Systemd for Administrators

Lennart Poettering
Author of the Handbook

Christian Rebischke
Merged everything together

February 26, 2016

Contents

1 Abstract

As many of you know, systemd is the new Fedora init system, starting with F14, and it is also on its way to being adopted in a number of other distributions as well (for example, OpenSUSE). For administrators systemd provides a variety of new features and changes and enhances the administrative process substantially. This blog story is the first part of a series of articles I plan to post roughly every week for the next months. In every post I will try to explain one new feature of systemd. Many of these features are small and simple, so these stories should be interesting to a broader audience. However, from time to time we'll dive a little bit deeper into the great new features systemd provides you with.

2 Disclaimer

This handbook is written by Lennart Poettering. There are maybe some additions, cuts or other changes for increasing readability. Please visit Lennarts Blog for the original Blogposts:

`https://0pointer.de/blog`

3 Verifying Bootup

Traditionally, when booting up a Linux system, you see a lot of little messages passing by on your screen. As we work on speeding up and parallelizing the boot process these messages are becoming visible for a shorter and shorter time only and be less and less readable – if they are shown at all, given we use graphical boot splash technology like Plymouth these days. Nonetheless the information of the boot screens was and still is very relevant, because it shows you for each service that is being started as part of bootup, wether it managed to start up successfully or failed (with those green or red [OK] or [FAILED] indicators). To improve the situation for machines that boot up fast and parallelized and to make this information more nicely available during runtime, we added a feature to systemd that tracks and remembers for each service whether it started up successfully, whether it exited with a non-zero exit code, whether it timed out, or whether it terminated abnormally (by segfaulting or similar), both during start-up and runtime. By simply typing systemctl in your shell you can query the state of all services, both systemd native and SysV/LSB services:

```
[root@lambda] ~# systemctl
UNIT                                                LOAD   ACTIVE       SUB
dev-hugepages.automount                             loaded active       running
dev-mqueue.automount                                loaded active       running
proc-sys-fs-binfmt_misc.automount                   loaded active       waiting
sys-kernel-debug.automount                          loaded active       waiting
sys-kernel-security.automount                       loaded active       waiting
sys-devices-pc...0000:02:00.0-net-eth0.device       loaded active       plugged
sys-devices-virtual-tty-tty9.device                 loaded active       plugged
-.mount                                             loaded active       mounted
boot.mount                                          loaded active       mounted
dev-hugepages.mount                                 loaded active       mounted
dev-mqueue.mount                                    loaded active       mounted
home.mount                                          loaded active       mounted
proc-sys-fs-binfmt_misc.mount                       loaded active       mounted
abrtd.service                                       loaded active       running
bus.service                                        loaded active       running
getty@tty2.service                                  loaded active       running
getty@tty3.service                                  loaded active       running
getty@tty4.service                                  loaded active       running
getty@tty5.service                                  loaded active       running
getty@tty6.service                                  loaded active       running
haldaemon.service                                   loaded active       running
hdapsd@sda.service                                  loaded active       running
irqbalance.service                                  loaded active       running
iscsi.service                                       loaded active       exited
iscsid.service                                      loaded active       exited
livesys-late.service                                loaded active       exited
livesys.service                                     loaded active       exited
lvm2-monitor.service                                loaded active       exited
mdmonitor.service                                   loaded active       running
modem-manager.service                               loaded active       running
netfs.service                                       loaded active       exited
NetworkManager.service                              loaded active       running
ntpd.service                                        loaded maintenance  maintenance

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
JOB    = Pending job for the unit.

221 units listed. Pass --all to see inactive units, too.
[root@lambda] ~#
```

(I have shortened the output above a little, and removed a few lines not relevant for this blog post.) Look at the ACTIVE column, which shows you the high-level state of a service (or in fact of any kind of unit systemd maintains, which can be more than just services, but we'll have a look on this in a later blog posting), whether it is **active** (i.e. running), **inactive** (i.e. not running) or in any other state. If you look closely you'll see one item in the list that is marked

maintenance and highlighted in red. This informs you about a service that failed to run or otherwise encountered a problem. In this case this is ntpd. Now, let's find out what actually happened to ntpd, with the *systemctl status* command:

```
[root@lambda] ~# systemctl status ntpd.service
ntpd.service - Network Time Service
      Loaded: loaded (/etc/systemd/system/ntpd.service)
      Active: maintenance
        Main: 953 (code=exited, status=255)
      CGroup: name=systemd:/systemd-1/ntpd.service
[root@lambda] ~#
```

This shows us that NTP terminated during runtime (when it ran as PID 953), and tells us exactly the error condition: the process exited with an exit status of 255.

In a later systemd version, we plan to hook this up to ABRT, as soon as this enhancement request is fixed. Then, if systemctl status shows you information about a service that crashed it will direct you right-away to the appropriate crash dump in ABRT.

Summary: use *systemctl* and *systemctl status* as modern, more complete replacements for the traditional boot-up status messages of SysV services. *systemctl status* not only captures in more detail the error condition but also shows runtime errors in addition to start-up errors. That's it for this week, make sure to come back next week, for the next posting about systemd for administrators!

4 Which Service Owns Which Processes?

On most Linux systems the number of processes that are running by default is substantial. Knowing which process does what and where it belongs to becomes increasingly difficult. Some services even maintain a couple of worker processes which clutter the "ps" output with many additional processes that are often not easy to recognize. This is further complicated if daemons spawn arbitrary 3rd-party processes, as Apache does with CGI processes, or cron does with user jobs.

A slight remedy for this is often the process inheritance tree, as shown by "ps xaf". However this is usually not reliable, as processes whose parents die get reparented to PID 1, and hence all information about inheritance gets lost. If a process "double forks" it hence loses its relationships to the processes that started it. (This actually is supposed to be a feature and is relied on for the traditional Unix daemonizing logic.) Furthermore processes can freely change their names with PR_SETNAME or by patching argv[0], thus making it harder to recognize them. In fact they can play hide-and-seek with the administrator pretty nicely this way.

In systemd we place every process that is spawned in a control group named after its service. Control groups (or cgroups) at their most basic are simply groups

of processes that can be arranged in a hierarchy and labelled individually. When processes spawn other processes these children are automatically made members of the parents cgroup. Leaving a cgroup is not possible for unprivileged processes. Thus, cgroups can be used as an effective way to label processes after the service they belong to and be sure that the service cannot escape from the label, regardless how often it forks or renames itself. Furthermore this can be used to safely kill a service and all processes it created, again with no chance of escaping.

In today's installment I want to introduce you to two commands you may use to relate systemd services and processes. The first one, is the well known ps command which has been updated to show cgroup information along the other process details. And this is how it looks:

```
$ ps xawf -eo pid,user,cgroup,args
  PID USER     CGROUP                                  COMMAND
    2 root     -                                       [kthreadd]
    3 root     -                                        \_ [ksoftirqd/0]
[...]
 4281 root     -                                        \_ [flush-8:0]
    1 root     name=systemd:/systemd-1                 /sbin/init
  455 root     name=systemd:/systemd-1/sysinit.service /sbin/udevd -d
28188 root     name=systemd:/systemd-1/sysinit.service  \_ /sbin/udevd -d
28191 root     name=systemd:/systemd-1/sysinit.service  \_ /sbin/udevd -d
 1131 root     name=systemd:/systemd-1/auditd.service auditd
 1133 root     name=systemd:/systemd-1/auditd.service  \_ /sbin/audispd
 1135 root     name=systemd:/systemd-1/auditd.service      \_ /usr/sbin/sedispatch
 1193 root     name=systemd:/systemd-1/rsyslog.service /sbin/rsyslogd -c 4
 1195 root     name=systemd:/systemd-1/cups.service cupsd -C /etc/cups/cupsd.conf
 1210 root     name=systemd:/systemd-1/irqbalance.service irqbalance
 1216 root     name=systemd:/systemd-1/dbus.service /usr/sbin/modem-manager
 1219 root     name=systemd:/systemd-1/dbus.service /usr/libexec/polkit-1/polkitd
1317 root     name=systemd:/systemd-1/abrtd.service /usr/sbin/abrtd -d -s
 1332 root     name=systemd:/systemd-1/getty@.service/tty2 /sbin/mingetty tty2
 1339 root     name=systemd:/systemd-1/getty@.service/tty3 /sbin/mingetty tty3
 1342 root     name=systemd:/systemd-1/getty@.service/tty5 /sbin/mingetty tty5
 1343 root     name=systemd:/systemd-1/getty@.service/tty4 /sbin/mingetty tty4
 1344 root     name=systemd:/systemd-1/crond.service crond
 1346 root     name=systemd:/systemd-1/getty@.service/tty6 /sbin/mingetty tty6
 1362 root     name=systemd:/systemd-1/sshd.service /usr/sbin/sshd
 1759 lennart  name=systemd:/user/lennart/1         gnome-screensaver
 909 lennart  name=systemd:/user/lennart/1         gnome-terminal
 1913 lennart  name=systemd:/user/lennart/1          \_ gnome-pty-helper
 1914 lennart  name=systemd:/user/lennart/1          \_ bash
29231 lennart  name=systemd:/user/lennart/1          |   \_ ssh tango
 2221 lennart  name=systemd:/user/lennart/1          \_ bash
 4193 lennart  name=systemd:/user/lennart/1          |   \_ ssh tango
 2461 lennart  name=systemd:/user/lennart/1          \_ bash
27251 lennart  name=systemd:/user/lennart/1              \_ empathy
```

(Note that this output is shortened, I have removed most of the kernel threads here, since they are not relevant in the context of this blog story)

In the third column you see the cgroup systemd assigned to each process. You'll find that the udev processes are in the name=systemd:/systemd-1/sysinit.service cgroup, which is where systemd places all processes started by the sysinit.service service, which covers early boot.

My personal recommendation is to set the shell alias psc to the ps command line shown above:

```
alias psc='ps xawf -eo pid,user,cgroup,args'
```

With this service information of processes is just four keypresses away! A different way to present the same information is the systemd-cgls tool we ship with systemd. It shows the cgroup hierarchy in a pretty tree. Its output looks like this:

```
$ systemd-cgls
+     2 [kthreadd]
[...]
+ 4281 [flush-8:0]
+ user
| \ lennart
|   \ 1
|     +  1495 pam: gdm-password
|     +  1521 gnome-session
|     +  1534 dbus-launch --sh-syntax --exit-with-session
|     +  1603 /usr/libexec/gconfd-2
|     +  1612 /usr/libexec/gnome-settings-daemon
|     +  1615 /ushr/libexec/gvfsd
|     \ 29519 systemd-cgls
\ systemd-1
  + 1 /sbin/init
  + ntpd.service
  | \ 4112 /usr/sbin/ntpd -n -u ntp:ntp -g
  + systemd-logger.service
  | \ 1499 /lib/systemd/systemd-logger
  + accounts-daemon.service
  | \ 1496 /usr/libexec/accounts-daemon
  + rtkit-daemon.service
  | \ 1473 /usr/libexec/rtkit-daemon
  + console-kit-daemon.service
  | \ 1408 /usr/sbin/console-kit-daemon --no-daemon
  + prefdm.service
  | + 1376 /usr/sbin/gdm-binary -nodaemon
  | + 1419 /usr/bin/dbus-launch --exit-with-session
  | \ 1511 /usr/bin/gnome-keyring-daemon --daemonize --login
  + getty@.service
  | + tty6
  | | \ 1346 /sbin/mingetty tty6
  | + tty4
  | | \ 1343 /sbin/mingetty tty4
  | + tty5
  | | \ 1342 /sbin/mingetty tty5
  | + tty3
  | | \ 1339 /sbin/mingetty tty3
  | \ tty2
  |   \ 1332 /sbin/mingetty tty2
   \ 28191 /sbin/udevd -d
```

5 References