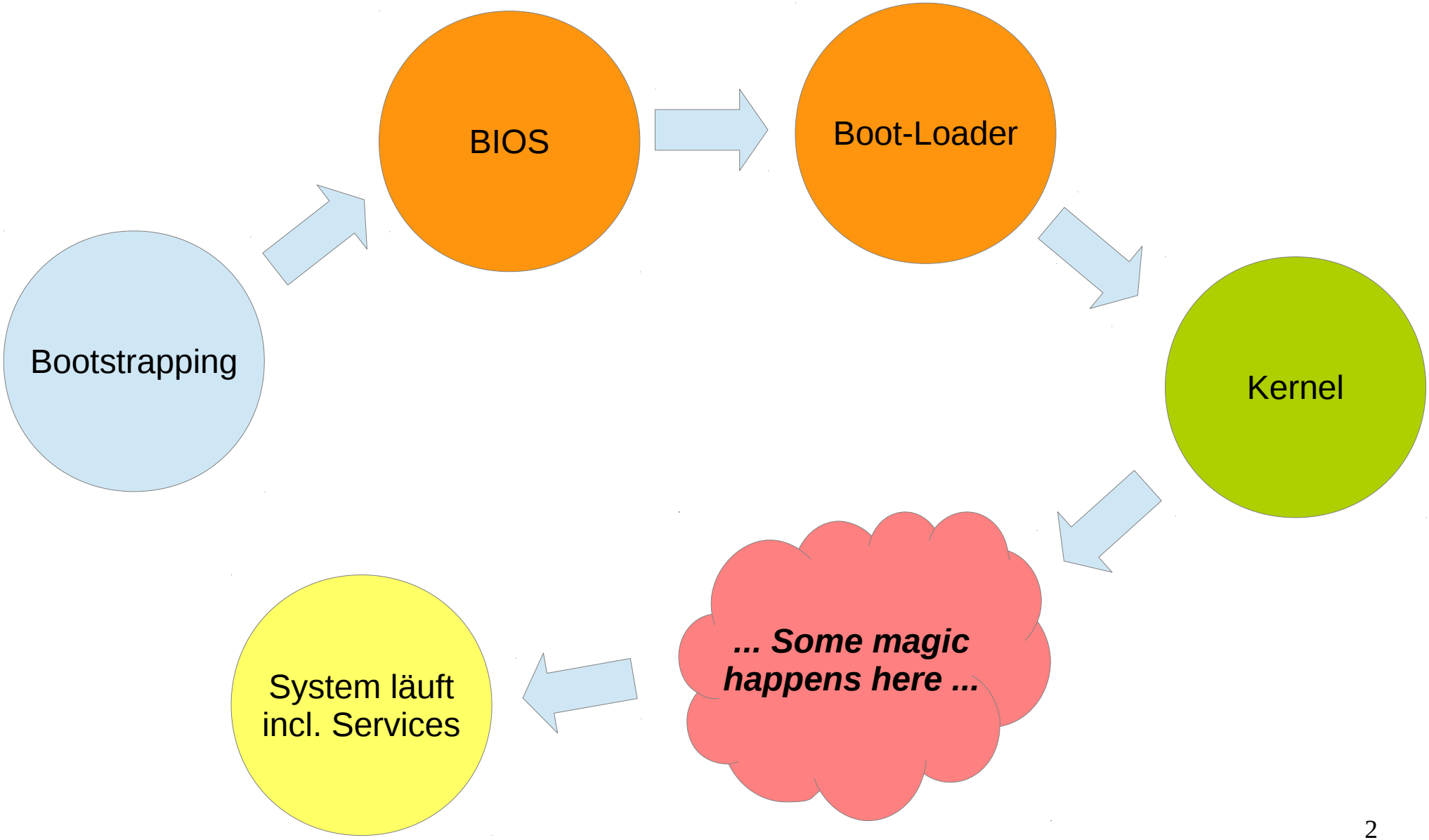
systemd

Der Bootprozess eines GNU/Linux Systems (Vortrag LUG Frankfurt 2010)



Der Init Prozess

Erste User-Space Applikation (/sbin/init) mit PID 1

Traditionell System V Init (von Unix System V, 1983)

- Ablauf kontrolliert von `/etc/inittab`
- Skripte unter `/etc/init.d`
- Symlinks in `/etc/rc<X>.d`
- Zustände des Systems über *Runlevel*
- Je nach Runlevel spezifische Skripte, um Services (daemons) zu starten oder zu stoppen

SysVInit Skript (rsyslog)

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          rsyslog
# Required-Start:    $remote_fs $time
# Required-Stop:     umountnfs $time
# X-Stop-After:      sendsigs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: enhanced syslogd
# Description:       Rsyslog is an enhanced multi-threaded syslogd.
#                    It is quite compatible to stock sysklogd and can be
#                    used as a drop-in replacement.
### END INIT INFO

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="enhanced syslogd"
NAME=rsyslog

RSYSLOGD=rsyslogd
RSYSLOGD_BIN=/usr/sbin/rsyslogd
RSYSLOGD_OPTIONS="-c5"
RSYSLOGD_PIDFILE=/var/run/rsyslogd.pid

SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$RSYSLOGD_BIN" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Define LSB log_* functions.
. /lib/lsb/init-functions

do_start()
{
        DAEMON="$RSYSLOGD_BIN"

DAEMON_ARGS="$RSYSLOGD_OPTIONS"
        PIDFILE="$RSYSLOGD_PIDFILE"

        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   other if daemon could not be started
or a failure occured
        start-stop-daemon --start --quiet --pidfile
$PIDFILE --exec $DAEMON --
$DAEMON_ARGS
}

do_stop()
        DAEMON="$RSYSLOGD_BIN"
        PIDFILE="$RSYSLOGD_PIDFILE"

        # Return
        #   0 if daemon has been stopped
        #   1 if daemon was already stopped
        #   other if daemon could not be
stopped or a failure occurred
        start-stop-daemon --stop --quiet
--retry=TERM/30/KILL/5 --pidfile $PIDFILE
--exec $DAEMON
}

[...]
```

Modernere Init Systeme

Probleme mit dem veralteten SysVInit:

- Skripte werden seriell abgearbeitet
- Bei I/O Block kann Startup u.u. sehr lange dauern
- Skripte können komplex werden
- Skripte erzeugen Overhead (CPU & Memory -> Embedded!) (bash, awk, sed etc.)
- Keine Kontrolle über Ressourcenverbrauch der Services
- Tracking / Beenden von Prozessen schwierig / hacky

  ➔ PID-Files, pidof, killall, pgrep...

- *upstart* (Ubuntu 2006 - 2015, Nokia N900 Maemo)
- *systemd* (Jetzt Standard in meisten Linux-Distros, April 2015 Ubuntu & Debian)
- *OpenRC* (Gentoo)
- *runit* (Void Linux)

systemd

- Lennart Poettering & Kay Sievert, 2010
- Ersatz für veraltetes SysVInit
- System & Service Manager (anstatt simples Startup)
- Deklarative *unit* files anstatt init Skripte
- Asynchronous & concurrent
- Socket Aktivierung von Services
- Isoliert Services in cgroups / namespaces
  (Namespace isolation, Resource Control)
- Logging, Timer units / cron-Ersatz, Container Management,
  Session Management, eigene Konsole, Netzwerkkonfiguration ...

systemd - Die Kontroverse

systemd ist einer der größten Streitpunkte in der Linux Community:

- Verabschiedung von der traditionellen Unix-Philosophie kleiner Tools, wo jedes genau eine Aufgabe erledigt

  ➔ wurde schon mit svchost.exe (Windows) verglichen
- Vorwurf von übermäßiger Komplexität & 'Feature Creep'
- "Zweiter Kernel neben dem Kernel"
- Nur unter Linux lauffähig (cgroups)
- Abhängigkeiten (z.B. GNOME), damit vollendete Tatsachen
- Kritik an Designentscheidungen (Logfiles in Binärformat etc.)
- Massive persönliche Auseinandersetzungen, Boykottaufrufe, Forks (*devuan*) ...

systemd Units

Unit zentrales Konzept von systemd

Units werden (meist) in einem Konfigurationsfile konfiguriert

Units haben einen Status:

Active / Inactive / Failed / Activating / Deactivating

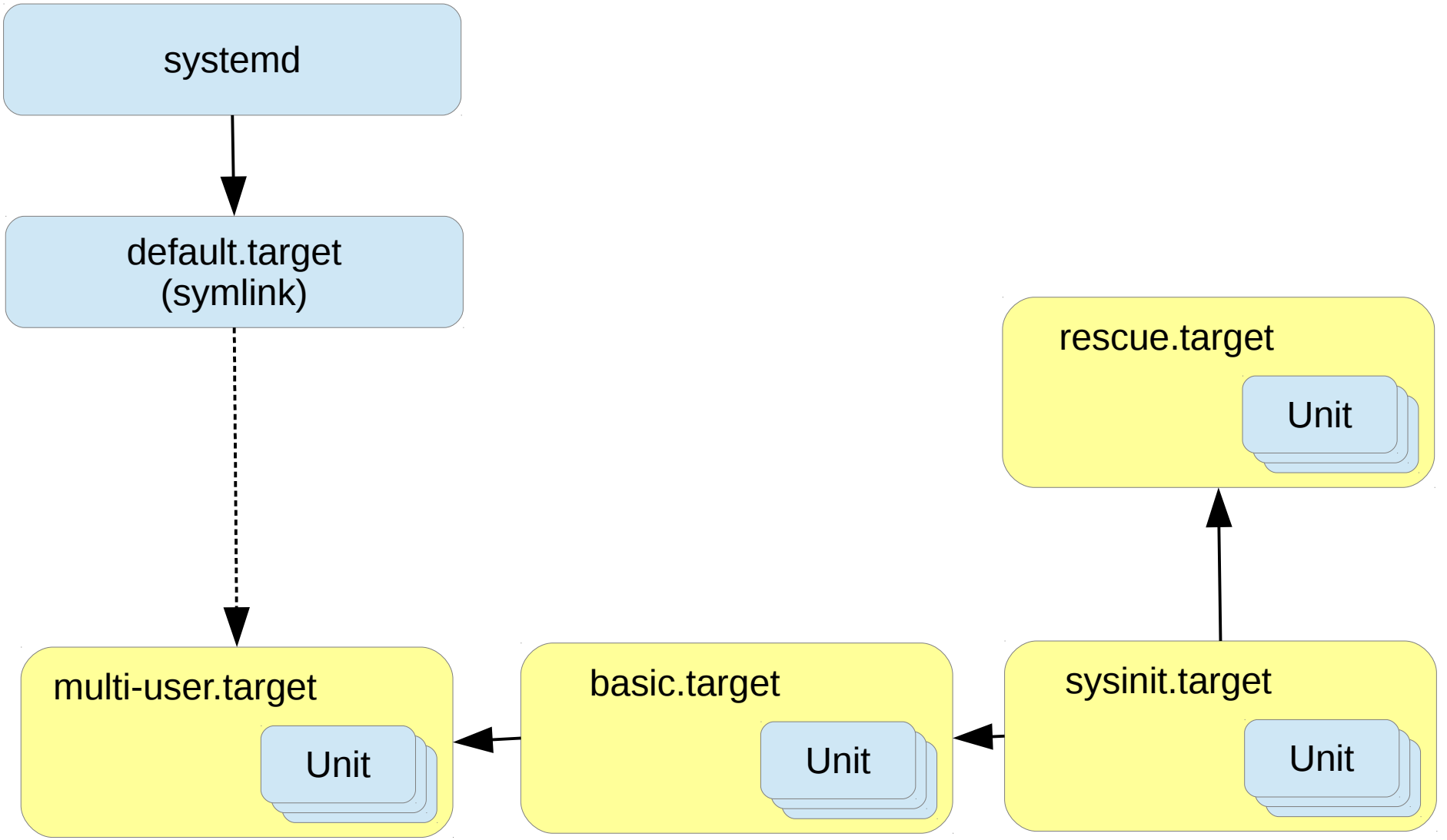
Unit Typen:

- *Service* - Daemons und deren Prozesse
- *Socket* - Netzwerk oder IPC Sockets, für socket-based activation
- *Target* - Gruppen von Units; ähnlich Runlevel
- *Device, Mount, Timer, Snapshot*, *Slices*, ...

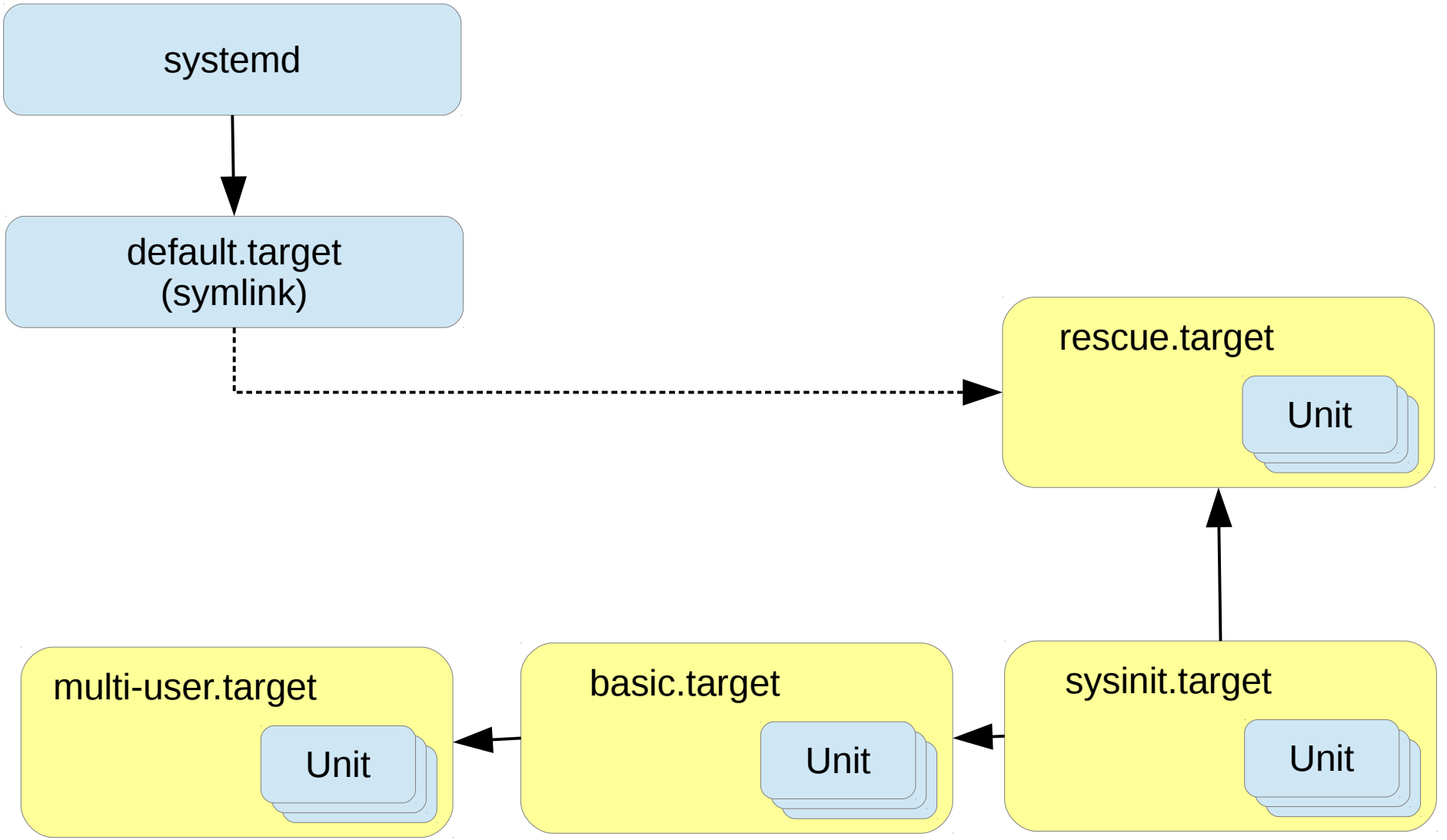
SysVInit Script vs. systemd Unit (rsyslog)

```
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
Documentation=http://www.rsyslog.com/doc/

[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

systemd Startup (Standard boot)



systemd Startup (Rescue)



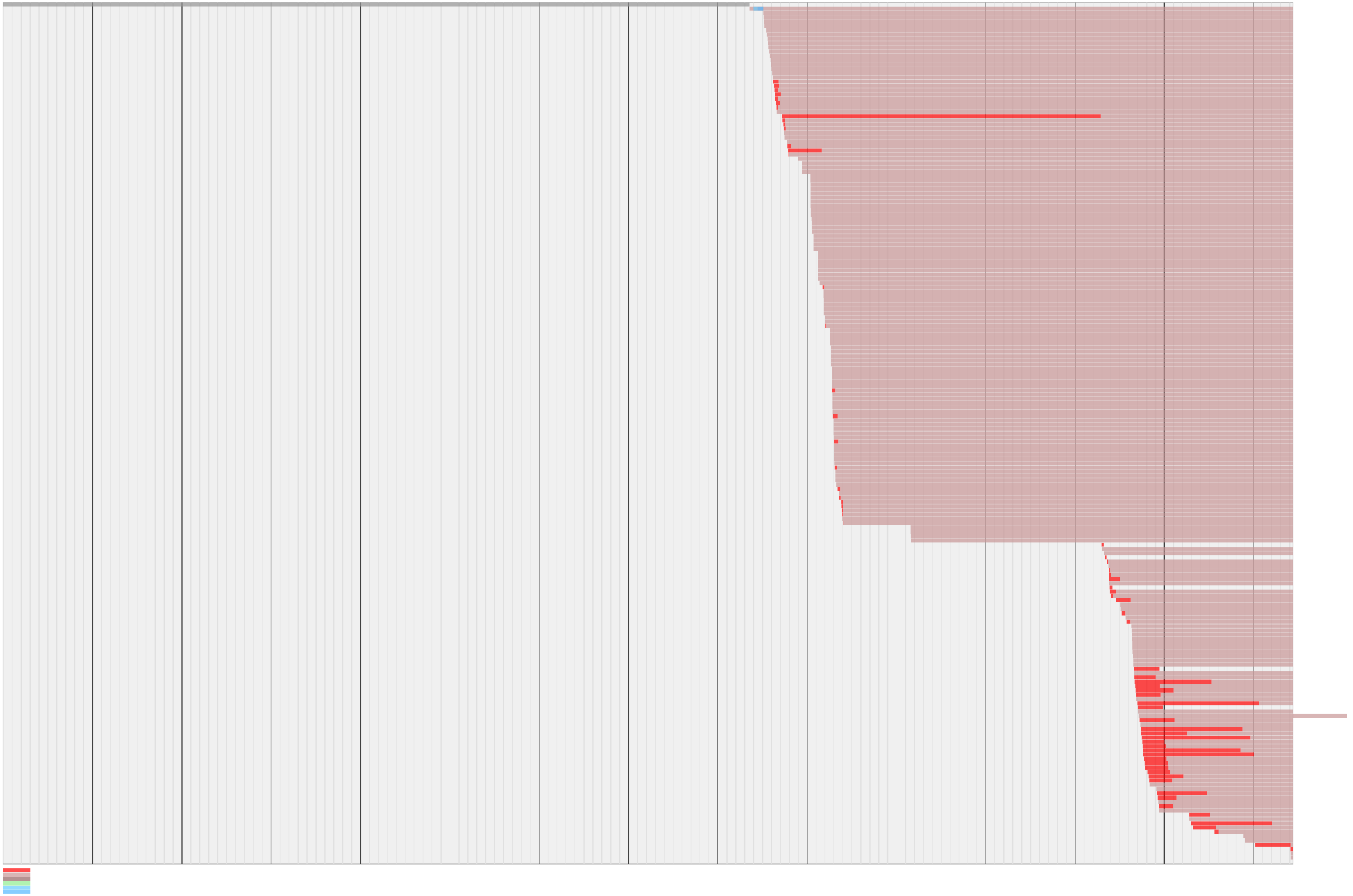
systemctl - Control the systemd system and service manager

```
# Unit information
systemctl [list-units] [--type=<type>] # Show known units
systemctl --failed                     # Show failed units

# Service control
systemctl start|stop|restart|reload <unit>
systemctl disable|enable <unit>        # En/Disable on startup
systemctl mask|unmask <unit>           # Completely disable
systemctl status <unit>                # Show unit status
systemctl help <unit>                  # Show unit help file

# Target control
systemctl graphical|multi-user|rescue|emergency
systemctl poweroff|reboot
systemctl get-default                   # Show default target
systemctl set-default <target>          # Set default target

systemctl daemon-reload                # Reload systemd conf
```

System startup analysis

```
> systemd-analyze time
Startup finished in 8.354s (kernel) + 6.083s (userspace) =
14.437s

> systemd-analyze blame
 3.566s systemd-udev-settle.service
 1.361s NetworkManager.service
 1.241s nmbd.service
 1.216s winbind.service
 [...]

> systemd-analyze critical-chain
 graphical.target @6.054s
 └─multi-user.target @6.054s
   └─smbd.service @5.660s +393ms
     └─nmbd.service @4.409s +1.241s
       └─basic.target @4.301s
        [...]
```

```
> systemd-analyze plot > startup.svg
```



journalctl

- *journald* Komponente des systemd packages
- Loggt syslog, kernel log, init log, sowie stdout/stderr aller services
- Leitet alles an syslog weiter (läuft parallel zu syslog)
- Logfiles in Binärformat

```
journalctl                          # Show full log
journalctl -b                       # Since last boot
journalctl -r                       # Newest entriest first
journalctl -u <unit>                # Filter by unit
journalctl --since <time term>      # By time
```

Timer

- *.timer* Unitfile um Services zeitgesteuert zu starten
- Alternative zu *cron* und *at*
- *Monotonic Timer* -> relativ zu einem Startereignis
- *Realtime Timer* -> zu einer bestimmten Kalenderzeit

```
systemctl enable <timer unit>          # Start timer
systemctl list-timers                  # Show all active timers
```

```
# systemd-tmpfiles-clean.timer
[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)

[Timer]
OnBootSec=15min                        # Start 15 min after boot
OnUnitActiveSec=1d                     # ... then daily
```

systemd und cgroups

- *cgroups* sind ein Kernel-Mechanismus zur Allokation von Ressourcen
- Jeder Service läuft in eigener cgroup
- *slices* sind Gruppen von Services, die gemeinsam verwaltet werden

```
systemd-cgls       # Show control groups recursively

systemd-cgtop      # Show control groups by resource usage
```

Service Isolation, Tuning und Security

cgroups erlaubt Isolation von Services (Security) sowie Feintuning von Systemressourcen.

Konfiguration im Unit-File:

```
[Service]
PrivateTmp=yes                  # Separate /tmp, /var/tmp
PrivateNetwork=yes              # Isolate network, lo only
ProtectSystem=full              # /usr, /etc read-only
ProtectHome=yes                 # /home empty
ReadOnlyDirectories=/var        # Set read-only directories
LimitNPROC=1                    # Prevent from forking

CPUQuota=20%                    # Set CPU Quota
MemoryLimit=1G                  # Set Memory Limit
BlockIOWeight=500               # Set IO weight (default 1000)
```