Unshrouding Systemd

Neal Gompa (Conan Kudo [ニール・ゴンパ])

Who am I?

- Professional technologist
- Humble maintainer of a handful of packages in the Fedora Project
- Diligent follower of the telecommunications industry
- Associate SQA Engineer at Datto, Inc

Contact Points:

- Twitter: @Det_Conan_Kudo
- Google+: +NealGompa

What is systemd?

According to Wikipedia:

- ***systemd*** *is a suite of system management daemons, libraries, and utilities designed as a central management and configuration platform for the Linux computer operating system.*

Err, daemons?

If you are coming from the Windows world, you know “daemons” as “services”. Or if you’re really old-school, these are analogous to DOS TSR (terminate and stay resident) applications.

Pronounced as *DAY-mon* or *DEE-mon*, these are special applications that spin up a process that remains in the background to ***do something***.

Note that “dæmon” is not considered a valid spelling, and actually is pronounced differently (*DAH-mon*), though you will hear speakers of Germanic languages other than English say it this way since “ae” still is pronounced that way in most languages.

Major features of systemd

- Low-level logging functionality from startup to shutdown
- Highly efficient process and service management with socket and timer activation
- Efficient, simple, and flexible network management
- Device enumeration and management
- User login and session management
- Service oriented framework for launching and managing containers
- Hostname management
- Time and date management
- Locale management
- “99%” compatibility with SysVinit scripts
- And much more…

Whoa… That's too much!

So… The truth is that systemd doesn't *really* do all those things itself. Systemd is actually a collection of applications (roughly 70 of them) that do each of those things. A great deal of these components are optional, and don't have to be used.

Die-hard Unix folks can lower their pitchforks, as systemd isn't one big monolithic beast that does ***everything!***

I’ve never heard of this, prove it!

Err, okay…

- Logger: systemd-journald
- Service manager: systemd
- Network management: systemd-networkd
- Device management: systemd-udevd
- Login & sessions: systemd-logind
- Container framework: systemd-nspawn, systemd-machined
- Hostname management: systemd-hostnamed
- Time & date management: systemd-timedated
- Locale management: systemd-localed
- And of course, there are even more examples…

An example of a systemd service

```
[Unit]
Description=Bandwidthd Network Traffic Monitor
After=network.target

[Service]
Type=forking
PIDFile=/run/bandwidthd.pid
ExecStart=/usr/sbin/bandwidthd

[Install]
WantedBy=multi-user.target
```

The same one, as sysvinit script

```
#! /bin/bash
# bandwidthd
# chkconfig: - 90 26
# description: Activates/Deactivates bandwidthd network traffic monitor
PROGNAME=/usr/sbin/bandwidthd
# Source function library.
. /etc/init.d/functions
if [ ! -f /etc/sysconfig/network ]; then
    exit 0
fi
. /etc/sysconfig/network
if [ -f /etc/sysconfig/bandwidthd ]; then
    . /etc/sysconfig/bandwidthd
fi
# Check that networking is up.
[ "${NETWORKING}" = "no" ] && exit 0
```

Continued on the right →→→→→→→→→→→→→→→→→→→→

```
# See how we were called.
case "$1" in
  start)
        #Register to dns
        echo -n $"Starting Bandwidthd network traffic monitor: "
                daemon $PROGNAME $OPTIONS
                RETVAL=$?
                echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/bandwidthd
                exit $RETVAL
        ;;
  stop)
        echo -n $"Shuting down Bandwidthd network traffic monitor: "
                killproc `basename $PROGNAME`
        RETVAL=$?
                echo
        [ $RETVAL -eq 0 ] && success || failure
                echo
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/bandwidthd
                exit $RETVAL
        ;;
```

Wait, there's still more!

```
 status)
                status $PROGNAME
                ;;
 condrestart)
     if [ -f /var/lock/subsys/bandwidthd ]; then
        $0 stop
        $0 start
     fi
     ;;
 restart|reload)
                $0 stop
                $0 start
                ;;
 *)
     echo $"Usage: $0 {start|stop|restart|reload|status}"
     exit 1
esac
exit 0
```

Wait, what's *socket activation*?

Socket activation is when a service isn't activated and running until data comes in on a specific socket (typically an internet socket). When a connection is established, systemd activates the service and passes control of the socket to the service. When no more clients are connected to the socket, the service shuts down until it is needed again.

Socket-activated service example

cockpit.service

```
[Unit]
Description=Cockpit Web Server
Documentation=man:cockpit-ws(8)

[Service]
ExecStartPre=/usr/sbin/remotectl certificate --ensure
--user=root --group=cockpit-ws
ExecStart=/usr/libexec/cockpit-ws
PermissionsStartOnly=true
User=cockpit-ws
Group=cockpit-ws
```

cockpit.socket

```
[Unit]
Description=Cockpit Web Server Socket
Documentation=man:cockpit-ws(8)

[Socket]
ListenStream=9090

[Install]
WantedBy=sockets.target
```

And *timer activation*?

Timer activation functions similarly to socket activation, except that it uses a timing schedule instead of sockets to trigger services. Timing granularity goes down to ***one second***.

Essentially, it's a more efficient cron job system.

Timer-activated service example

dnf-makecache.service

```
[Unit]
Description=dnf makecache

[Service]
Type=oneshot
Nice=19
IOSchedulingClass=2
IOSchedulingPriority=7
Environment="ABRT_IGNORE_PYTHON=1"
ExecStart=/usr/bin/dnf -v makecache timer
```

dnf-makecache.timer

```
[Unit]
Description=dnf makecache timer
ConditionKernelCommandLine=!rd.live.image

[Timer]
OnBootSec=10min
OnUnitInactiveSec=1h
Unit=dnf-makecache.service

[Install]
WantedBy=basic.target
```

Okay… So how do I use it?

- Installing your own unit files
  - /etc/systemd/system
    - After adding files, run systemctl daemon-reload
- Overriding configuration of system provided services
  - Distribution provided services are in /usr/lib/systemd/system
    - Can be overridden by putting override *.conf files in /etc/systemd/system/<name>.<type>.d/
      - After adding files, run systemctl daemon-reload
- Start/stop/enable/disable/etc. services
  - systemctl {start,stop,restart,reload,enable,disable} <name>.<type>
    - “.service” is assumed when type isn’t defined

Journaling in systemd

In a system using systemd *properly*, the journaling mechanisms in Linux are extended to cover every stage of the system's life (system startup, online state, and system shutdown).

Systemd also presents a unified journal interface: journalctl

Using the journal

In pre-systemd systems, accessing the journal was a matter of locating the right log file and using tools like grep to “parse” it.

Through systemd-journald, all journal data is stored in an ACID-like database that can be queried through an API or journalctl. While the other files will likely also exist on the system, journalctl is a very handy tool to use to read logs directly.

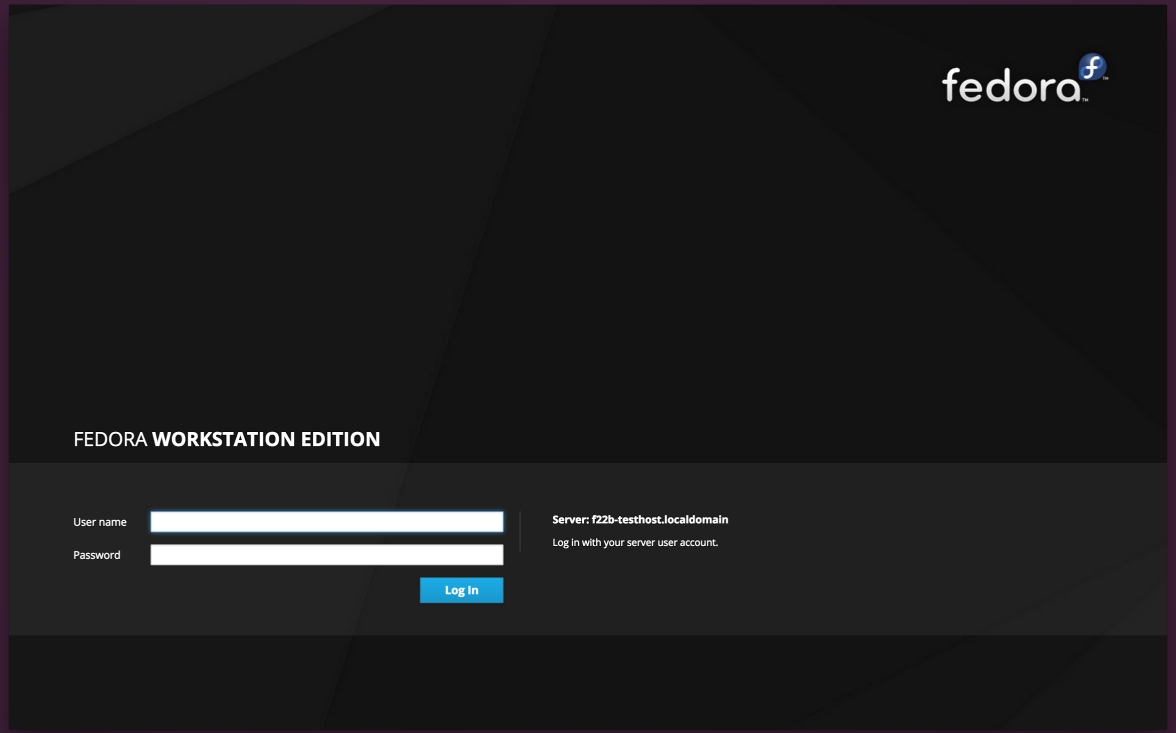
The journald journal is a special kind of data store that allows it to remain safe to use even when writes fail, which isn’t always true for legacy journal schemes.

And there’s a lot more!

These are some of the basics of systemd that every Linux user will want to know about. There’s a lot more that can be done through various capabilities in systemd and its associated services and APIs.

But I use GUIs to do stuff!

Well… I'd argue that you don't need one, with how easy the tools are, but…



Cockpit, the web GUI

Desktop GUIs

Aside from Cockpit, there are these GUI programs:

- systemd-ui
  - A bit old and unmaintained, but still very good
- KDE systemd applets
  - KDE 4 and KDE 5 have built-in applets in the System Settings configuration panel
- GNOME system configuration
  - GNOME's system configuration programs already utilize systemd APIs to manage parts of the system

Other desktop environments may have their own equivalents...

Demonstration

Cockpit GUI and Systemd CLI

Some additional resources…

- Systemd Wikipedia page: https://en.wikipedia.org/wiki/Systemd
- Systemd website: http://www.freedesktop.org/wiki/Software/systemd/
- Systemd documentation center: http://0pointer.de/blog/projects/systemd-docs.html
- Lennart Poettering: *Rethinking PID 1*: http://0pointer.de/blog/projects/systemd.html
- Lennart Poettering: *Why systemd?*: http://0pointer.de/blog/projects/why.html
- Ben Breard & Lennart Poettering: *Demystifying systemd*
  - YouTube recording: https://www.youtube.com/watch?v=S9YmaNuvw5U
  - PDF slides: http://videos.cdn.redhat.com/summit2015/presentations/12720_demystifying-systemd.pdf
  - This presentation was shamelessly inspired from it
- Fedora Project: *SysVinit to Systemd Cheatsheet*: https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet
- Cockpit Project: *Cockpit Guide*: http://files.cockpit-project.org/guide/latest/

The End

Any Questions?