We Need to Talk About Systemd

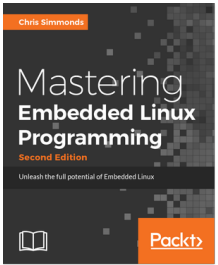
Boot Time Optimization for the new init daemon

Chris Simmonds

Embedded Linux Conference Europe 2019

License

About Chris Simmonds



- Consultant and trainer
- Author of *Mastering Embedded Linux Programming*
- Working with embedded Linux since 1999
- Android since 2009
- Speaker at many conferences and workshops

"Looking after the Inner Penguin" blog at `http://2net.co.uk/`

@2net_software

`https://uk.linkedin.com/in/chrisdsimmonds/`

Previously...

- ELC-E 2017: A pragmatic guide to boot-time optimization
  https://elinux.org/images/6/64/Chris-simmonds-boot-time-elce-2017_0.pdf
- Focused on optimizing bootloader and kernel boot

Previously...

- ELC-E 2017: A pragmatic guide to boot-time optimization
  https://elinux.org/images/6/64/Chris-simmonds-boot-time-elce-2017_0.pdf
- Focused on optimizing bootloader and kernel boot
- Bypassed init daemon:

```
init=/usr/bin/run-qtdemo.sh
```

Previously...

- ELC-E 2017: A pragmatic guide to boot-time optimization
  https://elinux.org/images/6/64/Chris-simmonds-boot-time-elce-2017_0.pdf
- Focused on optimizing bootloader and kernel boot
- Bypassed init daemon:

```
init=/usr/bin/run-qtdemo.sh
```

- This works in some cases, but lacks elegance (amongst other things)
- Perhaps there is a better way

Agenda

- What does systemd do?

Agenda

- What does systemd do?
- Systemd 101

Agenda

- What does systemd do?
- Systemd 101
- Optimizing boot

Agenda

- What does systemd do?
- Systemd 101
- Optimizing boot
- AOB

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1
- At boot, init has to

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1
- At boot, init has to
  - start system daemons

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1
- At boot, init has to
  - start system daemons
  - configure stuff

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1
- At boot, init has to
  - start system daemons
  - configure stuff
  - restart daemons that have stopped

What is an init daemon?

- **init** is launched by the kernel after it has booted
  - Hence, PID 1
- At boot, init has to
  - start system daemons
  - configure stuff
  - restart daemons that have stopped
- Then it sits in the background ...
  - being a parent of last resort for orphans
  - starting daemons that have stopped
  - reaping zombies

Init daemons for embedded use cases

Embedded Linux systems generally use one of three init daemons

Metric	BusyBox init	System V init	systemd
Complexity	Low	Medium	High
Boot-up speed	Fast	Slow	Medium (*)
Required shell	ash	ash or bash	None
Number of executables	0	4	50
libc	Any	Any	glibc
Size (MiB)	0	0.1	34

(*) but we hope to improve that by the end of this presentation

Systemd is not *just* an init daemon

It's more of a way of life: aims to be a general purpose system manager

Here are the main components that are relevant to this discussion

Component	Description
systemd	The init daemon
journald	Event logger
logind	User login manager
udevd	Device manager and kernel events
networkd	Configures network interfaces
timesyncd	Sync local time (e.g. via NTP)
resolved	DNS name resolver

What are the advantages of systemd?

- Explicit dependencies between services

What are the advantages of systemd?

- Explicit dependencies between services
- Parallel init - faster boot

What are the advantages of systemd?

- Explicit dependencies between services
- Parallel init - faster boot
- No more shell scripts (which are definitely slow)

What are the advantages of systemd?

- Explicit dependencies between services
- Parallel init - faster boot
- No more shell scripts (which are definitely slow)
- Per-daemon resource control

What are the advantages of systemd?

- Explicit dependencies between services
- Parallel init - faster boot
- No more shell scripts (which are definitely slow)
- Per-daemon resource control
- Per-daemon watchdogs

Units, services and targets

- **Unit**: describes a target, a service, and several other things
- **Service**: a daemon that can be started and stopped
- **Target**: a group of services, similar to a Sys V runlevel

Units

- Systemd searches for units working from most specific to most general configuration
  - `/etc/systemd/system`: Local configuration
  - `/run/systemd/system`: Runtime configuration
  - `/lib/systemd/system`: Distribution-wide configuration
- To override a unit, just place a unit with the same name earlier in the sequence
- To disable a unit, replace it with an empty file or a link to `/dev/null`

Units

- All units have a `[Unit]` section
- Contains a description, reference to documentation and dependencies on other units
- Example: the Unit section from `/lib/systemd/system/dbus.service`

```
[Unit]
Description=D-Bus System Message Bus
Documentation=man:dbus-daemon(1)
Requires=dbus.socket
[...]
```

Unit dependencies

- **Requires**: a list of units this depends on, which should be started before this unit is started
- **Wants**: a weaker form of Requires: this unit is not stopped if any in the list fail to start
- **Conflicts**: a negative dependency: the units listed are stopped when this one is started and, conversely, if one of them is started, this one is stopped

Order: Before and After

- These keywords determine the order that units are started
- **Before**: This unit should be started before the units listed
- **After**: This unit should be started after the units listed
- Example: start web server *after* the network target

```
[Unit]
Description=Lighttpd Web Server
After=network.target
[...]
```

- Without Before or After, units are started in no particular order

Service

- A service is a unit that controls a daemon
- Name ends in `.service`
- Has a `[Service]` section
- Example, `lighttpd.service`

```
[Unit]
Description=Lighttpd Web Server
After=network.target
[Service]
ExecStart=/usr/sbin/lighttpd -f /etc/lighttpd/lighttpd.conf -D
ExecReload=/bin/kill -HUP $MAINPID
```

Target

- A Target is a Unit that lists dependencies on other Targets
- Name ends in `.target`
- Example, `/lib/systemd/system/multi-user.target`

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

The default target

- At boot, systemd starts `default.target`
- Usually a symbolic link to the target desired
- Example

```
/etc/systemd/system/default.target ->
       /lib/systemd/system/multi-user.target
```

- Default target may be overridden on kernel command line:
  `system.unit=<new target>`

Reverse dependencies: WantedBy

- **Requires** and **Wants** create **outgoing** dependencies
  - Used, for example, to create a dependency tree of targets
- Other types of Unit are started by **incoming** dependencies
- Incoming dependencies are created by **WantedBy**
- Example: a server that is started by multi-user.target:

```
[Unit]
Description=Simple server
[Service]
ExecStart=/usr/bin/simpleserver
[Install]
WantedBy=multi-user.target
```

The Install section

- Incoming dependencies are expressed by links
- A Target can have a subdirectory named `<target name>.wants`
- Contains symbolic links to the Units that should be started
- Example: installing simpleserver creates this link

```
/etc/systemd/system/multi-user.target.wants/simpleserver.service ->
       /lib/systemd/system/simpleserver.service
```

systemctl

- `systemctl` is a command line interface for systemd
- Useful commands
  - `start [unit]`: start a unit
  - `stop [unit]`: stop a unit
  - `enable[unit]`: install the unit, creating the wants link
  - `disable[unit]`: uninstall the unit
  - `status [unit]`: show status of a unit
  - `get-default`: show default target
  - `list-dependencies`: list dependency tree

Reducing boot time

- Boot time = from power on to running the critical app

Reducing boot time

- Boot time = from power on to running the critical app
- A generic system image is designed to cater for all likely circumstances

Reducing boot time

- Boot time = from power on to running the critical app
- A generic system image is designed to cater for all likely circumstances
- To reduce boot time you need to make it less generic

Reducing boot time

- Boot time = from power on to running the critical app
- A generic system image is designed to cater for all likely circumstances
- To reduce boot time you need to make it less generic
- There are two ways to do it
  - Leave out tasks that you don't need

Reducing boot time

- Boot time = from power on to running the critical app
- A generic system image is designed to cater for all likely circumstances
- To reduce boot time you need to make it less generic
- There are two ways to do it
  - Leave out tasks that you don't need
  - Change the order of tasks

Measuring systemd boot time

`systemd-analyze` is a useful tool for measuring systemd boot time

Summary of boot time

```
systemd-analyse
```

List units in order of start-up time

```
systemd-analyse blame
```

Print a tree of the time-critical chain of units

```
systemd-analyze critical-chain
```

First attempt

- PocketBeagle running Debain Stretch

systemd-analyze 1/3

```
systemd-analyze
```

```
Startup finished in 18.722s (kernel) + 47.875s (userspace) = 1min 6.597s
```

systemd-analyze 2/3

```
systemd-analyze --no-pager blame
```

```
42.423s generic-board-startup.service
19.787s dev-mmcblk0p1.device
 5.840s networking.service
 4.365s loadcpufreq.service
 3.414s systemd-udev-trigger.service
 2.986s apache2.service
 2.744s connman.service
 2.621s udhcpd.service
 2.383s systemd-logind.service
 2.372s avahi-daemon.service
 2.314s ti-ipc-dra7xx.service
 2.276s rc_battery_monitor.service
 2.221s robotcontrol.service
 2.061s keyboard-setup.service
 2.003s pppd-dns.service
 1.939s ssh.service
 1.475s rsyslog.service
[...]
```

systemd-analyze 3/3

```
systemd-analyze --no-pager critical-chain
```

```
The time after the unit is active or started is printed after the "@" character.
The time the unit takes to start is printed after the "+" character.

graphical.target @47.683s
 -multi-user.target @47.678s
   -getty.target @47.450s
     -serial-getty@ttyGS0.service @47.433s
       -dev-ttyGS0.device @47.411s
```

Note `serial-getty@ttyGS0.service`. There is no ttyGS0

2nd attempt

- Slimmed down systemd
- Change default target from graphical to multiuser
- Remove `serial-getty@ttyGS0.service`
- Remove other services, including robotcontrol, bluetooth, apache2

```
systemd-analyze
```

```
Startup finished in 16.880s (kernel) + 12.375s (userspace) = 29.255s
```

Boot time reduced by 35s

Still too long, but it's a start!

Other useful systemd features

- Watchdog
- Resource limits

Watchdog

- A watchdog will trigger a service to restart on timeout
- Example: service restarts if no watchdog keepalive is sent within 30s. If it restarts 4 times in 5 minutes it will force a reboot

```
[Unit]
 ...
[Service]
WatchdogSec=30s
Restart=on-watchdog
StartLimitInterval=5min
StartLimitBurst=4
StartLimitAction=reboot-force
```

Resource limits

- Limits system resources that a program can consume
- *systemd.resource-control(5)* for details
- Example, service with CPU quota 25% and memory limit 4MB

```
[Unit]
 ...
[Service]
ExecStart=/usr/bin/simpleserver
CPUQuota=20%
MemoryAccounting=true
MemoryMax=4096K
```

Limits are enforced using Linux control groups, aka **cgroups**

- Questions?