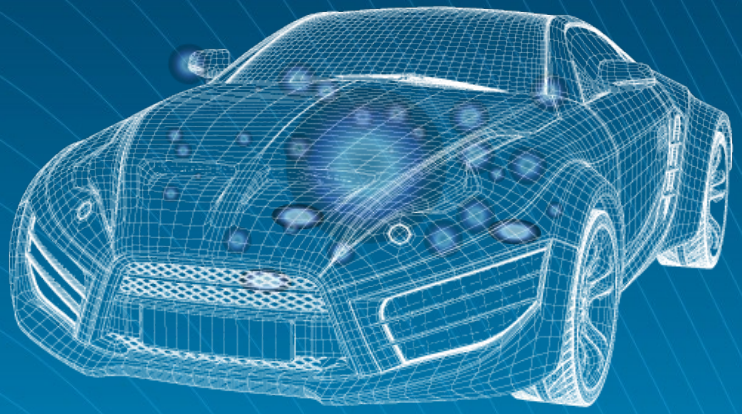
systemd, the modern Linux service and resource manager



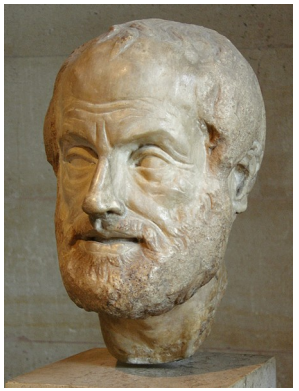
Alison Chaiken

Sept. 8, 2015

Mentor® Automotive

mentor.com/automotive

Philosophy of systemd



- *Extract duplicate functionality from daemons and move it to systemd core or kernel.*
- *Replace /etc scripts with declarative config files.*
- *Expose newer kernel APIs to userspace via a simple interface.*

systemd is:

- ***modular***;
- ***asynchronous*** and ***concurrent***;
- described by ***declarative*** sets of properties;
- bundled with analysis tools and ***tests;***
- supplied with a fully ***language-agnostic*** API.

init.d scripts ⇒ systemd unit properties

- Unit's action and parameters: ExecStart=
- Dependencies: Before=, After=, Requires=, Conflicts= and Wants=.
- Default dependencies:
  - Requires= and After= on basic.target;
  - Conflicts= and Before= on shutdown.target.
- Types of unit files: service, socket, device, mount, scope, slice, automount, swap, target, path, timer, snapshot

Example: Automotive Message Broker

From /lib/systemd/system/ambd.service:

```
# ambd systemd service unit file
[Unit]
Description=Automotive Message Broker
After=syslog.target
[Service]
Type=dbus
BusName=org.automotive.message.broker
ExecStart=/usr/bin/ambd
[Install]
WantedBy=multi-user.target
```

Example: Make the gdp-hmi-controller only start when weston is ready

From: Jonathan Maw <jonathan.maw@codethink.co.uk>

[. . .]

--- /dev/null

+++ b/recipes-demo-hmi/genivi-demo-platform-hmi/gdp-hmi-launcher2/StartLauncher.path

+[Unit]

+Description=GENIVI Start Launcher when weston is ready

+Requires=weston.service

+After=weston.service

+[Path]

+PathExists=/var/run/user/0/wayland-0

sysVinit runlevels ≈ systemd targets

- Targets are *synchronization points*.

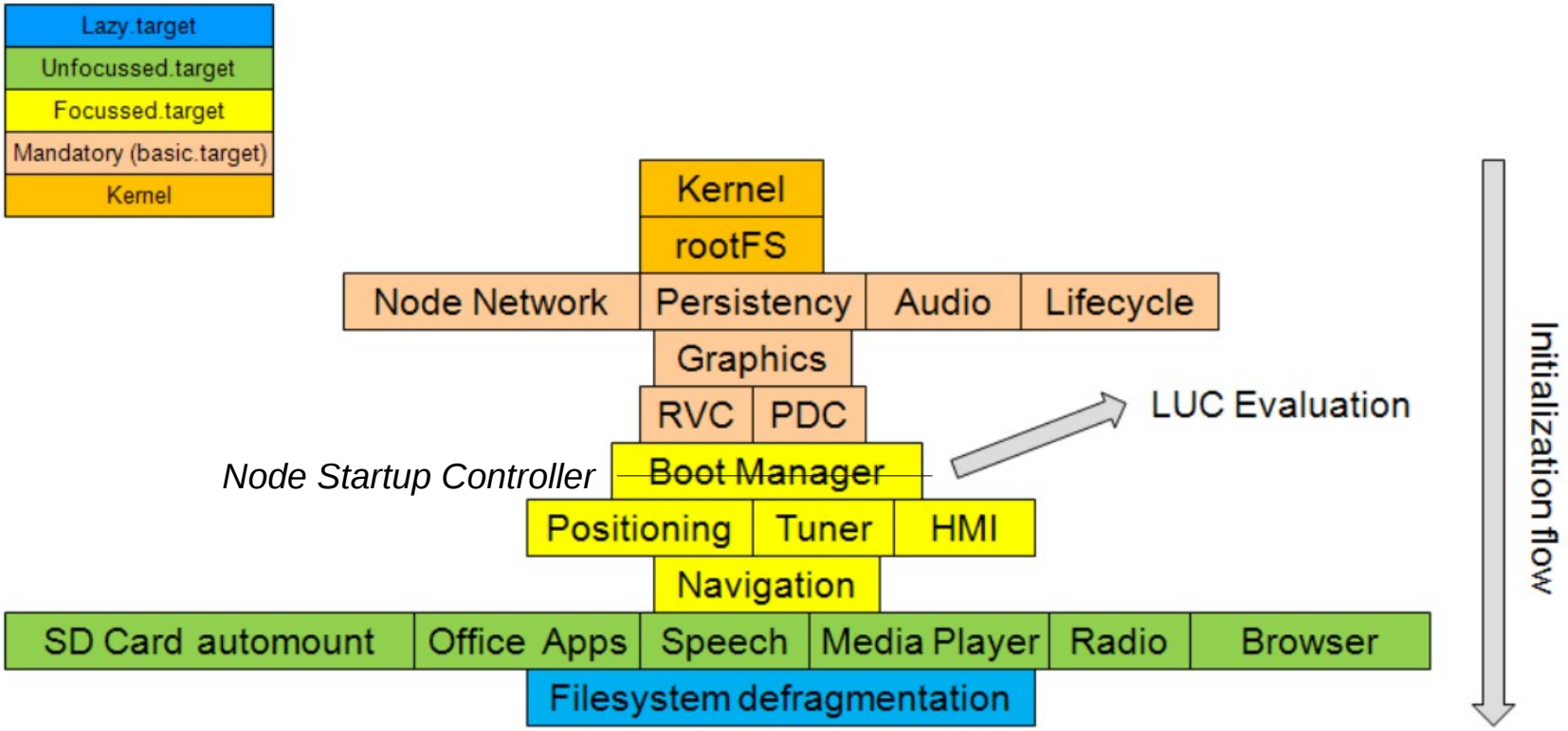
- Check */lib/systemd/system/runlevel?.target* symlinks:
  - multi-user.target (runlevel 3)
  - graphical.target (runlevel 5)

- Select boot-target :
  - via */etc/systemd/system/default.target* symlink;
  - appending number or *systemd.unit=<target>* to bootargs.

Illustration: systemd-bootchart and custom targets

Courtesy Marko Hoyer, ADIT.

Custom Targets in GENIVI's Lifecycle Management



LUC = Last User Context

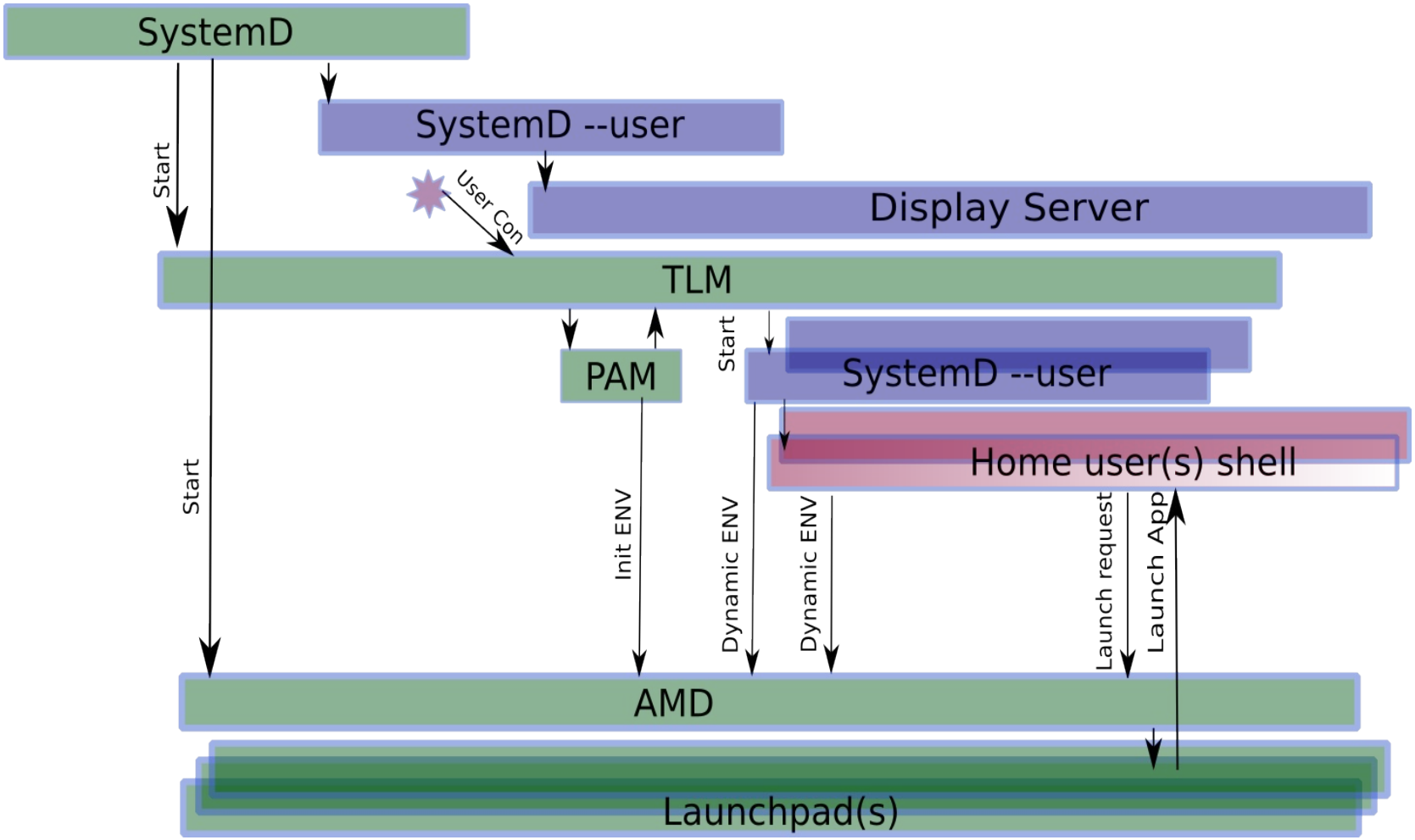
Hierarchy of unit files for system and user sessions

- Organized into *system* and *user* units
- */lib/systemd/system:* systemd upstream defaults for system-wide services
- */etc/systemd/system:* local customizations by *override* and *extension*
- */lib/systemd/user*: systemd's upstream defaults for per-user services
- $HOME/.local/share/systemd/user for user-installed units
- 'drop-ins' are run-time extensions

Illustration:Tizen Multi-user system

Source: Dominig ar Foll, “Tizen 3 IVI Architecture Multi User App FW”, FOSDEM 2014

Tizen Login Manager and User Sessions



Source: Dominig ar Foll, “Tizen 3 IVI Architecture Multi User App FW”, FOSDEM 2014

using the systemd journal

- Easily pushed to a remote.
- Can be cryptographically 'sealed'.
- Configurable max size and rotation.
- Simple log-reading tools are scriptable:

```
journalctl -xn
journalctl -p err
journalctl  -u cron
journalctl -o json-pretty
systemctl status
systemctl is-failed bluetooth
systemctl --failed
```

Both systemd journal and DLT:

- are modular and extensible.
- support pushing data to AF_INET socket.
- can share data with legacy syslog.
- natively output binary formats.

  → Linking systemd and DLT involves:

  - translating binary formats;
  - handshaking;
  - synchronization.
- Performed by remote?

simplifies access to kernel's security, resource and watchdog controls

- Including Capabilities, Watchdog, Cgroups and kdbus (still a 'coming attraction')

- Kernel features configurable via simple ASCII options in unit files.

- Encourages creation of system ***policies*** via unit templates.

granular resource control: systemd and cgroups

- cgroups: a kernel-level mechanism for allocating storage, memory, CPU and network.
- *slices* are groups of *daemons* whose resources are managed jointly.
- systemd *scopes* are resultant groups of *user* processes.
- Can set BlockIOWeight, IOSchedulingPriority, OOMScoreAdjust, CPUShares, MemoryLimit …

Tomorrow: “An introduction to Control Groups”

systemd and security:
granular encapsulation via kernel's *capabilities*

- CapabilityBoundingSet
- PrivateTmp, PrivateDevices, PrivateNetwork
- JoinNamespaces
- ProtectSystem (/usr and /etc), ProtectHome
- ReadOnlyDirectories, InaccessibleDirectories
- systemd-nspawn: systemd's native containers

systemd's watchdog support

- Provides simple configuration of soft or hard watchdogs.
- RuntimeWatchdogSec sets a timer for petting the dog.
- ShutdownWatchdogSec sets a timer to force reboot if shutdown hangs.

Summary

- Adaptation to systemd requires considerable thought.
- Uniform dependency language is attractive.
- Managing resources, security and and watchdogs is potentially simplified.
- Backwards compatibility eases transition, but project moves quickly.

Resources

- Man pages are part of systemd git repo.
- freedesktop.org: systemd mailing list archives and wiki
- Poettering's 0pointer.de blog
- At wayback machine: “Booting up” articles
- Neil Brown series at LWN
- Fedora's SysVinit to systemd cheatsheet
- Poettering's 'What's new' talk from FOSDEM 2015
- Josh Triplett's Debconf talk video
- Linux Action Show interviews with Mark Shuttleworth and Lennart Poettering
- A bunch of videos and slides linked at my website

resource utilization of systemd itself

- systemd-211 in Poky includes 17 packages = 8 MB.
- systemd-219 builds 90 MB of executables (not all needed).
- minimal build = systemd, udevd and journald.
- Memory (RSS) of fully featured build: ≈9 MB; minimum build ≈5 MB.
- Features added/removed via './configure'.