What I like about systemd

Ondřej Caletka

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

March 30, 2021

systemd

- modern replacement of traditional shell scripts
- inspired by macOS's `launchd`
- many *(often relevant)* negative feelings[1]
- adopted and used by *the silent majority*
- easier dependency management, faster system start
- many cooperating small utilities like `udev`

[1] https://nosystemd.org/

udev

- dynamic manager of device nodes in `/dev`
- listens to kernel events
- sets up proper permissions, including special permissions for on-console users (with help of `ConsoleKit`)
- ensures persistent device naming (including NICs)
  - which can be turned off for trivial cases like laptop
  - yet it comes very handy for complex cases like servers with multiple multi-port NICs
  - it **uncovers hidden problems** in computer firmwares no one was aware before

RIPE NCC

Deamon manager systemd

- controlled by `systemctl`
- vendor unit files in `/lib/systemd/system`
- override in `/etc/systemd/system/`
- plain INI-files describing *units* (services, sockets, targets, timers,...)
- new startup modes:
  - socket activation (sort-of `inetd`)
  - putting each process in its own control group (`systemd-cgls`)
  - support for simple services

RIPE NCC

Unit file editting

`systemctl cat <unit>`	list current unit file(s)
`systemctl edit <unit>`	open editor allowing to override parts of current unit file
`systemctl edit --full <unit>`	copy unit file to `/etc`, open editor
`systemctl edit --runtime …`	copy unit file to `/run`, changes will disappear after reboot
`systemctl daemon-reload`	reload all unit files

RIPE NCC

Do not clear screen after bootup

```
# systemctl cat getty@tty1
…
[Service]
# the VT is cleared by TTYVTDisallocate
…
TTYVTDisallocate=yes
…

# systemctl edit getty@tty1
[Service]
TTYVTDisallocate=no
```

RIPE NCC

Socket activation

listening socket `socket() -> bind() -> listen()`

connected socket `accept(<listening socket>)`

- very handy for all network (or unix socket) services
- elliminates inter-services dependency resolving – service is available, even when it is not (yet) running
- unit `<unit>.socket` creates a *listening socket*
- unit `<unit>.service` is started on incoming connection, *listening socket(s)* are shared as `fd=3,…`
- alternatively, service is run with every connection with *connected socket* on `fd=0` – `inetd` emulation
- service have to be adjusted to allow receiving sockets from caller

Instantiated services

- for services that have to be run in multiple instances
- unit like `getty@tty1.service` defined in `getty@.service`
- placeholders `%I`/`%i` represent readable/escaped instance name
- each instance can be separately overriden
- there are small *generators* creating run-time dependencies for particular instances (e. g. `openvpn.service` depends on `openvpn@vpn1.service`)

RIPE NCC

systemd-journald

- sort of a lightweight yet powerful **syslog replacement**
- collects logs from `/dev/log`, kernel, services' stdout and stderr
- efficient **binary format** with extra metadata, automatically and transparently rotated
- allows chaining another *syslogd* listening to `/run/systemd/journal/syslog`
- non-persistent logging to `/run/log/journal` by default
- no support for logging over network

RIPE NCC

journalctl in examples

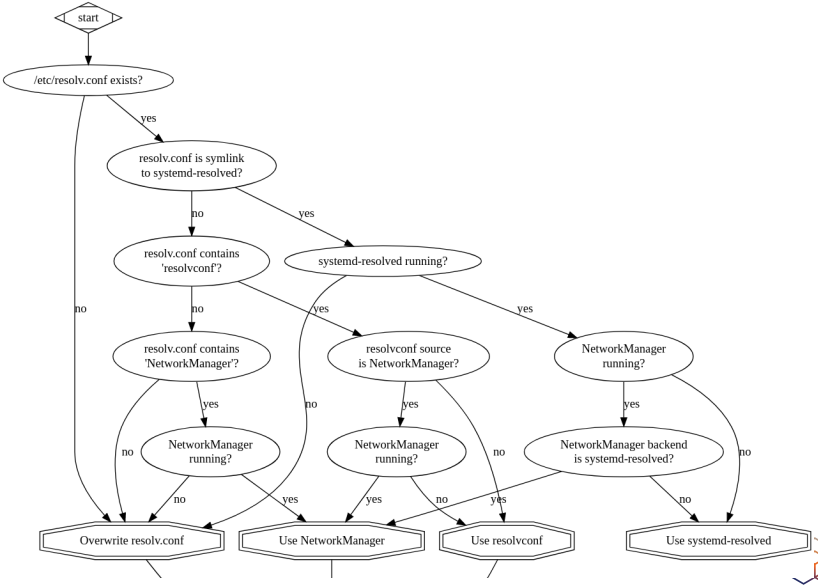
`journalctl`	all logs since the very beginning
`journalctl -e`	start from the last line
`journalctl -f -u <unit>`	follow output of particular unit
`journalctl --since today`	all logs since today
`journalctl /dev/sda1`	logs regarding a device

RIPE NCC

systemd-resolved

- userspace stub resolver implementation
- dbus, glibc and DNS API
- resolves local names from `/etc/hosts`, special name `_gateway`
- dotless domains resolved using LLMNR
- you can forward particular suffixes to a particular resolver
- some basic DNSSEC validation support

RIPE NCC



Source: Dave Anderson on Twitter

Thank you!

Ondřej Caletka
Ondrej.Caletka@ripe.net
https://**Ondřej.Caletka.nl**

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

The slides are already published on my website.