmux has-session -t main; then
  exec tmux attach -t main
else
  exec tmux new-session -s main 'irssi -!'
fi
```

Copy Mode

- Beginnen mit `[`, pasten mit `]`
- Modus bricht nicht so schnell ab wie in screen!

- Innerhalb Navigation wie in Vi oder Emacs (je nach `$EDITOR`)
- Buffer werden stack-artig nummeriert (neuster: `0`)
- Buffer auflisten: `#`
- Buffer pasten: `=`

- Es gibt kein simples Konzept, um Buffer zwischen Sessions auszutauschen

Windows und Panes

- Ein *Window* ist Teil einer *Session*
- Es hat einen Namen und eine Nummer
  - `:4`, `comm:mutt*`
  - Umbenennen: `,`
  - Umnummerieren: `.`
- Ein *Window* besteht aus *Panes*, also Teil-Fenstern
- *Windows* können zwischen Sessions verlinkt (geteilt, verschoben) werden, Panes nicht

Sinnvolle Split-Keybindings

```
bind | split-window -h
bind - split-window -v
```

Panes und Layouts

- Panes können nach verschiedenen Layout angeordnet werden
- Befehle `select-layout`, `next-layout`, `previous-layout`
- Der Befehl `list-windows` gibt das Layout mit aus
- "Spezielle" Layouts können in der Konfigurationsdatei gespeichert werden

- Panes werden nummeriert (anzeigen via `q`)
- Anderes Fenster als Pane "hereinholen": `join-pane`
- Aktuelles Pane als neues Fenster aufteilen: `break-pane` (`!`)
- Aktuelles Pane schließen: `kill-pane`

Windows

- Ein Window gehört zu mindestens einer Session!
- Man kann ein Window einer anderen Session attachen (linken)
  - `link-window -s <source> [-t <target>]`
  - `move-window` analog

Schnellzugriff auf `mutt` und `irssi`

```
bind M link-window -s comm:mutt*
bind I link-window -s comm:irssi*
bind K unlink-window
```

Schneller Fensterwechsel

- `next-window -a` springt zum nächsten aktiven Fenster (bell)
- `f` sucht nach einem Fenster
  - Ausdruck via `fnmatch(3)`
  - ... auch auf dem Fenster**inhalt**
- `w` zeigt eine Liste der Fenster an (mit ausgeführtem Kommando)
- `s` zeigt eine Liste der Sessions an
- Session-Namen mit Bedacht wählen, eindeutige Titel verwenden (`,`)
  - Für "wichtige" Fenster kann man dann Keybinds definieren (Syntax: `<session>:<window-name>`)

Nette Features

- ClusterSSH-Ersatz
  - `set -w synchronize-panes` (toggle)
  - → Bei Bedarf Keybinding definieren
- Vi-Modus (und Tab-Completion) in der Status-Zeile und Copy-Mode
- Command chaining
  - `bind R source-file ~/.tmux.conf \; display-message "tmux:  config reloaded"`
- Ein `tmux`-Kommando aus `tmux` heraus kennt die eigene Pane
  - `[ -n "$TMUX" ] && tmux set -w ...`
- Watch auf beliebige Strings (bisher erst halbwegs nützlich)

Was kann `screen`, was `tmux` nicht kann?

- Keybindings mit > 1 Tasten
- Digraph Input
- Cutbuffer-Sharing

Danke!

Das war's!