# Systemd for Embedded Linux
# –
# Challenges and Opportunities

**Embedded Linux Conference Europe**
**Düsseldorf, 2014-10-14**
**Michael Olbrich <m.olbrich@pengutronix.de>**

Pengutronix.

# Preconceptions about Systemd

- “It's only for desktop and server systems.”
- “It's too big.”
- “Everything is different.”
- “I don't need all this extra ...”
- “We have our own custom init system.”

# Anatomy of an Embedded System

## The Application!

# Anatomy of an Embedded System

- The Application

- There are actually multiple processes in the main application
- Login via ssh during development only
- Logging
- Network configuration
- Web server or other auxiliary servers
- Robustness / availability
- ...

# Anatomy of an Embedded System

- What you “sell” is “The Application”
- Nobody cares about the rest - until it breaks

➔ Use systemd to handle the ugly details

# Service Manager vs. Service Launcher

- System V init:
  - Fire and forget: just runs the init script
  - No global automated failure handling
  - “Magic” scripts

- Systemd:
  - Declarative service description
  - Less freedom than scripts
  - Systemd knows what should happen and can detect problems
  - Processes are monitored during the whole live time of a service

# Reliability / Availability

- Restart failed services
- Reboot when restarting doesn't work
- Better failure detection with per service watchdog
- Resource management

# Optional and On-Demand Services

Problem:

- The ssh server is only needed during development
- Starting an extra server changes the startup sequence

Solution:

- Systemd can listen on a port and start the server only when someone tries to connect
- Activation can depend e.g. on a special kernel command line option that is only used during development

# Logging with Journald

- Catch-all log: dmesg, syslog and service stdout/stderr
- Basic journal on a tmpfs
- Optional journal on a persistent file system
- Built-in size limit, so no logrotate needed

# Startup Notification

- Init systems need to know when a service is ready
- Use case:
  - Hard dependencies between services
  - Soft dependencies, e.g. delay things for smooth animations
- Implementation:
  - Classic “deamonizing”
    - Many steps needed
    - Hard to get it right
  - With systemd:
    - `sd_notify(0, "READY=1");`

# Security

- It's easy to start services as a different user
- Integrated namespace handling
- Integration with several LSMs (SELinux, AppArmor, ...)
- ...

# Network Configuration with Networkd?

- It works well for fixed configurations
- No support for wireless networks
- Configuration changes at runtime are not really supported

- It's rather new - Let's see what the future brings

# Starting with Systemd on Embedded Linux

- Choose an embedded Linux distribution that provides a recent version of systemd
- Read the systemd README! It contains all mandatory and optional requirements
- Kernel on embedded systems a often missing mandatory features
- Systemd 'feels' different. Take the time to familiarize yourself with it

# Future

- Improved network configuration?
  - Systemd-networkd is quite new. Where will it go?
- Kdbus?
  - Possibilities for faster interprocess communication?
  - Implications for the required Linux kernel version?
- “Stateless Systems, Factory Reset, Golden Master Systems and systemd”
  - How do “stateless systems and embedded systems with read-only rootfs fit together?
  - Can we reuse “golden master systems” for in-system updates in embedded systems?

# Questions?