systemd how-to

Frédéric Crozat
Project Manager Enterprise Desktop
fcrozat@suse.com

Agenda

- What is systemd
- Unit:
  - What is it ?
  - How to write one ?
- Tools:
  - Systemctl
  - Journalctl
  - loginctl
  - Systemd-analyze
- Debugging

What is systemd ?

- systemd is a system and session manager for Linux, compatible with SysV and LSB init scripts.
- systemd
  - provides aggressive parallelization capabilities,
  - uses socket and D-Bus activation for starting services,
  - offers on-demand starting of daemons,
  - keeps track of processes using Linux cgroups,
  - supports snapshotting and restoring of the system state,
  - maintains mount and automount points
  - implements an elaborate transactional dependency-based service control logic.
- It can work as a drop-in replacement for sysvinit.

Unit file

Unit file

- Generic term used by systemd for the following:
  - Service (ends with .service)
  - Targets (ends with .target)
  - Sockets (ends .socket)
  - Path (ends with .path, used to trigger other units)
  - Timer (ends with .timer)
  - Mount point (ends with .mount), usually autogenerated by fstab generator
  - Automount point (ends with .automount)
  - Swap (ends with .swap)
  - Device (ends with .device)
  - Scope / Slice (ends with .scope/.slice, introduced in v205)

Creating a service file

[Unit]

Description=blablabla

Requires/Wants=units_list

After/Before=units_list

[Service]

ExecStart|ExecStop=<full_path_to_binary>

ExecStartPre|ExecStartPost|..

Type=simple|forking|oneshot|dbus|notify|idle

[Install]

WantedBy=multi-user.target

Special targets

- runlevelX.target
- multi-user.target => runlevel 3
- graphical.target => runlevel 5
- local-fs.target
- network.target
- network-online.target
- remote-fs.target

- Targets are synchronization points, nothing more

- See man systemd.special(7) for complete list

Creating a service file (cont.)

Example for a daemon :

```
[Unit]
Description=Daemon to detect crashing apps

[Service]
ExecStart=/usr/sbin/abrtd
PIDFile=/run/abrtd.pid
Type=forking
[Install]
WantedBy=multi-user.target
```

Creating a service file (cont.)

- Example for a D-Bus service :

```
[Unit]
Description=Network Manager

Wants=network.target
Before=network.target

[Service]
Type=dbus
BusName=org.freedesktop.NetworkManager
ExecStart=/usr/sbin/NetworkManager --no-daemon

[Install]
WantedBy=multi-user.target
Alias=dbus-org.freedesktop.NetworkManager.service
```

Creating a service file (hints 1/2)

- ExecStart|ExecStop|Exec* are not started by a shell but directly executed. But environment variables are available.
- You can also use EnvironmentFile=/etc/foobar or Environment=foobar=value
- Type is important:
  - Simple is the default (but best to specify it)
  - Oneshot will block following units until process is finished (usually, fast scripts). Usually useful to specify RemainAfterExit=true. You can use several ExecStart with “oneshot”
  - Forking: ensure to specify PIDFile to allow systemd to track properly main daemon PID. One daemon per service.
- ExecReload isn't specified by default. You will often add: ExecReload=/bin/kill -HUP $MAINPID

Create a service file (hints 2/2)

- Prepending “-” for a file will prevent error if file doesn't exist when read or will ignore exit code for Exec* lines.
- Specifiers can be used: %n for unit name %u username, %U UID, etc..
- You can have automatic restart by adding “Restart=on-success|on-failure|on-abort|always”
- To customize error code handling, you can use SuccessExitStatus|RestartPreventExitStatus
- You can create templates: (openvpn@.service, getty@.service) where some expanded specifiers will be used (%n...)

Secure Services

- Limit network access (namespace):
  - PrivateNetwork=yes
- Private /tmp:
  - PrivateTmp=yes
- Restrict access to directories:
  - InaccessibleDirectories=/home
  - ReadOnlyDirectories=/var
- Restrict capabilities:
  - CapabilitiesBoundingSet=CAP_CHOWN CAP_KILL
  - CapabilitiesBoundingSet=~CAP_PTRACE (all but this one)
- User / Group / Root Directory:
  - User= ; Group= ; RootDirectory=
- Device access restriction (whitelist):
  - DeviceAllow=/dev/null rw
- Resources restriction:
  - LimitNPROC=1, LimitFSIZE=0...

Cgroup handling (changed since v205)

- With cpu cgroup:
  - CpuShares=1500
- With blkio cgroup:
  - BlockIOWeight=(optional path) 500
  - BlockIORead(Write)Bandwidth=/var/log 5M
- With Memory cgroup:
  - MemoryLimit=1G
- See systemd.cgroup(5) for full list
- Since v205, No longer possible to directly access any random cgroup properties (report to flames on mailing list ;)
  - Slice concept has been introduced, to partition system, using the high level cgroup knobs detailed above
  - A service can refer to a slice, with :

    Slice=foobar.slice

How to easily modify an installed unit

- /etc/systemd/system is always take precedence over /usr/lib/systemd/system
- Since openSUSE 12.3, the easiest way is to use a drop-in file:

  For foobar.service, just create:

  /etc/systemd/system/foobar.service.d/ directory

  And create additional files :

  /etc/systemd/system/foobar.service.d/whatever.conf

```
[Service]
CPUShares=1500
BlockIOWeight=500
MemoryLimit=1G
```

- Before 12.3, you could use .include fullpath but it is no longer the best way

Systemd and packages

- Unit should be installed in **/usr/lib/systemd/system**
- Various RPM macros to use :

  %{systemd_requires}
- ***%pre***

  %service_add_pre demo.service demo1.service
- *%post*

  %service_add_post demo.service demo1.service
- *%preun*

  %service_del_preun demo.service
- *%postun*

  %service_del_postun demo.service

Systemd and packages in openSUSE (cont.)

- Handle migration from sysvinit to service file (must use the same name)
- Services aren't enabled by default : it will be handled by presets (files in /usr/lib/systemd/system-preset), which should be specified in systemd-presets-branding-openSUSE package. Please use submit request to this package if needed, reviewed by secteam.

Tools

Systemctl: Service status

- systemctl : give you a list of all started services and their status
- systemctl status foobar.service : status for one specific service

```
$ systemctl status icecream.service
icecream.service - LSB: icecc
      Loaded: loaded (/etc/init.d/icecream)
      Active: active (running) since Fri, 2013-04-19 09:27:31 CEST; 4 days ago
      CGroup: name=systemd:/system/icecream.service
          ├ 4786 /usr/sbin/icecc-scheduler -d -l /var/log/icecc_sch...
          └ 4791 /usr/sbin/iceccd -d -l /var/log/iceccd --nice 5 -u...

Apr 19 09:27:31 foobar systemd[1]: Starting LSB: icecc...
Apr 19 09:27:31 foobar icecream[4777]: Starting Distribut...
Apr 19 09:27:31 foobar systemd[1]: Started LSB: icecc.
```

Where is process coming from

- Hard to debug where a process is coming from before systemd (for average user), hunt for parent PID (pstree, etc..)
- In systemd, each service has its own cgroup, allowing to identify PID easily:

```
systemd-cgls:

 system
   ├ 1 /sbin/init showopts
   ├ icecream.service
   │ ├ 4786 /usr/sbin/icecc-scheduler -d -l /var/log/icecc_scheduler
   │ └ 4791 /usr/sbin/iceccd -d -l /var/log/iceccd --nice 5 -u icecream -b /...
   ├ colord.service
   │ └ 1677 /usr/lib/colord
   ├ udisks2.service
   │ └ 1498 /usr/lib/udisks2/udisksd --no-debug
   ├ rtkit-daemon.service
   │ └ 1353 /usr/lib/rtkit/rtkit-daemon
   ├ upower.service
   │ └ 1161 /usr/lib/upower/upowerd
   ├ accounts-daemon.service
   │ └ 1125 /usr/lib/accounts-daemon
   ├ xdm.service
   │ ├  964 /usr/sbin/gdm
   │ ├  966 /usr/lib/gdm/gdm-simple-slave --display-id /org/gnome/DisplayMan...
   │ ├ 1021 /usr/bin/Xorg :0 -background none -verbose -auth /run/gdm/auth-f...
   │ └ 1515 gdm-session-worker [pam/gdm-password]
```

Systemctl: Start / Stop Service

- systemctl start|stop|restart|try-restart|reload foobar.service

- systemctl kill foobar.service
- systemctl kill -s SIGKILL foobar.service
- systemctl kill -s HUP –kill-who=main crond.service

Systemctl:Enable / disable a service

- systemctl enable foobar.service
- systemctl disable foobar.service
- systemctl mask / unmask foobar.service

Systemctl: misc commands

- systemctl daemon-reload: force reloading on disk configuration. Required after ANY change to unit files.
- systemctl show foobar.unit : dump internal representation of a unit
- systemctl reboot|halt|suspend... : as expected (legacy commands will still work)

journalctl

- By default, will only work on current boot
- To enable on disk persistence, install systemd-logger package (will allow to uninstall other syslogs) or create /var/log/journal
- Agregate all syslog + stdout/stderr entries from all services
- journalctl: output all logs
- Journalctl -b | -b -1: only currently boot | previous boot (v207)
- Journalctl /dev/sda: all logs from this device

Journalctl : how to query

- journalctl _SYSTEMD_UNIT=foobar.service: logs from a particular service
- journalctl _UID=1152 : all logs from this UID
- If several expressions are used, “AND” is used:

  journalctl _EXE=/usr/bin/ntpd _UID=1

- To have “OR”, use +:

  journalctl _EXE=/usr/bin/ntpd + _UID=1

- Often used query have shortcuts :

  -u for _SYSTEMD_UNIT

  path_to_executable for _EXE

  -k for _TRANSPORT=kernel aka dmesg output

Loginctl: session handling

- Logind / loginctl is replacing ConsoleKit
- loginctl [list-sessions]: output all sessions
- loginctl session-status:

```
2 - fcrozat (1000)
           Since: lun. 2013-07-29 11:58:41 CEST; 4h 13min ago
          Leader: 1550 (gdm-session-wor)
            Seat: seat0; vc7
         Display: :0
         Service: gdm-password; type x11; class user
           State: active
          CGroup: systemd:/user/1000.user/2.session
                  ├─ 1550 gdm-session-worker [pam/gdm-password]
                  ├─ 1557 /usr/bin/gnome-keyring-daemon --daemonize
                  ├─ 1560 /usr/bin/gnome-session
```

- `loginctl kill-session|kill-user|terminate-seat <name>`

Systemd-analyze: deep debugging

- systemd-analyze time: boot time
- systemd-analyze blame : what is taking so much time
- systemd-analyze critical-chain [unit]: time-critical chain of units (specified or for the boot)
- systemd-analyze set-log-level info|debug|..: (v207+) change log verbosity (not persistent across reboot)
- systemd-analyze plot : bootchart in svg

Debugging systemd 1/2

- Check journalctl -b output (or journalctl -b _PID=1 for systemd PID1 only output)
- Check journalctl -b -u foobar.service
- Enable debugging output:
  - systemd-analyze set-log-level debug (only for current session)
  - Reboot with “systemd.log_level=debug log_buf_len=1M” or “debug” (v205+)
  - Change /etc/systemd/system.conf (LogLevel=debug)
- For initscript, you might want to enable trace (-x) in their shell

Debugging systemd 2/2

- If you modify any .service file, remember to run systemctl daemon-reload
- To debug shutdown bug (should be less frequent since v198):
  - Create a hook script at /usr/lib/systemd/systemd-shutdown/debug.sh containing :

```
#!/bin/sh

mount -o remount,rw /

dmesg > /shutdown-log.txt

mount -o remount,ro /
```

  - Reboot and check the trace file

References

- Systemd on openSUSE :
  - http://en.opensuse.org/index.php?title=SDB:Systemd
  - http://en.opensuse.org/openSUSE:Systemd_status
- Upstream:
  - http://www.freedesktop.org/wiki/Software/systemd/
  - Check manpages, they are extremely verbose (if something is missing there, it is a bug !)

Questions ?

Thank you.