systemd: the modern init system you will learn to love

Alison Chaiken
alison@she-devel.com
http://she-devel.com
Jan. 7, 2015

Text in blue is hyperlinked.

On-the-fly audience exercises.
Topics

- Motivation
- Concepts
- Usage
- Controversy
Quiz:
what is the most widely used Linux init system?
Aversion to change

sysVinit

systemd

Never go back!
systemd is . . .

- already the basis of Fedora, RHEL, CentOS, OpenSUSE and much embedded.
- soon to be the basis of Debian and Ubuntu.
- praised by Jordan Hubbard of FreeBSD.
- after OpenStack and Docker, the most discussed new Linux feature.
Concepts
systemd is:

- *modular*;
- *asynchronous* and *concurrent*;
- described by *declarative* sets of properties;
- bundled with analysis tools and *tests*;
- features a fully *language-agnostic* API.
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.
One daemon to rule them all

**xinetc**: a daemon to lazily launch **internet services** when activity is detected on an **AF_INET** socket

**systemd**: a daemon to lazily launch **any system service** when activity is detected on an **AF_UNIX** socket (oversimplification)
which services are started by sysVinit?

Try: 'ls/etc/init.d'
Which daemons started by systemd directly?

Try: `ls /lib/systemd/system/*.service`

Try: `systemctl list-sockets`
SysV already has a big service manager: bash

[user@localhost]$ wc -l /sbin/init
  64

[user@localhost]$ wc -l /bin/bash
  4154

[user@localhost]$ wc -l /lib/systemd/systemd
  5944
Side-by-side comparison

[user@localhost]$ wc  /etc/rc5.d/S16rsyslog
  126 380 2796 /etc/rc5.d/S16rsyslog

[user@localhost]$ wc  /lib/systemd/system/rsyslog.service
  15 16 290 /lib/systemd/system/rsyslog.service
Major Differences with SysVInit

- clean environment
- socket-based activation
[Socket activation demo with cups and ncat]
init.d scripts ⇒ systemd units

- Unit's action and parameters: ExecStart=
  - Can start a daemon, a bash script ...
- Dependencies: Before=, After=, Requires=, Conflicts= and Wants=.
- Default dependencies:
  - Requires= and After= on basic.target;
  - Conflicts= and Before= on shutdown.target.
- Types of units: service, socket, device, mount, scope, slice, automount, swap, target, path, timer, snapshot
Modularity can produce complexity
Sequence of targets on a typical system

$ ls -l /lib/systemd/system/default.target
   /lib/systemd/system/default.target -> graphical.target

$ cat /lib/systemd/system/graphical.target
   After=multi-user.target

$ cat /lib/systemd/system/multi-user.target
   After=basic.target

$ cat /lib/systemd/system/basic.target
   After=sysinit.target sockets.target timers.target paths.target slices.target
Understanding dependencies

Try:

```bash
systemctl list-dependencies basic.target
systemctl list-dependencies --after tmp.mount
```
Usage
systemd is easy to use

- systemd utilities:
  - Try: `apropos systemd | grep ctl`
- All-ASCII configuration files: no hidden “registry”.
- Customization is by **overriding** default files.
- Many choices are controllable via symlinks.
- Bash-completion by default.
- Backwards compatibility with SysVinit
Hierarchy of unit files for system and user sessions

- `/lib/systemd/system`: systemd upstream defaults
- `/etc/system/system`: local customizations by `override` and `extension`
- `/usr/lib/systemd/user/`: distro's unit files for user sessions
- `$HOME/.local/share/systemd/user/` for user-installed units
- 'drop-ins' are run-time extensions
Override your defaults!

- Replace a unit in `/lib` by creating one of the same name in `/etc`.

- Add services by symlinking them into
  `/etc/systemd/system/default.target.wants`.

- Best practice: do not change the files in `/lib/systemd`.

- `systemd-delta` to see customizations.

- Read in-use unit with `systemctl cat`.
Controversy
using the systemd journal

• Binary format is (rightfully) controversial.
• Run “addgroup $USER systemd-journal” for access.
• Log-reading tools are simple:
  – `journalctl -xn`
  – `journalctl -p err`
  – `journalctl /usr/sbin/cron`
  – `systemctl status`
  – `systemctl is-failed bluetooth`
  – `systemctl --failed`
<table>
<thead>
<tr>
<th>Old way</th>
<th>New way</th>
<th>History</th>
</tr>
</thead>
<tbody>
<tr>
<td>static /dev, then devfs</td>
<td>udev</td>
<td>“The return of devfs”</td>
</tr>
<tr>
<td>getrlimit, setrlimit</td>
<td>cgroups</td>
<td>“The evolution of control groups”</td>
</tr>
<tr>
<td>KDE3 and GNOME2</td>
<td>KDE4 and GNOME3</td>
<td>KDE and GNOME</td>
</tr>
<tr>
<td>sysVinit</td>
<td>systemd</td>
<td>in progress</td>
</tr>
<tr>
<td>X11 client-server model</td>
<td>Wayland compositor</td>
<td>in progress</td>
</tr>
</tbody>
</table>
Crux of the problem: Dave Neary

“There is no freedesktop.org process for proposing standards, identifying those which are proposals and those which are de facto implemented, and perhaps more importantly, there is no process for building consensus around a specification . . . .”

(comment regarding GNOME3)
Summary

• Systemd has:
  – a superior design;
  – tight integration with the Linux kernel;
  – a vibrant developer community.

• Rants against systemd are largely FUD.

• Control over userspace has migrated:
  – away from distros;
  – toward kernel and freedesktop.org.

• Most users will not notice.

• The transition from X11 to Wayland will break more.
Thanks

- Mentor Graphics for sending me to Germany to hack on systemd.
- Kevin Dankwardt for teaching me about LWN and cscope.
- Vladimir Pantelic, Tom Gundersen and Lennart Poettering for corrections (without implied 'ack').
- Bill Ward and Jym Dyer for use of their images.
Everything will be okay.
Resources

- Man pages are part of systemd git repo.
- freedesktop.org: systemd mailing list archives and wiki
- At Poettering's 0pointer.de blog
- At wayback machine: “Booting up” articles
- Neil Brown series at LWN
- Fedora's SysVinit to systemd cheatsheet
- Steve Smethurst's Hacker Public Radio episode
- Josh Triplett's Debconf talk video
- Carla Schroeder's linux.com tutorial
Special topics
tight integration: **systemd and cgroups**

- cgroups are a kernel-level mechanism for allocating resources like storage, memory, CPU and network
- Userspace configures cgroups through cgroupfs
- `[user@localhost]$ sudo mount | grep cgroup`
  
  cgroup on /sys/fs/cgroup/cpuset type cgroup
  cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup
  cgroup on /sys/fs/cgroup/blkio type cgroup
  
  [ . . . ]

- BSDs do not have cgroups.
- Demo: `sudo systemd-cgls; sudo systemd-cgtop`
tight integration: systemd and udev

- udev is a daemon that handles kernel events related to changes in device status.

- udev was (controversially) merged into the systemd project.

- Like cgroups, udev is tightly integrated into the Linux kernel.

- Related imminent improvement: kdbus
**systemd and security**: granular encapsulation

- PrivateTmp, PrivateDevices, PrivateNetwork
- JoinNamespaces
- ProtectSystem (/usr and /etc), ProtectHome
- ReadOnlyDirectories, InaccessibleDirectories
- systemd-nspawn: systemd's native containers
- Easy configuration of kernel's capability properties
systemd in embedded systems

- systemd is widely adopted in embedded systems because
  - proper allocation of resources is critical;
  - fastboot is required;
  - customization of boot sequence is common.
- Lack of backward compatibility for older kernels (due to firmware loading) is a pain point.
- Embedded use cases are not always understood by systemd devs.
systemd and outside projects: CoreOS

- **networkd** was initially contributed by CoreOS developers.

- CoreOS's **fleet** “tool that presents your entire cluster as a single init system” is based on systemd.
  - Spin up new containers due to events on sockets.

- CoreOS devs are outside systemd inner circle.

- systemd has many patches from Arch, Intel, Debian . . .
developing systemd

- git clone git://anongit.freedesktop.org/systemd/systemd
- systemd-devel list: submit patches or ask questions
- Impressive and featureful utility library in `src/shared/
  
  #define streq(a,b) (strcmp((a),(b)) == 0)
  
  #define strneq(a, b, n) (strncmp((a), (b), (n)) == 0)
  
  #define strcaseeq(a,b) (strcasecmp((a),(b)) == 0)
  
  #define strncaseeq(a, b, n) (strncasecmp((a), (b), (n)) == 0)

- Complex but automated build system with many dependencies.
- 'Plumbing' dev tools in `/lib/systemd`, 'porcelain' tools in /bin
  
  find /lib/systemd -executable -type f
Leftover Materials
sysVinit runlevels ≈ systemd targets

- Check `/lib/systemd/system/runlevel?.target` symlinks:
  - `multi-user.target.wants` (runlevel 3 == text session)
  - `graphical.target.wants` (runlevel 5 == graphical session)

- **Select boot-target**: 
  - via `/etc/systemd/system/default.target` symlink;
  - appending number ('3' or '5') or `systemd.unit=<target>` to kernel cmdline;

- Change current target by
  - using `runlevel` (or `telinit`) command;
  - or `systemctl isolate multi-user.target`
Extensions: drop-ins

Try: systemd-delta

Try: systemctl cat <list from 1st command>
Customizing your installation

- *Replace* a unit in `/lib` (upstream) by creating one of the same name in `/etc` (local changes).

- *Add* services to boot by symlinking them into `/etc/systemd/system/default.target.wants`.

- *Best practice*: do not change the files in `/lib/systemd`
Example: set display manager

[user@localhost ~]$ ls -l `locate display-manager.service`
lrwxrwxrwx. 1 root root 35 Dec 11  2013
/etc/systemd/system/display-manager.service ->
/usr/lib/systemd/system/gdm.service

[user@localhost ~]$ cat /usr/lib/systemd/system/gdm.service
[Unit]
Description=GNOME Display Manager
[ . . . ]
[Install]
Alias=display-manager.service

or

WantedBy=graphical.target
sysinit, sockets and multi-user are composite targets

```bash
>$ ls /lib/systemd/system/multi-user.target.wants/
dbus.service@ systemd-ask-password-wall.path@ systemd-update-utmp-runlevel.service@ getty.target@

>$ ls /lib/systemd/system/sockets.target.wants:
dbus.socket@ systemd-shutdownd.socket@
systemd-initctl.socket@ systemd-udevd-control.socket@

>$ ls /lib/systemd/system/sysinit.target.wants:
cryptsetup.target@ systemd-journald.service@
debian-fixup.service@ systemd-journal-flush.service@
```

Symlinks replace lines of conditional code in SysVinit scripts.
Example: change the default target

[alison@localhost ~]$ ls /etc/systemd/system/default.target /etc/systemd/system/default.default.target -> /lib/systemd/system/graphical.target

[alison@localhost ~]$ sudo rm /etc/systemd/system/default.target
[alison@localhost ~]$ sudo ln -s /lib/systemd/system/multi-user.target /etc/systemd/system/default.target

[alison@localhost ~]$ ~/bin/systemd-delta
[ . . . ]
[REDIRECTED] /etc/systemd/system/default.target →
   /usr/lib/systemd/system/default.default.target
Misconceptions

• systemd is more complex than sysVinit.

• systemd is full of binary configuration files.

• The system log is now unreadable! And liable to corruption!

• {Fedora/GNOME/RedHat/Poettering} are trying to take over all of Linux.
problems

• systemd is modular, but:
  - interopability with other SW may be inadequately tested.

• Potentially rocky piecemeal transition by distros.
  - e.g., Debian installer doesn't warn about a separate /usr partition.

• Merciless deprecation of features (firmware loading, readahead . . . ).

• Frequent releases, not particularly stable.
Greg K-H: “Tightly-coupled components”

Greg Kroah-Hartman originally shared:
Here's the summary that people seem to be missing these days:

"There are a number of folk in the Linux ecosystem pushing for a small core of tightly coupled components to make the core of a modern Linux distro. The idea is that this "core distro" can evolve in sync with the kernel, and generally move fast. This is both good for the overall platform and very hard to implement for the "universal" distros."

I touched on this a week or so when I did the FoodFight podcast interview, but I don't think that people really understand what is happening here, and why it is happening.

Given the recent flames on the Gentoo mailing lists about how "horrible" it is that someone could even consider using an initrd to boot a system that has a separate /usr partition, and the weird movements by some Gentoo developers to deny that there really is a problem at all that is being solved by this type of work, I seriously wonder how much longer a "general" distribution such as Gentoo or Debian can keep up the charade of trying to provide all options for all users.

I just don't think it can be done well, sorry, which is why I strongly recommend tightly-coupled distros for desktops for anyone (like
Taxonomy of systemd dependencies

Requires, RequiresOverridable, Requisite, RequisiteOverridable, Wants, BindsTo, PartOf, Conflicts, Before, After, OnFailure PropagateReloadsTo, ReloadPropagateFrom,