```
                     __                   .___
  _________.__. _______/  |_  ____   _____    __| _/
 /  ___<   |  |/  ___/\   __\/ __ \ /     \  / __ |
 \___ \ \___  |\___ \  |  | \  ___/|  Y Y  \/ /_/ |
/____  >/ ____/____  > |__|  \___  >__|_|  /\____ |
     \/ \/         \/            \/      \/      \/
```

shutdown
Activation
services
handling
files
logging
Plymouth
processes
Automatic
without
Early
compatibility
tools
service
support
integration
Builtin
termination
state
sysv
device
system
generation
machine
boot
bootup
console
Socketbased
CPU
editing
cgroups
Reliable
Dynamic
Graphical
Reexecution
advanced
Respawning
embedded
Masking
PolicyKit
kmsgbased
daemon
Systemwide
devlog
Dependencybased
random
Profiling
SSL
Static
utmpwtmp
process
kernel
sessions
management
Minimal
Loopback
procsys
binfmtmisc
built
suitable
delivery
Optionally
ReadAhead
loading
use
Unique
components
Infrastructure
Remote
Saverestore
cleaning
Interfacing
volatile
Included
accessCluster
Container
losing
remaining
kills
crash
name
meta
SELinux
various
started
client
UI
wall1
ID
compatible
locale
modules
inetd
even
Robust
peruser
Timerbased
Instantiated
Mount
Swap
dependencies
data
writable
Modular
serial
rules
TTY
users
inotify
full
hard
LUKS
like
syslog
parseable
Automount
keyboard
Password
fsck
Quota
host
OS
extensible
GNOME
Pathbased
distributions
setup
DBus
list
PAM
CertificateLUKS
within
Linux
chroot
coded
udev
user
record
Shellfree
Control
Devicebased
disk
temporary
seed
via
SysVcompatible
devinitctl
kexec
Configuration
Groups
Audit
agents
enterprise
removing
Interactive
Disabling
replacement
Busbased
serialization
PID
sysctl
upgrades
controllable
Snapshotting
usage
Signal
Gapless
XDGRUNTIMEDIR
Upstream
perservice
Network
Encrypted
creating
Easily
native
including
manipulation
identify
connectivity

fhLUG

Agenda

```
 -------------------
( Was ist Systemd ? )
( Background        )
( Vorteile          )
( Verwendung        )
( Mehr Verwendung   )
 -------------------
        o    ,-^-.
         o   !oYo!
          o /./=\.\______
               ##        )\/\
                ||-----w||
                ||      ||

               Cowth Vader
```

Was ist systemd

System / Service Manager

SysV Nachteile

Sehr alt

Scripts haben Nachteile

- Schwer wartbar
- Single threaded
- Shellscripts
- Keine Abildung vom Beziehungen

Vorteile

Einfache Verwaltung

Abghänigkeiten

Gute Performance

Einfacheres Logging und Debugging

Abwärtkompatible

Service On-Demand

Einfach zu lernen

Features über Features

Systemd vs the World

Why ?

- Bloat Software
- Not Unix Style
- Growing beyond scope
- Reifegrad
- Backdoor :P

Systemd vs the World

fhLUG

Systemd-Free

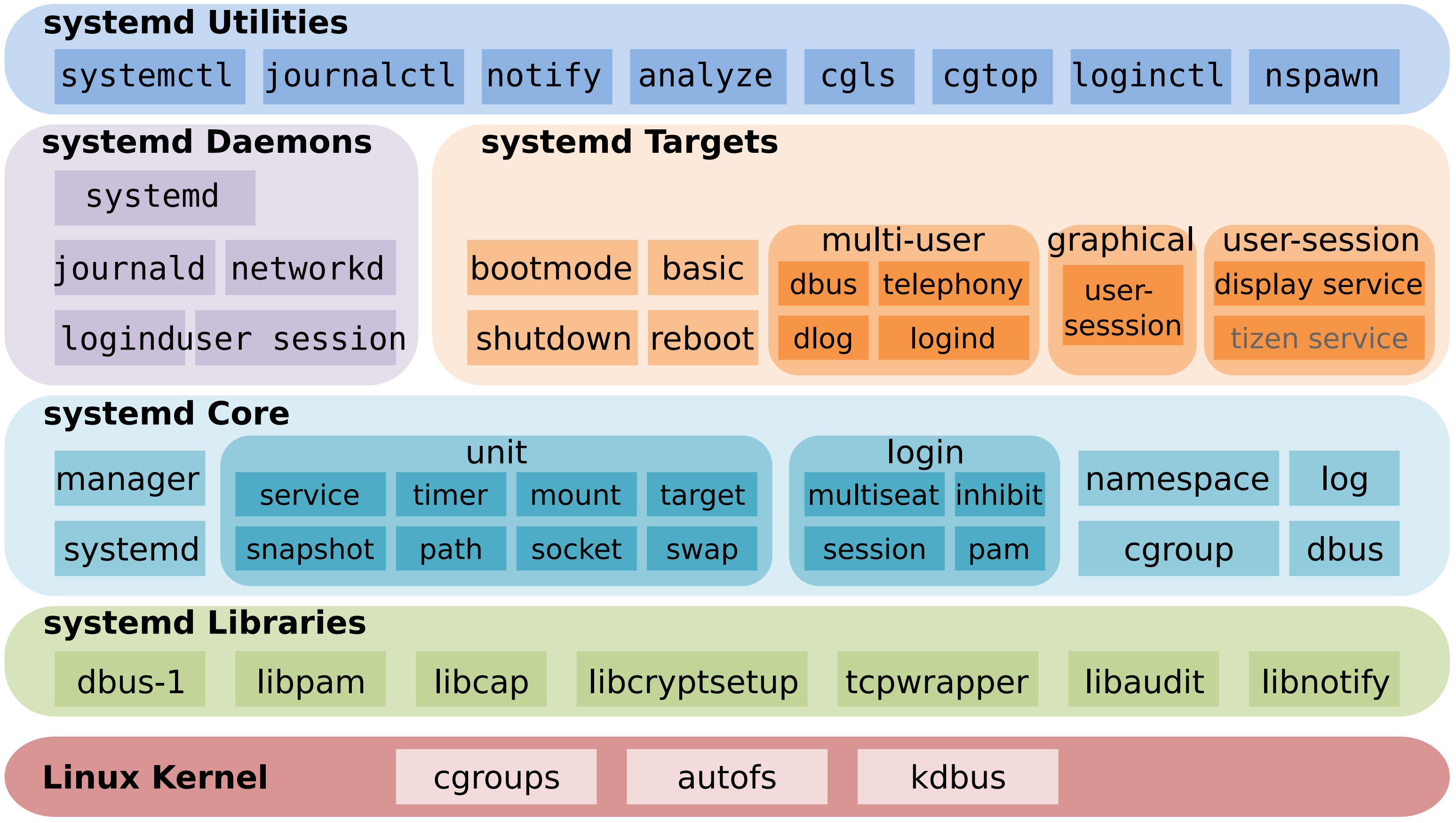
- Gentoo
- Devuan
- Slackware
- Void Linux
- Crux
- Alpine Linux
- gNewSense

http://without-systemd.org/

Countless Features

- Interfacing via D-Bus
- Shell-free bootup
- Modular C coded early boot services included
- Read-Ahead
- Socket-based Activation
- Socket-based Activation: inetd compatibility
- Bus-based Activation
- Device-based Activation
- Configuration of device dependencies with udev rules
- Path-based Activation (inotify)
- Timer-based Activation
- Mount handling
- fsck handling
- Quota handling
- Automount handling
- Swap handling
- Snapshotting of system state
- XDG_RUNTIME_DIR Support
- Optionally kills remaining processes of users logging out
- Linux Control Groups Integration
- Audit record generation for started services
- SELinux integration
- PAM integration
- Encrypted hard disk handling (LUKS)
- SSL Certificate/LUKS Password handling, including Plymouth, Console, wall(1), TTY and GNOME agents
- Upstream support in various other OS components
- Service files compatible between distributions
- Signal delivery to services
- Reliable termination of user sessions before shutdown
- utmp/wtmp support
- Easily writable, extensible and parseable service files, suitable for manipulation with enterprise management tools
- Network Loopback device handling
- binfmt_misc handling
- System-wide locale handling
- Console and keyboard setup
- Infrastructure for creating, removing, cleaning up of temporary and volatile files
- Handling for /proc/sys sysctl
- Plymouth integration
- Save/restore random seed
- Static loading of kernel modules
- Automatic serial console handling
- Unique Machine ID handling
- Dynamic host name and machine meta data handling
- Reliable termination of services
- Early boot /dev/log logging
- Minimal kmsg-based syslog daemon for embedded use
- Respawning on service crash without losing connectivity
- Gapless service upgrades
- Graphical UI
- Built-In Profiling and Tools
- Instantiated services
- PolicyKit integration
- Remote access/Cluster support built into client tools
- Can list all processes of a service
- Can identify service of a process
- Automatic per-service CPU cgroups to even out CPU usage between them
- Automatic per-user cgroups
- SysV compatibility
- SysV services controllable like native services
- SysV-compatible /dev/initctl
- Reexecution with full serialization of state
- Interactive boot-up
- Container support (as advanced chroot() replacement)
- Dependency-based bootup
- Disabling of services without editing files
- Masking of services without editing files
- Robust system shutdown within PID 1
- Built-in kexec support
- Dynamic service generation

Systemd Verwendung



fhLUG

Systemd Verwendung

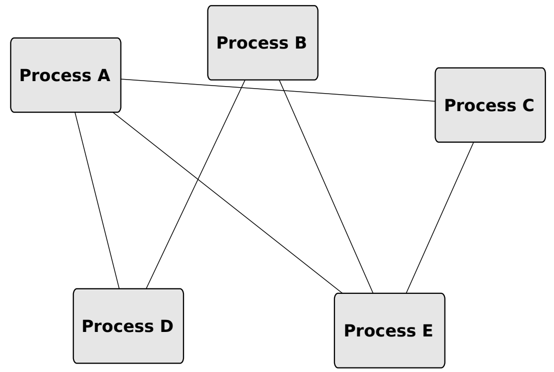
Steuert Ressourcen des Systems (mit Unit Files)

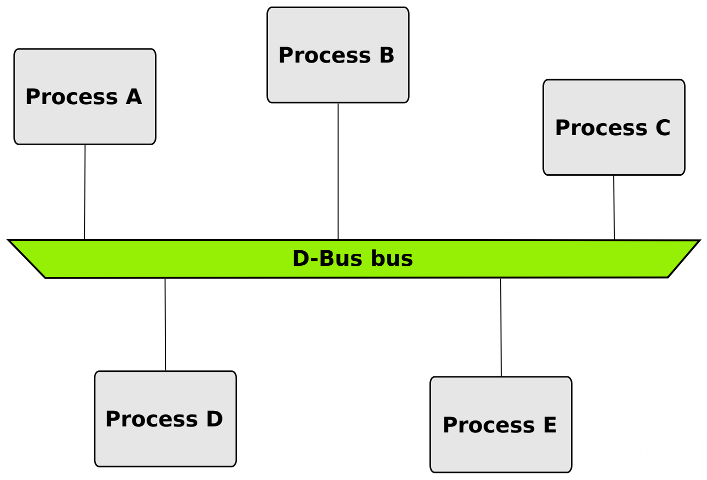
Es gibt keine RUN-Level mehr → Targets

/etc/fstab != Unit File → Generator

Für User und System

Benutzt D-Bus !





Unit File - Verwaltung

systemctl <status|start|stop|enable|disable> [unit]

systemctl list-unit-files

systemctl list-units

systemctl list-timer

systemctl edit [unit]

Tipp: Many systemd tools end with “ctl “ like busctl

Unit File – Types

systemd.service	systemd.target
systemd.socket	systemd.timer
systemd.device	systemd.slice
systemd.mount	systemd.scope
systemd.automount	systemd.network
systemd.swap	system.link

man system.unit

Unit Files - Status

States

- active
- inactive
- activating
- deactivating
- failed

LOAD

Reflects whether the unit definition was properly loaded.

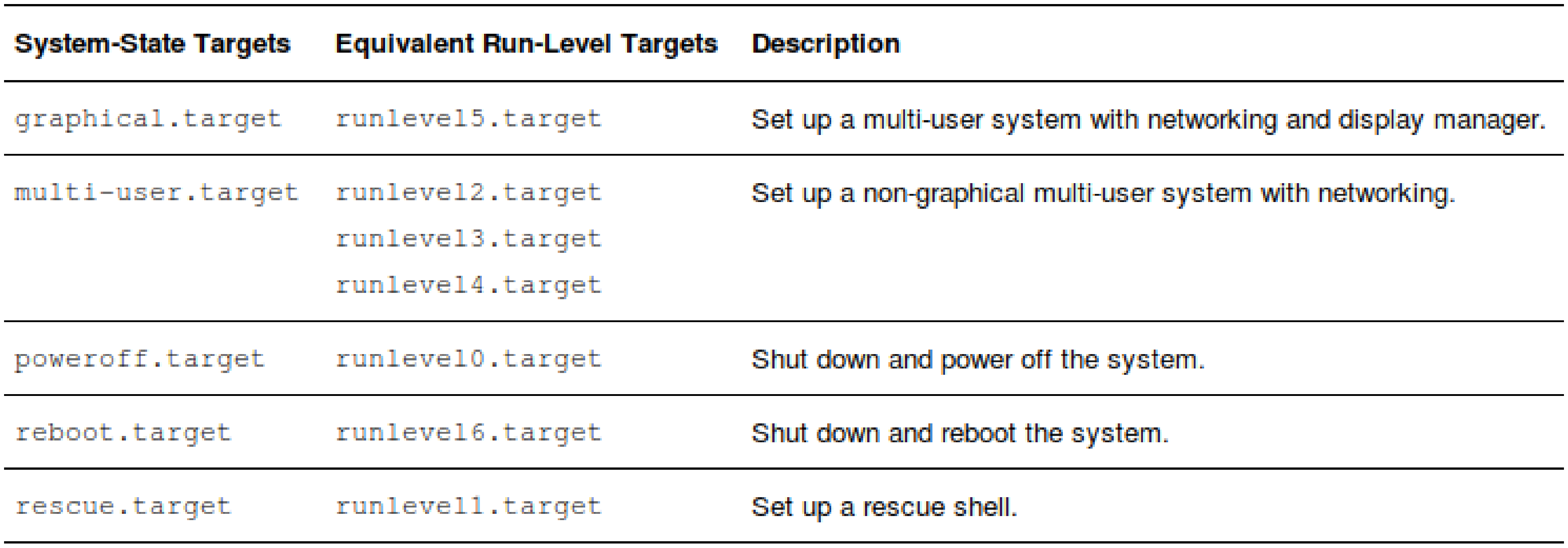
ACTIVE

The high-level unit activation state, i.e. generalization of SUB.

SUB

The low-level unit activation state, values depend on unit type.

Unit File - Target



System-State Targets	Equivalent Run-Level Targets	Description
`graphical.target`	`runlevel5.target`	Set up a multi-user system with networking and display manager.

`multi-user.target`	`runlevel2.target`
	`runlevel3.target`
	`runlevel4.target`
		Set up a non-graphical multi-user system with networking.

`poweroff.target`	`runlevel0.target`	Shut down and power off the system.
`reboot.target`	`runlevel6.target`	Shut down and reboot the system.
`rescue.target`	`runlevel1.target`	Set up a rescue shell.

systemctl list-units -p "Wants=multi-user.target”

Workshop - Time

Lass die List all deiner Unit Files ausgeben

Sieh dir den Status eines deiner Unit Files

Starte einen Service den du gerade nicht brauchst (z.B. sshd)

Probier den Prozess mit “kill -9 “ zu beenden was passiert ?

Teste mal das hier: “systemctl status /home”

Unit Files - Locations

System Services

Installed: /usr/lib/systemd/system

Configured: /etc/systemd/system

Runtime : /run/systemd

Drop-ins: /etc/systemd/system/[name.type].d/*.conf

User Services

Installed: ~/.local/share/systemd/user

Configured: ~/.config/systemd/user/

Runtime: /run/systemd/user

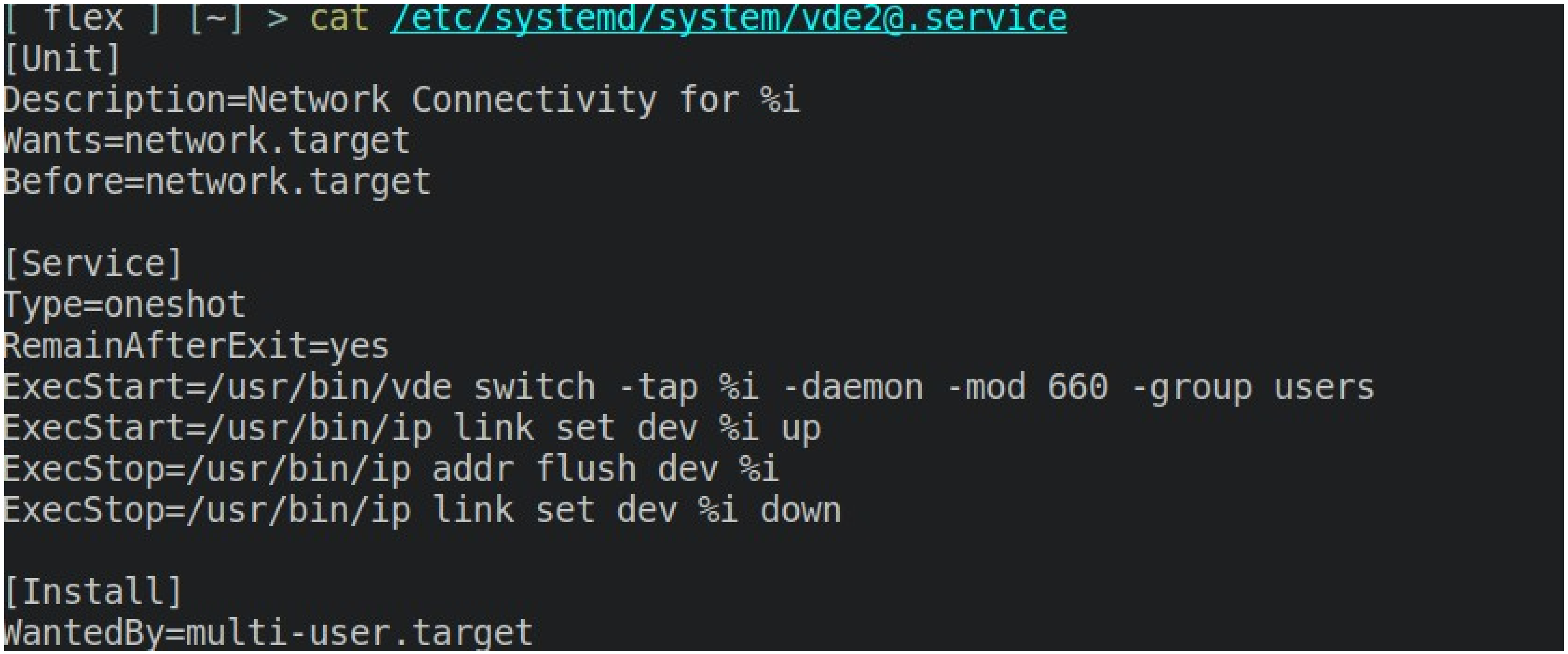
Unit File - Syntax

```
[Unit]
Description=OpenSSH Daemon
Wants=sshdgenkeys.service
After=sshdgenkeys.service
After=network.target

[Service]
ExecStart=/usr/bin/sshd -D
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=multi-user.target
```

Unit File – Syntax



```
[ flex ] [~] > cat /etc/systemd/system/vde2@.service
[Unit]
Description=Network Connectivity for %i
Wants=network.target
Before=network.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/vde switch -tap %i -daemon -mod 660 -group users
ExecStart=/usr/bin/ip link set dev %i up
ExecStop=/usr/bin/ip addr flush dev %i
ExecStop=/usr/bin/ip link set dev %i down

[Install]
WantedBy=multi-user.target
```

Examples - Link

```
# Change MAC
[Match]
MACAddress=a0:d0:96:03:b2:ca

[Link]
MACAddress=12:34:45:42:42:42
```

Example - Network

```
[Match]
Virtualization=container
Name=ethy1

[Network]
Address=192.168.1.10
Gateway=192.168.1.1
DNS=8.8.8.8
```

Example - Mount

```
[Unit]
SourcePath=/etc/fstab
Documentation=man:fstab(5) man:systemd-fstab-generator(8)
Before=local-fs.target
Requires=systemd-fsck@dev-disk-by\x2duuid-
64530e6e\x2d1e97\x2d4cb0\x2d90da\x2d6109792662b3.service
After=systemd-fsck@dev-disk-by\x2duuid-
64530e6e\x2d1e97\x2d4cb0\x2d90da\x2d6109792662b3.service

[Mount]
What=/dev/disk/by-uuid/64530e6e-1e97-4cb0-90da-6109792662b3
Where=/home
Type=ext4
Options=rw,relatime,data=ordered
```

Examples - Link

```
# Change MAC
[Match]
MACAddress=a0:d0:96:03:b2:ca

[Link]
MACAddress=12:34:45:42:42:42
```

Example - Network

```
[Match]
Virtualization=container
Name=ethy1

[Network]
Address=192.168.1.10
Gateway=192.168.1.1
DNS=8.8.8.8
```

Unit File - Installation

```
$EDITOR my_unit.service
```

System:

```
sudo mv my_unit.service /etc/systemd/system/
sudo systemctl start my_unit.service
Sudo systemctl enable my_unit.service
```

User:

```
mv  my_unit.service ~/.config/systemd/user
systemctl –user start my_unit.service
systemctl –user enable my_unit.service
```

Unit File – Security Things

- CGroups
- Einige Security Optionen:
  - InaccessibleDirectories=/home
  - ReadOnlyDirectories
  - MemoryAccounting=true
  - MemoryLimit=10M
- Tools
  - systemd-cgtop
  - systemd-cgls

Timers

Systemd ersetzt CRON

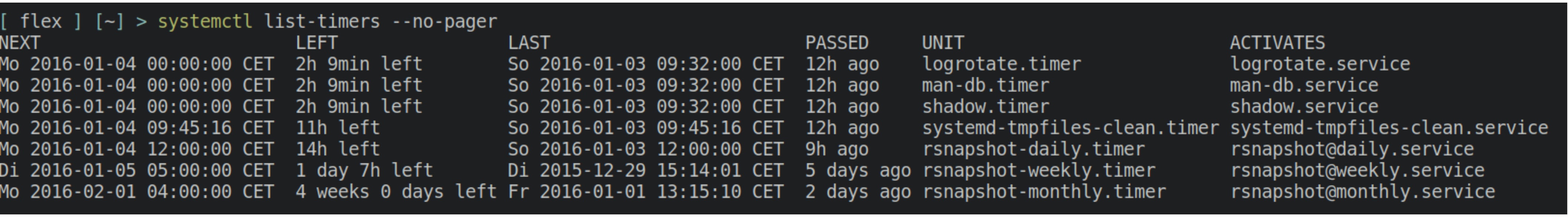
Pro

- Einfacher Fehler zu suchen
- Abhänigkeiten
- CGroups

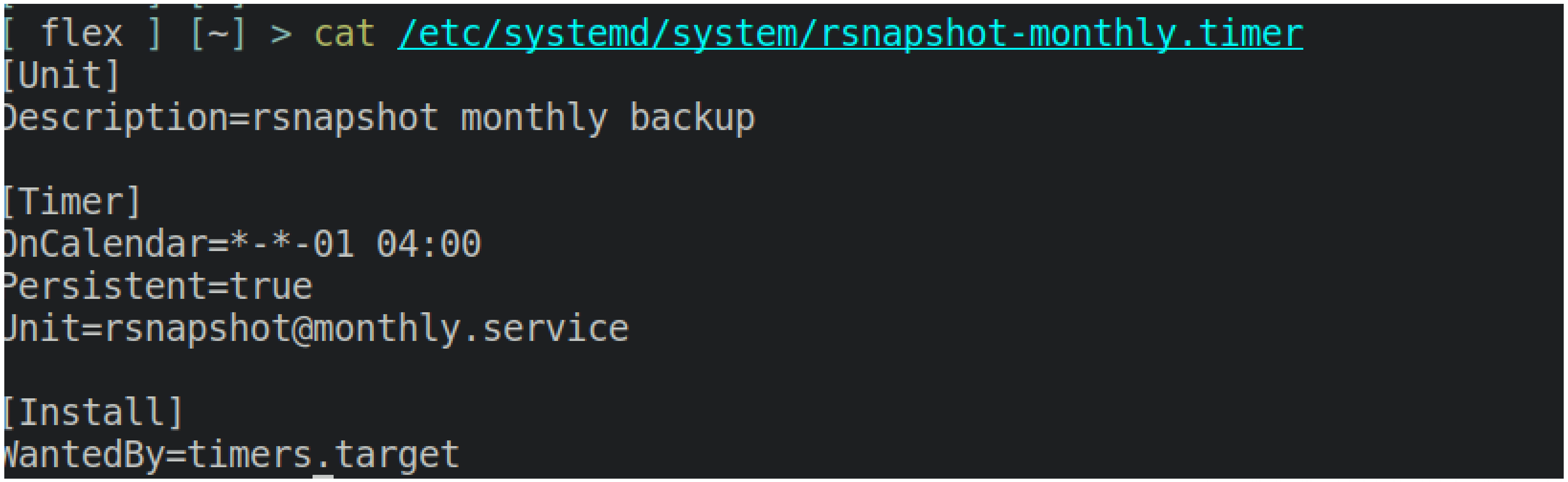
Con

- Kein MAILTO
- Mehr Aufwand

Timers



```
[ flex ] [~] > systemctl list-timers --no-pager
NEXT                          LEFT                LAST                          PASSED     UNIT                          ACTIVATES
Mo 2016-01-04 00:00:00 CET  2h 9min left        So 2016-01-03 09:32:00 CET  12h ago    logrotate.timer               logrotate.service
Mo 2016-01-04 00:00:00 CET  2h 9min left        So 2016-01-03 09:32:00 CET  12h ago    man-db.timer                  man-db.service
Mo 2016-01-04 00:00:00 CET  2h 9min left        So 2016-01-03 09:32:00 CET  12h ago    shadow.timer                  shadow.service
Mo 2016-01-04 09:45:16 CET  11h left            So 2016-01-03 09:45:16 CET  12h ago    systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.service
Mo 2016-01-04 12:00:00 CET  14h left            So 2016-01-03 12:00:00 CET  9h ago     rsnapshot-daily.timer         rsnapshot@daily.service
Di 2016-01-05 05:00:00 CET  1 day 7h left       Di 2015-12-29 15:14:01 CET  5 days ago rsnapshot-weekly.timer        rsnapshot@weekly.service
Mo 2016-02-01 04:00:00 CET  4 weeks 0 days left Fr 2016-01-01 13:15:10 CET  2 days ago rsnapshot-monthly.timer       rsnapshot@monthly.service
```



```
[ flex ] [~] > cat /etc/systemd/system/rsnapshot-monthly.timer
[Unit]
Description=rsnapshot monthly backup

[Timer]
OnCalendar=*-*-01 04:00
Persistent=true
Unit=rsnapshot@monthly.service

[Install]
WantedBy=timers.target
```

Logging

(Fast) Alle Logs

Loggt alle Nachrichten eines Unit Files

Binary Format → Nicht “grep” bar

Zentrales Dir. → /var/log/journal/ (meist)

Automatischer Upload möglich

Kryptographische Signaturen möglich

Umstieg auf syslog möglich

journalctl

Beispiele:

List aller Boots: journalctl --list-boots

Bestimmter Boot: journalctl -b 42

Follow Logging: journactl -f

Logs per Service: journalctl -u systemd-journald.service

Logs per Binary: journalctl /usr/bin/sudo

journalctl

Journalctl -p <syslog-level | syslog-id>

ID	Level
0	emerg
1	alert
2	err
3	warning
4	notice
5	info
6	debug

fhLUG

```
 __      __             __           .__                            ___________.__
/  \    /  \___________|  | __  _____|  |__   ____ _______          \__    ___/|__| _____   ____
\   \/\/   /  _ \_  __ \  |/ / /  ___/  |  \ /  _ \\____ \   ______   |    |   |  |/     \_/ __ \
 \        (  <_> )  | \/    <  \___ \|   Y  (  <_> )  |_> > /_____/   |    |   |  |  Y Y  \  ___/
  \__/\  / \____/|__|  |__|_ \/____  >___|  /\____/|   __/            |____|   |__|__|_|  /\___  >
       \/                   \/     \/     \/       |__|                                 \/     \/
```

Siehe dir eine List all deiner Boots an

Schau dir an was beim letzten Boot schief ging

Sieh dir alle Timer einmal an

Schau dir mal an was für Fehler es in deinem Log gibt

Login

Systemd hat PAM und Session Management

Session

- Gültige Anmeldung eines Nutzers am System
- Ein Benutzer kann viele Sessions haben
- Eine Session hat einen Seat

Seats

- Sammlung von HW
- Ein Seat – mehrer Sessions

loginctl

loginctl list-users

loginctl list-sessions

loginctl list-seats

loginctl user-status [uid]

loginctl session-status [session id z.B. c1]

loginctl seat-status [seat id z.B. seat0]

systemd-nspawn

- Kombi aus Chroot + Namespace
- Praktisch für sehr leichte Container
- Kombatibel mit Images von z.B. Docker
- Einfach in der Handhabung
- On-Board

systemd-nspawn

Beispiele

```
debootstrap --arch=amd64 unstable ~/debian-tree/
systemd-nspawn -D ~/debian-tree/
```

Systemmangement

timedatectl

localectl

hostnamectl

networkctl

busctl

loginctl

machinectl

systemd-analyze

```
 __      __             __               .__                               ___________.__
/  \    /  \___________|  | __  _____|  |__   ____ ______            \__    ___/|__| _____   ____
\   \/\/   /  _ \_  __ \  |/ / /  ___/  |  \ /  _ \\____ \   ______    |    |   |  |/     \_/ __ \
 \        (  <_> )  | \/    <  \___ \|   Y  (  <_> )  |_> > /_____/    |    |   |  |  Y Y  \  ___/
  \__/\  / \____/|__|  |__|_ \/____  >___|  /\____/|   __/             |____|   |__|__|_|  /\___  >
       \/                   \/     \/     \/       |__|                                  \/     \/
```

Schau dir deine User, Session und Seats an

Erstelle einen systemd-nspawn Container (tipp: man systemd.nspawn)

Sieh dir mal den output von hostnamectl an

“systemd-analyze”, wie schnell bist du ?

Thx for the fish