Keep It Simple and Stupid
Das KISS-Prinzip in der Praxis

Daniel Friesel Maximilian Gaß

Chaosdorf

30. April 2010

Was ist KISS?

- KISS – Keep It Simple and Stupid
- (auch: Keep It Simple, Stupid)
- Es geht nicht primär um Bloat oder Performancevorteile

Was ist KISS?

- KISS – Keep It Simple and Stupid
- (auch: Keep It Simple, Stupid)
- Es geht nicht primär um Bloat oder Performancevorteile

Quick C ist nicht Quick, SMTP ist nicht Simple, LDAP ist nicht Lightweight, SQL ist nicht Structured, Windows 2000 Professional ist nicht Professional. Seht ihr? Ist doch ganz einfach.

– Felix von Leitner, de.comp.security.misc

Initskripte

Was sind Initskripte?

- Starten/Stoppen Services (Daemons)
- Integraler Bestandteil jedes unixoiden Systems

Debian I

```
1  #!/bin/sh
2
3  ### BEGIN INIT INFO
4  # Provides:          mpd
5  # Required-Start:    $local_fs $remote_fs
6  # Required-Stop:     $local_fs $remote_fs
7  # Should-Start:      autofs $network alsa-utils pulseaudio
8  # Should-Stop:       autofs $network alsa-utils pulseaudio
9  # Default-Start:     2 3 4 5
10 # Default-Stop:      0 1 6
11 # Short-Description: Music Player Daemon
12 # Description:       Start the Music Player Daemon (MPD) service
13 #                    for network access to the local audio queue.
14 ### END INIT INFO
15
16 . /lib/lsb/init-functions
17
18 PATH=/sbin:/bin:/usr/sbin:/usr/bin
19 NAME=mpd
20 DESC="Music Player Daemon"
21 DAEMON=/usr/bin/mpd
22 MPDCONF=/etc/mpd.conf
23 START_MPD=true
24
25 # Exit if the package is not installed
```

Debian II

```
26  [ -x "$DAEMON" ] || exit 0
27
28  # Read configuration variable file if it is present
29  [ -r /etc/default/$NAME ] && . /etc/default/$NAME
30
31  if [ -n "$MPD_DEBUG" ]; then
32      set -x
33      MPD_OPTS=--verbose
34  fi
35
36  DBFILE=$(sed -n 's/^[[:space:]]*db_file[[:space:]]*"\?\([^"]*\)"\?/\1/p' $MPDCONF)
37  PIDFILE=$(sed -n 's/^[[:space:]]*pid_file[[:space:]]*"\?\([^"]*\)"\?/\1/p' $MPDCONF)
38
39  mpd_start () {
40      if [ "$START_MPD" != "true" ]; then
41          log_action_msg "Not starting MPD: disabled by /etc/default/$NAME".
42          exit 0
43      fi
44
45      log_daemon_msg "Starting $DESC" "$NAME"
46
47      if [ -z "$PIDFILE" -o -z "$DBFILE" ]; then
48          log_failure_msg \
49              "$MPDCONF must have db_file and pid_file set; cannot start daemon."
50          exit 1
```

Debian III

```
51          fi
52
53          PIDDIR=$(dirname "$PIDFILE")
54          if [ ! -d "$PIDDIR" ]; then
55              mkdir -m 0755 $PIDDIR
56              chown mpd:audio $PIDDIR
57          fi
58
59          if [ "$FORCE_CREATE_DB" -o ! -f "$DBFILE" ]; then
60              log_warning_msg "creating $DBFILE... "
61              $DAEMON --create-db "$MPDCONF" > /dev/null 2>&1
62          fi
63
64          start-stop-daemon --start --quiet --oknodo --pidfile "$PIDFILE" \
65              --exec "$DAEMON" -- $MPD_OPTS "$MPDCONF"
66          log_end_msg $?
67      }
68
69      mpd_stop () {
70          if [ "$START_MPD" != "true" ]; then
71              log_failure_msg "Not stopping MPD: disabled by /etc/default/$NAME".
72              exit 0
73          fi
74          if [ -z "$PIDFILE" ]; then
75              log_failure_msg \
```

Debian IV

```
76                  "$MPDCONF must have pid_file set; cannot stop daemon."
77              exit 1
78          fi
79
80          log_daemon_msg "Stopping $DESC" "$NAME"
81          start-stop-daemon --stop --quiet --oknodo --retry 5 --pidfile "$PIDFILE" \
82              --exec $DAEMON
83          log_end_msg $?
84      }
85
86      # note to self: don't call the non-standard args for this in
87      # {post,pre}{inst,rm} scripts since users are not forced to upgrade
88      # /etc/init.d/mpd when mpd is updated
89      case "$1" in
90          start)
91              mpd_start
92              ;;
93          stop)
94              mpd_stop
95              ;;
96          restart|force-reload)
97              mpd_stop
98              mpd_start
99              ;;
100         force-start|start-create-db)
```

Debian V

```
101             FORCE_CREATE_DB=1
102             mpd_start
103             ;;
104         force-restart)
105             FORCE_CREATE_DB=1
106             mpd_stop
107             mpd_start
108             ;;
109         *)
110             echo "Usage: $0 {start|start-create-db|stop|restart}"
111             exit 2
112             ;;
113     esac
```

Arch Linux I

```
1  #!/bin/bash
2  . /etc/rc.conf
3  . /etc/rc.d/functions
4  case "$1" in
5    start)
6      stat_busy "Starting Music Player Daemon"
7      /usr/bin/mpd /etc/mpd.conf &> /dev/null
8      if [ $? -gt 0 ]; then
9        stat_fail
10     else
11       add_daemon mpd
12       stat_done
13     fi
14     ;;
15   stop)
16     stat_busy "Stopping Music Player Daemon"
17     /usr/bin/mpd --kill /etc/mpd.conf &> /dev/null
18     if [ $? -gt 0 ]; then
19       stat_fail
20     else
21       rm_daemon mpd
22       stat_done
23     fi
24     ;;
25   create-db)
```

Arch Linux II

```
26        stat_busy "Creating mpd's database ..."
27            logpath="/var/log/mpd/mpd.db-creation"
28        /usr/bin/mpd --create-db /etc/mpd.conf > $logpath \
29            && stat_busy "Output written to $logpath"
30                  stat_done
31        ;;
32      restart)
33        $0 stop
34        sleep 1
35        $0 start
36        ;;
37      *)
38        echo "usage: $0 {start|stop|restart|create-db}"
39    esac
40    exit 0
```

daemontools

```
1 #!/bin/sh
2 exec mpd --no-daemon --stdout ./mpd.conf
```

upstart

```
1 start on runlevel [2345]
2 stop on runlevel [!2345]
3 respawn
4 exec mpd /etc/mpd.conf
5 expect fork
```

Fazit

- Gewaltige Unterschiede
- Simplicity hat in diesem Bereich keine Nachteile

Was ist Linking?

- Einbinden von Modulen in (C-)Programme
- Statisches vs. Dynamisches Linken

Statisches Linken

- Kein Linken beim Programmstart
- Keine Probleme mit ABI-Changes
- Weniger Abhängigkeiten auf Libraries

Dynamisches Linken

- Ein zentraler Patch fixt alle Binaries
- Geringerer RAM-Verbrauch und Plattenplatzbedarf
- Standard, daher wenig fummelig beim Bauen

Fazit

- Situationsabhängig, aber dynamisches Linken ist de-facto Standard

Zsh vs. mksh

- zsh: Z Shell, extrem konfigurierbar und featurereich
- mksh / pdksh: Korn Shell-Varianten, können einiges, aber weniger

Nachteile der zsh

- Sehr groß (536KiB Binary, mksh/pdksh: 224/196 KiB)
- Hoher RAM-Verbrauch (5-10M pro Instanz)
- Sehr komplex

Nachteile der zsh

- Sehr groß (536KiB Binary, mksh/pdksh: 224/196 KiB)
- Hoher RAM-Verbrauch (5-10M pro Instanz)
- Sehr komplex

shell programming is like LSD, you often see advantages that are not there.

– Zaba

Vorteile der mksh/pdksh

- Niedriger RAM-Verbrauch (500-800K)
- Schnelle Startup-Zeit
- Verhältnismäßig simpel

Aber: Bequemlichkeit

- zsh kann einfach mehr
- $\rightarrow$ Abwägen, was einem wichtiger ist

HAL

- “Hardware Abstraction Layer”
- Input-Hotplug, wobei X einzelne Geräte sehen kann
- Hotplug selbst ging und geht auch ohne HAL
- Klassisch: Flexibilität für wenige $\rightarrow$ mehr Komplexität für alle
- Obskure XML-Configs
- Inzwischen: libudev

udev vs. mdev

Device-Hotplugging (/dev)

udev

- Daemon (Sicherheitsprobleme)
- udevadm settle: Teilweise sehr langsam

mdev

- Kein Daemon
- kein Netlink-Socket, klassische Argumente / Environment
- Hat trotzdem fast alle udev-Features
- Schnell (0.1s für alle Devicenodes)

Fazit

Linux is becoming an idiot box; a nanny operating system. The complexity and the idiot box ideals are killing the power user, and the spare time developer.

– Tuomo Valkonen

Links

- suckless.org
- harmful.cat-v.org