systemd, the next-generation Linux system manager

LISA15
Nov. 9, 2015
Alison Chaiken
alison@she-devel.com

Latest version with fixes at http://she-devel.com/LISA15/LISA15_systemd.pdf

Topics

Stuart Chalmers

- Introduction: set up test environment.
- Basic concepts and tools
- Deeper dive into units, services and targets
- Dependencies and service activation
- Security and resource controls
- Performance tuning and failure analysis

Key to examples

- This font is for regular explanatory text and comments.
- Blue font is for hyperlinks.
- echo “green font for code snippets”
  - Some are OK on localhost, others only in container or VM!

Quiz!

1 What is the most-deployed Linux init system, by number of devices?
   a systemd;
   b sysVinit;
   c upstart;
   d other.
2 systemd exits shortly after userspace comes up. (T/F)
3 systemd runs as
   a one giant application;
   b many threads of execution;
   c a collection of processes;
   d a virtual machine.

Quiz, p. 2

1 The license of systemd is:
   a GPLv2;
   b GPLv3;
   c permissive;
   d proprietary.
2 systemd runs on Linux as well as BSD and MacOS (T/F).
3 systemd's first distro release was:
   a Fedora in 2011;
   b Debian in 2014;
   c RHEL in 2015.

Basic Concepts

Philosophy of systemd

Extract duplicate functionality from daemons and move it to systemd core or kernel.

Replace init.d scripts with declarative config files.

Expose newer kernel APIs to userspace via a simple interface.

Control behavior of applications via unit files rather than with code changes.

systemd is:

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

One daemon to rule them all

xinetd: 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)

How to RTFM Most Effectively

- Get the source:

  git clone git@github.com:systemd/systemd.git

- Provides a single grep-able directory with all man pages.
- As a last resort, grep the source to find the origin of an error message.
- The catch: must upload SSH key to github to clone from there.

Setup Test Environment

Exercise 0: Install a container or VM in which to test systemd

Either:

- boot up your favorite Linux container or VM;
- *or* follow instructions to create a Debian or Fedora container;
- *or* copy the Debian or Fedora container on the shared USB stick
- *or* bring a device (e.g. RPi) on which to run Linux.

Any systemd installation >= 208 should work fine:

```
ps -p 1; systemctl --version
```

Configure container or VM for easy testing

- Create a regular user (not root) and add to /etc/sudoers.
- Add the user to the systemd-journal group.
- If possible, install cups and nmap in the container/VM/device or on localhost.
- If possible, install graphviz on localhost.

(optional) systemd-nspawn lightning course

- systemd-nspawn manages systemd's native container type
- Basically a namespaced chroot that reuses host's kernel.
- Start console session for container:
  - sudo systemd-nspawn -D </path/to/container/>
- 'Boot' the container:
  - sudo systemd-nspawn -bD </path/to/container>
- Monitor and control from host:
  - machinectl list and machinectl status (not available in older versions)
  - sudo machinectl reboot <container name>
  - machinectl list-images

Preliminaries

Get started with systemctl and journalctl

- addgroup $USER systemd-journal for access.
- systemctl status; systemctl status ssh
- journalctl -xn; journalctl -u ssh
- systemctl --failed; journalctl -p err
- sudo systemctl start cups *(or* restart)
- systemctl show ntp
- sudo systemctl poweroff *or* sudo systemctl reboot

Units and Services

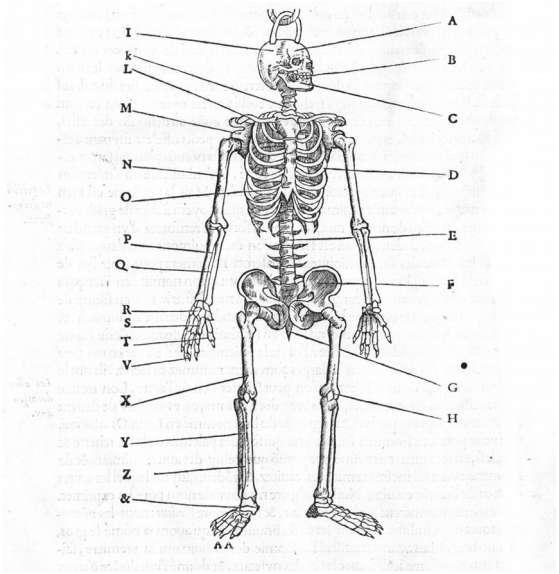
Complexity arising from many similar small units

Courtesy Bill Ward

init.d scripts ⇒⇒ systemd units

- 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
- See 'man systemd.unit' or freedesktop.org

Anatomy of a Unit File



- *ExecStart* can point to *any* executable, including a shell script.
- Unit files typically include a [Unit] section and a [Service] section.
- An [Install] section determines the *target* with which a unit is associated.
- Try: systemctl cat ssh or systemctl show ssh

Precedence of unit files

- */lib/systemd/system/:* upstream defaults for system-wide services
- */etc/systemd/system/:* local customizations by *override* and *extension*
- 'drop-ins' are extension fragments akin to those in /etc/yum.repos.d/ or /etc/apt.conf.d/.
- Try: *systemd-delta*

Exercise 1: create a HelloWorld service

1 Create HelloWorld.service in your container that prints “Hello World” into the systemd journal.

2 Situate it in the filesystem where systemd can find it.

3 Start the service using systemctl.

4 Check the status of your service. Where has “Hello, world” output appeared?

Solution: simple HelloWorld.service

1 With a text editor, create helloworld.sh:

```
#!/bin/bash
echo “Hello World!”
```

2 Copy the script into your container's filesystem:

```
chmod +x helloworld.sh
cp helloworld.sh /var/lib/machines/debian/usr/local/bin/
```

3 With a text editor, create HelloWorld.service:

```
[Unit]
Description=Hello World Service
Documentation=
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

4 Copy the unit file into the container's filesystem:

```
cp HelloWorld.service /var/lib/machines/debian/etc/systemd/system/
```

(or, on your localhost, `cp HelloWorld.service /etc/systemd/system/`)

5 Boot the container, then load and run the unit:

```
sudo systemd-nspawn -bD /var/lib/machines/debian
[inside container] sudo systemctl start HelloWorld
[inside container] systemctl status HelloWorld
[inside container]journalctl -u HelloWorld
```


Targets
vs.
Runlevels


sysVinit runlevels ≈ systemd targets

- Targets are *synchronization points*.
- Check */lib/systemd/system/runlevel?.target* symlinks:
  multi-user.target (runlevel 3 == text session)
  graphical.target (runlevel 5 == graphical session)
- Select boot-target :
  - via */etc/systemd/system/default.target* symlink;
  - by appending *systemd.unit=<target>* to bootargs.
- Helpful diagram: “man 7 bootup”

Target Basics

- Service *S* will be started as part of Target *T* iff S.service file is symlinked in the directory /etc/systemd/system/*T*.wants.
- If *S*'s unit file contains *WantedBy=T*, then

  systemctl enable S

  will create a symlink to S.service in /etc/systemd/system/*T*.wants
- Similarly

  systemctl disable S

  removes the symlink.
- To *blacklist* a service

  systemctl mask S.service
- 'rm' or 'ln' can manage the services: there is no binary 'registry' DB.

Exercise 2: Make HelloWorld.service run at Boot

- Modify HelloWorld.service.
- Enable it.
- Reboot and verify that the service is now started.
- Disable the service, reboot and verify that service is not started.

Solution: make HelloWorld.Service run at boot

- Append a “WantedBy” line to a new [Install] section in the unit:
  [Install]
  WantedBy=multi-user.target
- Boot container and enable the unit:

  sudo systemd-nspawn -bD /var/lib/machines/debian
  [inside container] sudo systemctl enable HelloWorld
  [inside container] ls /etc/systemd/system/multi-user.target.wants
- Reboot and check status:
  [inside container] sudo systemctl reboot
  [inside container] systemctl status HelloWorld
- Disable the service, reboot and check again:
  [inside container] sudo systemctl disable HelloWorld [fails if the file is cp'ed, not ln'ed]
  [inside container] sudo systemctl reboot
  [inside container] systemctl status HelloWorld

systemd's dependencies

Demo: Generate ASCII Dependency Graphs

Examples:

```
systemctl list-dependencies basic.target
systemctl list-dependencies --after cups.socket
systemctl list-dependencies --before multi-user.target
```

Generate SVG Dependency Graph

Generate dependency metadata:

```
systemd-analyze dot basic.target > basic.dot
```

Generate graph image:

```
dot -Tsvg basic.dot -o basic.svg
```

View graph:

```
eog basic.svg  (or view basic.svg with any web browser)
```

Note: dot is in graphviz package; eog is in eponymous one.

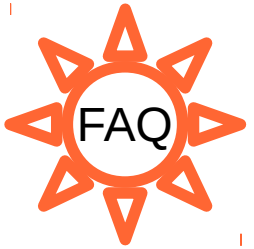
systemd bootup is ordered, but not deterministic

- Services start other services they 'Want' or 'Require'.
- Services stop if other services they 'Require' stop, but not if services they 'Want' stop.
- 'After' means '*start* after another service *starts'.*
  - ***Not*** 'start after another service is fully initialized' or finished.
  - 'Before' is similar.
- To express more nuanced sequence, use Path, PID or Socket-based signalling. Examples:
  - ConditionPathExists= in unit file listing /var/run/*.pid
  - systemd-notify messages to socket


Simple targets
vs.
runlevels


Not all targets are 'runlevels'

courtesy Pierre-Yves Beaudoin

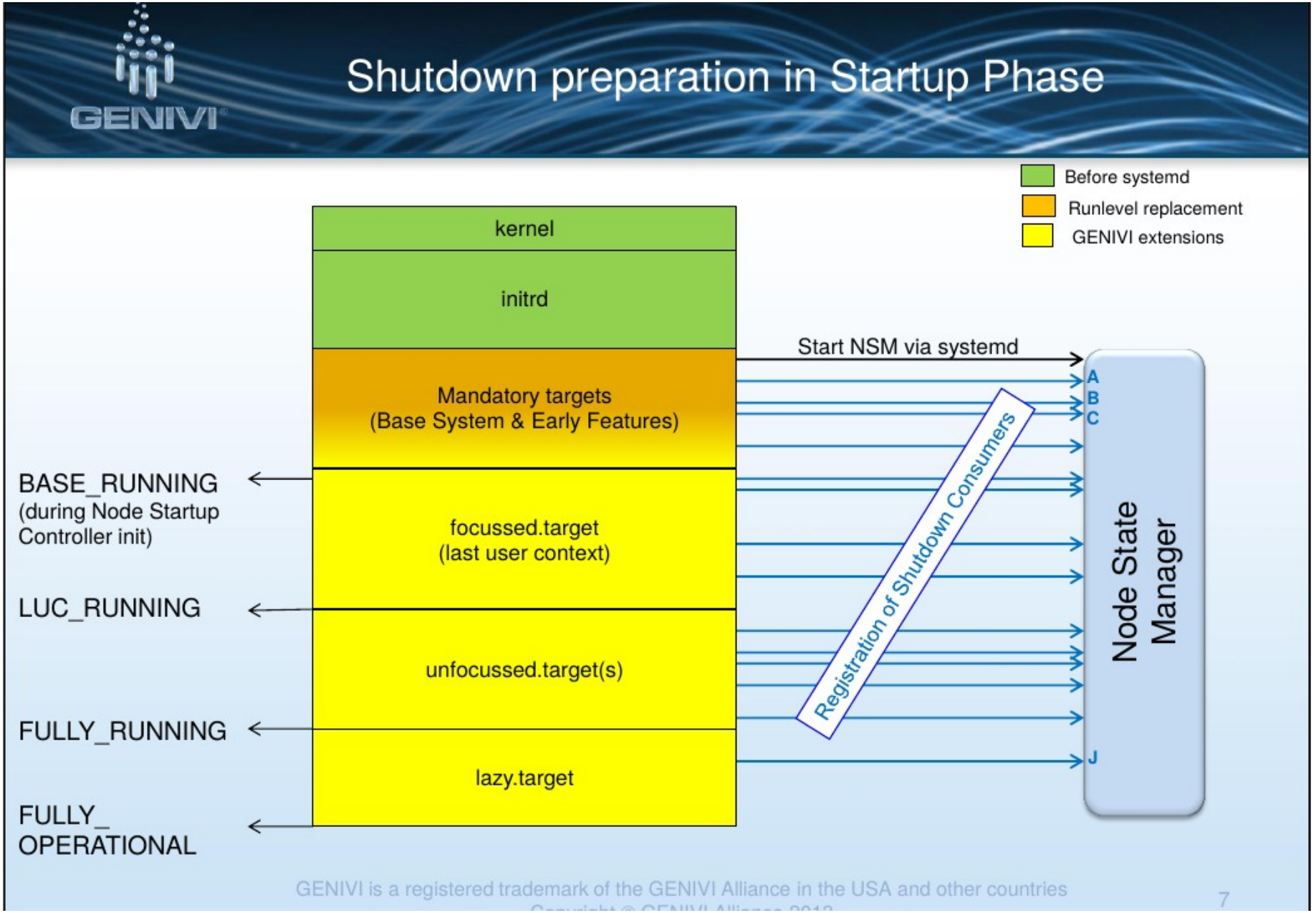


- Targets can simply be collections of services all *started* at once.
- A *runlevel* is a special target that is reached only when all wanted services reach completion.
- RTFM: man systemd.special
- New simple targets = new unit files + directories with symlinks.
- New runlevels require new code.

FAQ: how do I create a new runlevel?

- You *don't* want to.
  - Doing so involves writing a bunch of C/C++ code.
- Creating a new runlevel *is* possible.
  - GENIVI automotive Linux project has done it.
  - Code is available from

    git://git.projects.genivi.org/lifecycle/node-startup-controller.git
  - Webcast slides and audio
  - *Use case*: a LAN with many dumb no-OS MCUs.
- Is your use case *truly* so different from those considered by freedesktop.org?



From GENIVI Lifecycle Management webcast slides; the source code


Unit file hierarchy and precedence


system and user system instances

- systemd's system instance manages singleton daemons that provide systemwide services;
- systemd's user instance manages per-user services.
- Try:
  - systemctl --user status
- Discuss: why does systemctl --user status fail?
- Configuration files are in $HOME, not /etc/systemd.
- User instance only runs if systemd is built with PAM feature:

  systemctl --version | grep PAM

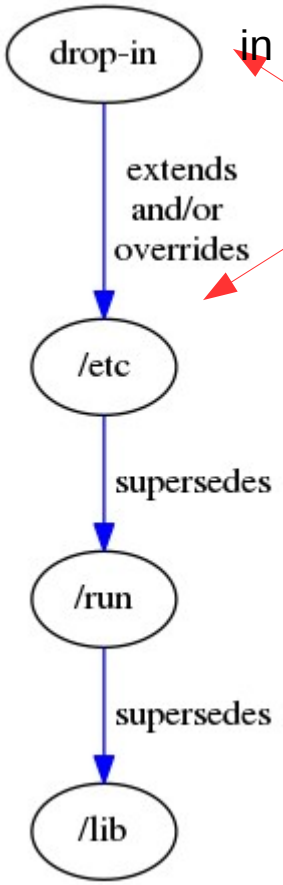
system and user units

- Organized into *system* and *user* units.
- */lib/systemd/system:* systemd upstream's defaults for system-wide services
- */usr/lib/systemd/user/*: systemd upstream's defaults for per-user services
- $HOME/.local/share/systemd/user/ for user-installed units
- 'drop-ins' are run-time extensions (man systemd.unit) for either user or system instances.

Precedence of *system* unit files

man systemd.unit

highest; optional;
static



in /etc/systemd/system/<unit-name>.d/foo.conf

alternatives that produce the same result

optional;
static

in /etc/systemd/system/foo.service

optional;
dynamically generated

in /run/systemd/system/foo.service

lowest;
from upstream

in /lib/systemd/system/foo.service

Tip: create unit files for *new* services in /etc. Drop-ins are for override.

Exercise 3:Understanding unit file hierarchy

- Display path and text of currently *loaded* unit file.

  systemctl cat systemd-logind

- Copy the currently loaded unit to a position higher in the unit-file hierarchy.

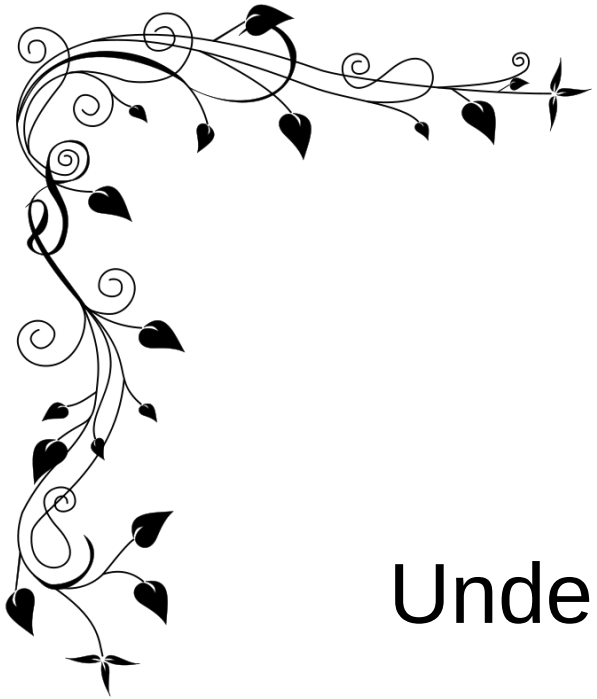
  sudo cp /lib/systemd/system/systemd-logind.service /etc/systemd/system

- Try: systemctl cat systemd-logind
  - Is the result what you expected? Why?
- Another clue:

  systemd-delta

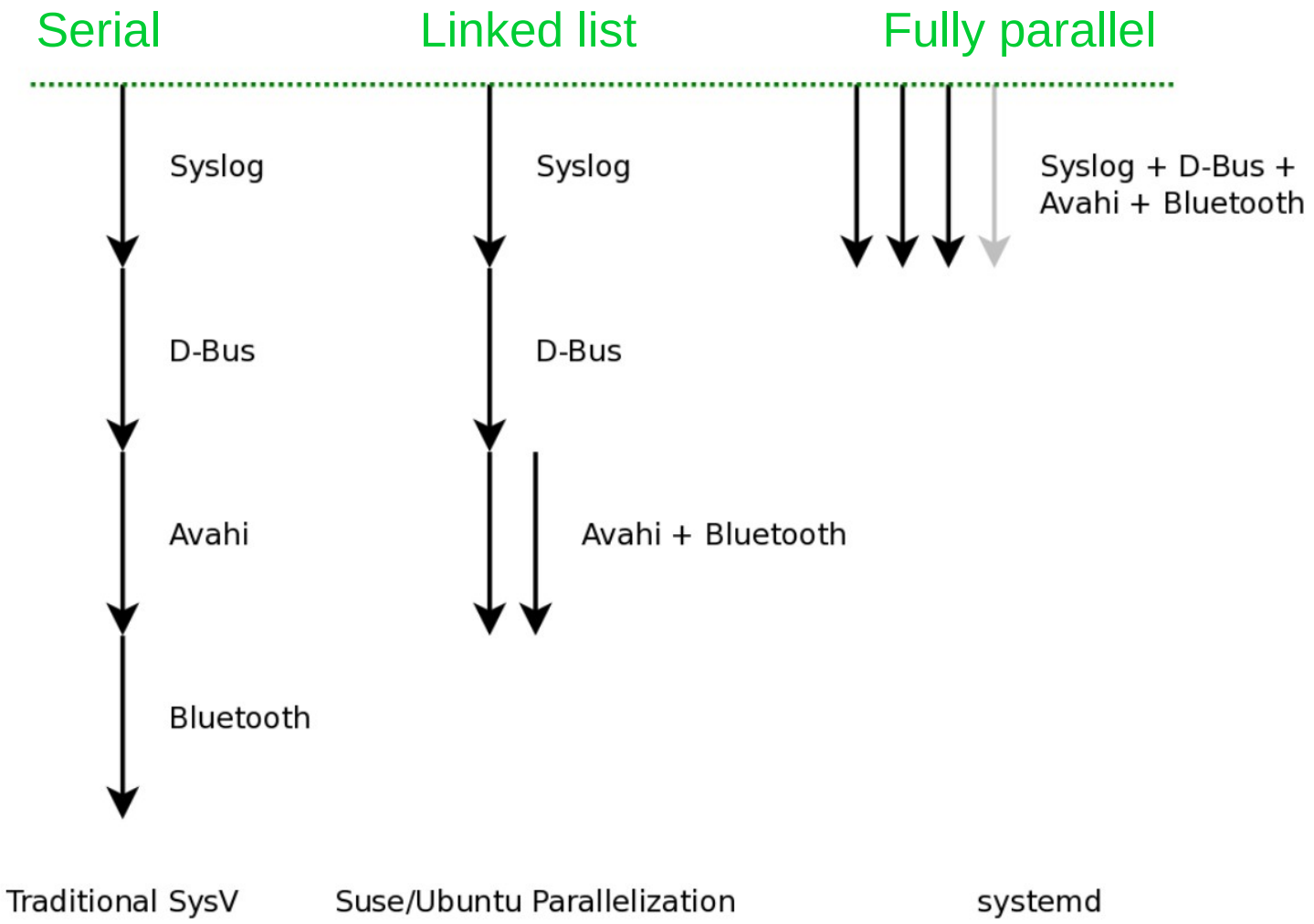
Unit file hierarchy puzzle: the answer

- sudo systemctl daemon-reload
- systemctl cat systemd-logind
- Clean-up. (Why is this important?)
  - sudo rm /etc/systemd/system/systemd-logind.service
- And repeat sudo systemctl daemon-reload




Understanding socket-based activation


Socket-based activation is key to systemd's fast boot



Demo:control cups via socket-based activation

- Check if cups is running and stop it:
  systemctl status cups.service
  sudo systemctl stop cups.service
  systemctl status cups.service
- What is cups.socket?
  systemctl cat cups.socket
  systemctl status cups.socket
- What is the difference between /lib/systemd/system/cups.socket and /var/run/cups/cups.sock?
- cups.sock is a normal AF_UNIX socket, so
  echo “HTTP POST” | ncat -U /var/run/cups/cups.socket
- Now check cups.service:
  systemctl status cups.service


Tune and control your configuration with systemd

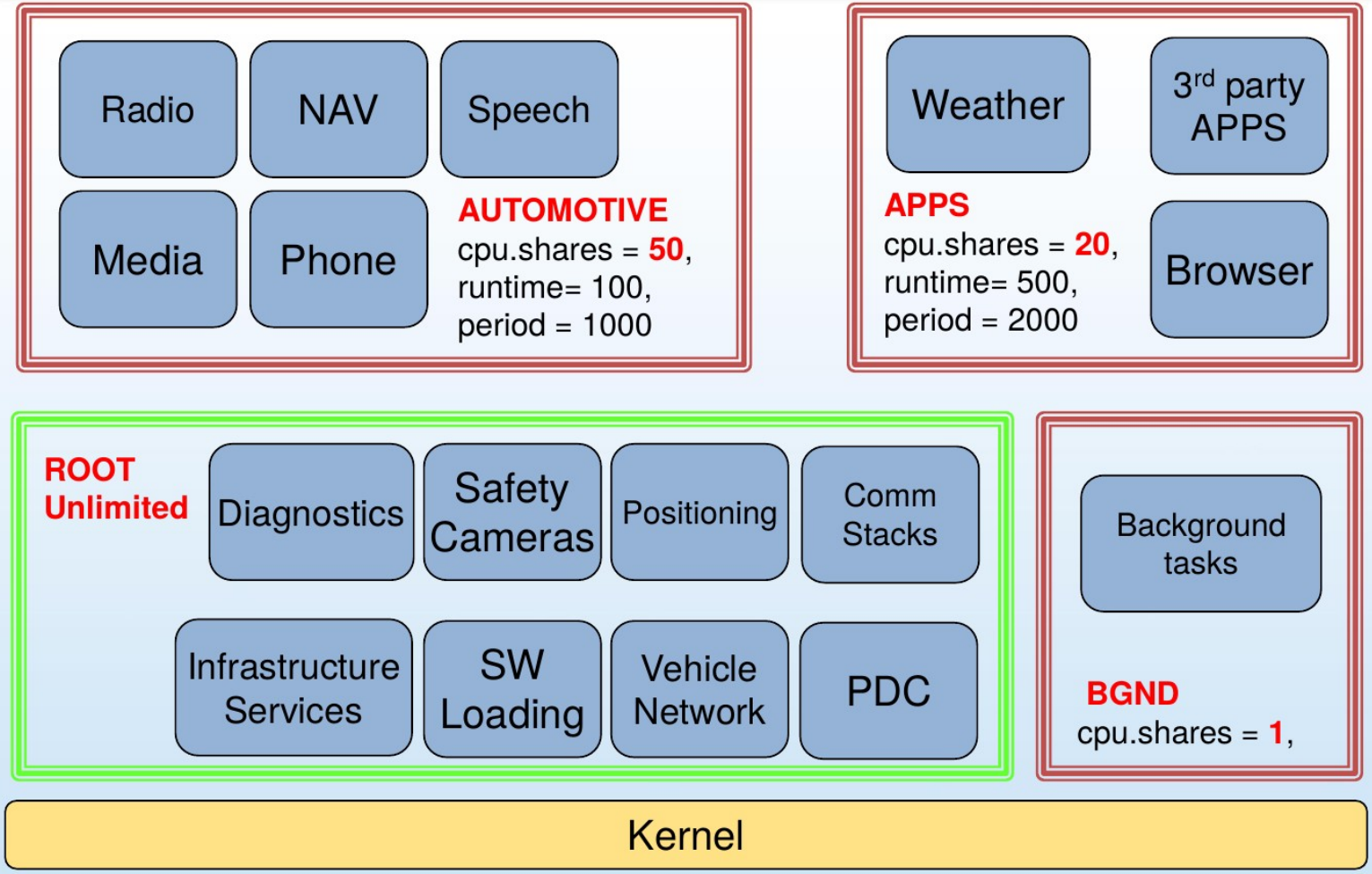

systemd intuitively exposes kernel interfaces

- Including Capabilities, Watchdog, Cgroups and kdbus ('coming attraction')
- Kernel features are configurable via systemd's unit files.
- Encourages creation of system-wide ***policies*** via unit templates.
- man 7 capabilities

systemd and cgroups

- cgroups were difficult to config prior to advent of systemd tools.
- *cgroups* are a kernel-level mechanism for allocating resources: storage, memory, CPU and network.
- *slices* are groups of *services* whose resources are managed jointly.
- systemd *scopes* are resultant groups of *processes*.
- Sysadmins can set BlockIOWeight, IOSchedulingPriority, OOMScoreAdjust, CPUShares, MemoryLimit, Nice …
- Reference: kernel's documentation and 'man systemd.resource-control'

Example cgroup configuration (CPU)



From GENIVI Lifecycle Management webcast slides (GENIVI automotive Linux consortium)

check cgroup configuration on your current system

- systemd-cgls
- systemd-cgtop
- mount | grep cgroup shows which 'controllers' are available
- NOTE:

  Which controllers are available depends on the kernel config:

  grep CGROUP /boot/config*

- NOTE:

  unless "CPUAccounting=1", "MemoryAccounting=1" and "BlockIOAccounting=1" are enabled for the services in question, no resource accounting will be available for system services and the data shown by systemd-cgtop will be incomplete.

Exercise 4: set 'niceness' of Firefox

- Create a service that starts Firefox with *per-user* settings in firefox.slice.
- Set the 'niceness' of Firefox.
- Check that the process runs at the 'niceness' you've set.
- Hints:
  - You may need to run 'xhost +localhost' on localhost.
  - Possibly add 'Environment=DISPLAY=:0' to your unit file.
  - man systemd.exec

Solution: nice Firefox

- Create a firefox.service file in /usr/lib/systemd/user:

  [Unit]
  Description=Firefox web browser

  [Service]
  Environment=DISPLAY=:0
  ExecStart=/usr/bin/firefox (might be /bin/firefox)
  Nice=12
  Slice=firefox.slice (optional but worth trying to see its effect)

- systemctl --user start firefox
- systemd-cgls
- Employ ps or top to check 'niceness'.
- Note that you now need

  systemctl --user enable firefox
  systemctl --user daemon-reload
  journalctl –user-unit=firefox (*not* journalctl --user though)

systemd and security: granular encapsulationvia kernel's *capabilities*

- *CapabilityBoundingSet* at boot; capability dropping possible
- *PrivateTmp, PrivateDevices, PrivateNetwork, JoinNamespaces*
- *ProtectSystem* (/usr and /etc), *ProtectHome*
- *ReadOnlyDirectories*, *InaccessibleDirectories*
- Set system-wide security policies via /etc/systemd/*conf files
- References: LWN on “Inheriting capabilities” and man capabilities

Exercise 6: control file access of firefox.service

- Add 'CapabilityBoundingSet=' to firefox.service and restart.
  - Investigate with getpcaps, journalctl and systemctl. (getpcaps may not be in your default $PATH.)
- Replace CapabilityBoundingSet directive with 'InaccessibleDirectories=/home'.
- Move to /etc/systemd/system and restart.
  - Try to read files in /home with the browser after starting it from 'sudo -i'.
  - Explain the behavior.
- Don't forget 'systemctl daemon-reload' and '--user'.

Solution: limiting Firefox's access

- Starting firefox.service as jack, from /etc/systemd/user, with CapabilityBoundingSet=

```
[jack@f22container ~]$ systemctl --user daemon-reload
[jack@f22container ~]$ systemctl --user start firefox
[jack@f22container ~]$ systemctl --user --failed
  UNIT            LOAD   ACTIVE SUB    DESCRIPTION
● firefox.service loaded failed failed Firefox web browser

[jack@f22container ~]$ journalctl --user -p err
Sep 19 16:44:03 f22container systemd[300]: Failed at step
CAPABILITIES spawning /bin/firefox: Operation not permitted
```

Solution: limiting Firefox's access

- Starting firefox.service with sudo from /etc/systemd/system, without 'CapabilityBoundingSet=',

  bash-4.3# getpcaps `pidof firefox`

  Capabilities for `1923': = cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_raw,cap_ipc_owner,cap_sys_chroot,cap_sys_ptrace,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap+ep

- With 'CapabilityBoundingSet=',

  bash-4.3# systemctl daemon-reload

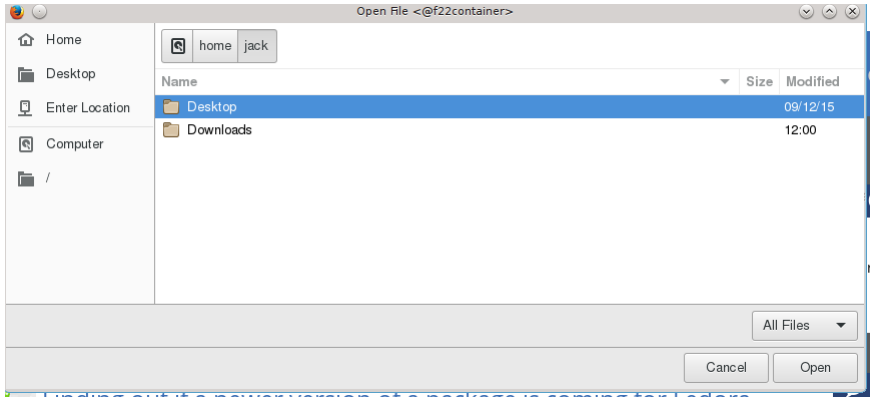
  bash-4.3# getpcaps `pidof firefox`

  Capabilities for `2036': =

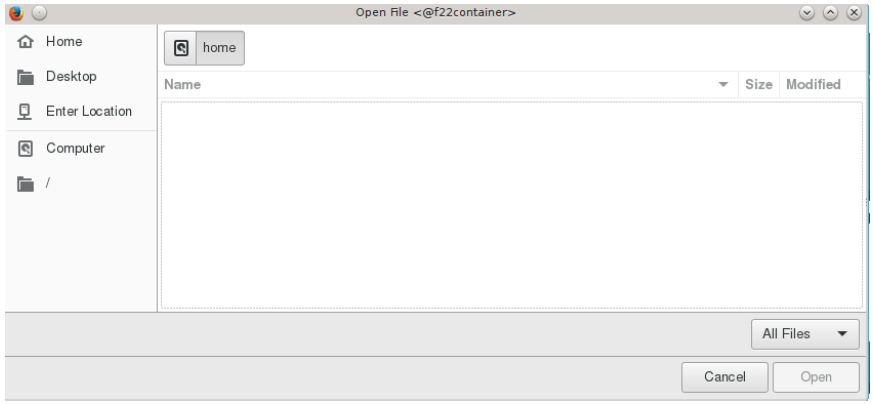
- A bit simpler than SELinux!

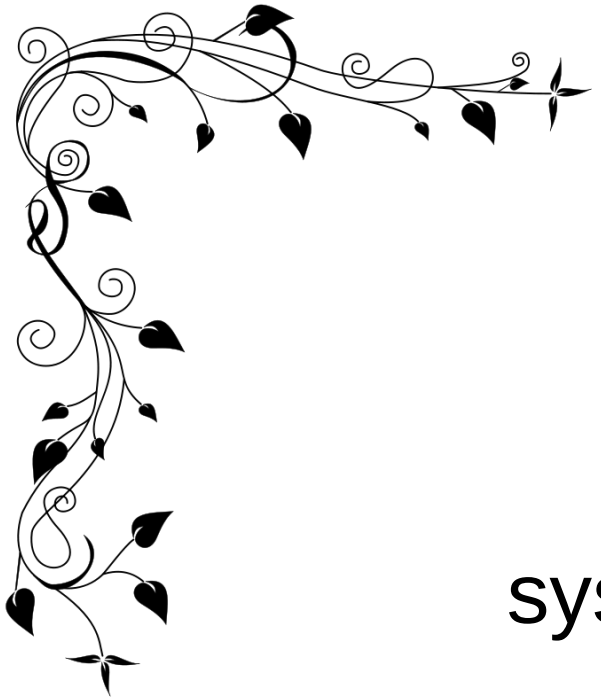
Solution: limit Firefox's access

- Starting firefox.service as root from /etc/systemd/system and *without* 'InaccessibleDirectories=/home',



- Starting firefox.service as root from /etc/systemd/system and *with* 'InaccessibleDirectories=/home',






systemd troubleshooting


ProTips!

When all else fails, consult the files in /etc/systemd/.conf.*

Dump all potential configuration items:

/lib/systemd/systemd --dump-configuration-items

Most useful man pages:

man systemd.exec
man systemd.unit
man systemd.service

Consult systemd mailing list archives and wiki.

A bit more about the systemd journal

- In binary format, but has a simple UI that beats 'grep' and 'awk'.
- Is fully compatible with parallel syslog output.
- Can push the journal to a remote via unit file configuration.
- Can be automatically cryptographically signed.
- Is, with udev, one of the required systemd components.

systemd prevents self-injury!

- Test out new units by trying them:
  - systemd-analyze verify <new unit>
  - in /run
  - in *.conf.d directory
  - via bootargs
- ***Do not ever modify files in /lib/systemd***.
  - Restore defaults by removing broken units with higher precedence.
- Services linked into basic.target.wants (≈runlevel 1) that won't work until graphical.target (runlevel 5) will start properly if their dependencies are correctly stated.

systemd's watchdog timer 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.

'systemd-analyze critical-chain': Why did that unit take so long to start?

```
[alison@hildesheim ~]$ systemd-analyze critical-chain ntp.service
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.

ntp.service +273ms
└─basic.target @2.354s
  └─sockets.target @2.353s
    └─dbus.socket @2.353s
      └─sysinit.target @2.331s
        └─nfs-common.service @2.278s +51ms
          └─rpcbind.target @2.277s
            └─rpcbind.service @2.239s +37ms
              └─network-online.target @2.238s
                └─network.target @2.237s
                  └─networking.service @2.173s +62ms
                    └─systemd-random-seed.service @2.140s +13ms
                      └─var.mount @2.117s +3ms
                        └─systemd-fsck@dev-disk-by\x2duuid-5a1ef8c6\x2d2ae4\x2d479f\x2d9345\x2d7366199025a5.service @
                          └─dev-disk-by\x2duuid-5a1ef8c6\x2d2ae4\x2d479f\x2d9345\x2d7366199025a5.device @2.044s
```

Note: ntp was started by SysVinit!!

Final quiz

- T/F: systemd is best characterized as an init system.
- Which of the following is not a recommended way to customize systemd?
  a. Edit /etc/systemd/*.conf files;
  b. Edit the files in /lib/systemd/system;
  c. Edit /etc/systemd/<unit-name.d>/*.conf files;
  d. Employ “systemctl enable” and “systemctl disable”.
- Which of the following is not a real systemd component?

  systemd-nspawn, systemd-logind, packagectl, systemd-delta
- Which of the following is true? The systemd journal:
  a. is incompatible with syslog;
  b. can be viewed with systemd-journalviewer or a browser;
  c. can be cryptographically signed automatically;
  d. is configured via an XML file.
- T/F: systemd services are always started via socket-based activation.

Summary

- systemd is easier to configure and customize than you fear.
- Most users will not notice (or *have not* notice*d*).
- There *are* real difficulties but
  - systemd is still relatively new;
  - system administration is complex.

Additional Resources

- Man pages are part of systemd git repo.
- freedesktop.org: systemd mailing list archives and wiki; Pöttering's blog
- #systemd on Freenode IRC
- ➟At wayback machine: “Booting up” articles
- systemd.conf YouTube channel and slides
- Neil Brown series at LWN on 'systemd programming' (design of NFS units)
- ➟Fedora's SysVinit to systemd cheatsheet
- LWN on “How Debian managed the systemd transition”
- Linux Action Show interview with Lennart Poettering
- “Who wrote systemd?” statistics
- Jordan Hubbard of FreeBSD describes launchd porting plans (at 40 mins.)

Acknowledgements

- twb and ohsix on #systemd on freenode IRC
- Zbigniew Jędrzejewski-Szmek on systemd-devel
- Kevin Dankwardt for help with organizing class
- USENIX/LISA for invitation.

https://commons.wikimedia.org/wiki/File:Pronam-mudra.png#/media/File:Pronam-mudra.png

Course evaluation

- The course was too introductory/too advanced.
- The amount of lecture versus exercises was too high/too low.
- The course content is relevant to my work: T/F.
- I now understand systemd better: T/F.
- I know how to find more information about systemd: T/F.

  Email to alison@she-devel.com

system and user units derive from D-Bus

- systemd cooperates with D-Bus to provide:
  - singleton daemons that provide systemwide services;
  - per-user services.
- Try:
  - busctl --system | head
  - busctl --user | head
- Same information is accessible via *qdbus* or *gdbus*.
- Reference: “Control your Linux desktop with D-Bus”

Exercise: control firefox's memory utilization

- The following works on systems where localhost's kernel is compiled with CONFIG_MEMCG=y.
  - Don't forget that containers share the kernel with localhost.
- Create a unit file that will start firefox.
- Turn on memory accounting.
- Check firefox's memory accounting via systemd-cgtop.
- Add a MemoryLimit field to the unit file.
- Restart your service and check the memory utilization again: top or ps -o slice,vsize,rss,%mem -C firefox.
- Hints: you may need to run 'xhost +localhost' on localhost and add 'Environment=DISPLAY=:0' to your unit file.

Firefox and cgroups solution

- firefox.service:

```
[Unit]
Description=Firefox web browser

[Service]
Environment=DISPLAY=:0
ExecStart=/usr/bin/firefox    (or /bin/firefox)
MemoryAccounting=true
MemoryLimit=10M
```

- sudo mv firefox.service /etc/systemd/user
- systemctl --user start firefox
- systemd-cgtop and ps -o slice,vsize,rss,%mem -C firefox
- Remove MemoryLimit and compare.

Taxonomy of systemd tools

- Analogous to 'git'.
- 'Porcelain' generalized tools: 'ls /bin/*ctl'
  - journalctl, systemctl, machinectl, busctl, loginctl, networkctl
  - Man pages, useful in bash scripts.
- 'Plumbing' components: 'find /lib/systemd -executable -type f'
  - A few lack man pages; try '--help'.
  - Tools that are invoked by other tools.
  - May be useful in testing.
- Domain-specific: 'ls /usr/bin/systemd-*'