system initialization needs to support hotplugging and events
services should be monitored
init scripts are full of boilerplate and are hard to get right
execution environment is badly defined and leaky
init scripts and other low-level configuration files are different on every distro
... and what

- initially a modern Linux init system with advanced features for reliable monitoring and controlling services
- its scope has broadened, it’s now a set of basic building blocks for a Linux based operating system
- very well documented
- boot and system state is introspectable and debuggable with the journal providing a log from early system start
- a service can be stopped without leaving runaway children
- often regarded as one monolithic binary it is actually made up of various services, which have a tight integration
- unifies service and hardware management
- SysV init scripts are first class citizens
rsyslog service file

[Unit]
Description=System Logging Service

[Service]
ExecStart=/usr/sbin/rsyslogd -n
Sockets=syslog.socket
StandardOutput=null

[Install]
WantedBy=multi-user.target
Alias=syslog.service

The corresponding SysV init script in comparison is 126 lines of shell script. In a lot of cases it is much worse, e.g. the sendmail init script is 1340 lines long (and that’s not even the craziest one)!
ordering and dependencies

- a dependency does not imply an ordering, needs to be configured explicitly
- dependencies: Wants / Requires / Requisite
- ordering: Before / After
- dependencies of type Wants can be expressed by hooking up the service in a foo.target.wants/ directory via WantedBy
- Wants/After is what you typically want
- you can declare dependencies on LSB/SysV init scripts and vice versa
unit types

- target
- service
- socket
- mount/automount/swap
- path
- timer

---

²http://0pointer.de/public/systemd-man/systemd.unit.html
wheezy - current state

- based on v44, just before the systemd and udev code bases were merged
- 139 releases were skipped to align udev and systemd version numbers
- current is v197, ie. 14 releases behind, a good deal is journal related
- early boot (rcS) is completely “native” i.e. the initscripts package could potentially be uninstalled
- problematic because of the Essential flag (and insserv), so we blacklist those init scripts instead
- core services like udev, dbus or rsyslog have systemd support
wheezy - integration

- 4 main binary packages: systemd, systemd-gui, systemd-sysv, libpam-systemd
- for a general purpose, desktop system you want **systemd** and **libpam-systemd**
- we intercept start/stop/reload requests via the LSB init-functions hook. This covers about 900 out of 1200 init scripts.
- can be installed alongside sysvinit
- systemd is started via boot parameter \texttt{init=/bin/systemd}
- DEMO
- insserv support incomplete, (virtual) facilities defined in insserv.conf.d are not handled
- NFS mounting via /etc/fstab is currently broken
- badly written SysV init scripts causing problems (you see a lot of scary stuff browsing through /etc/init.d)
sysvinit will most likely remain the default for most installations

systemd - udev merge, we will keep a separate udev binary package!

GNOME will start to depend on functionality of systemd though

systemd-logind as replacement for ConsoleKit

what about kFreeBSD?

grub snippet to easily boot systemd
jessie - packaging

- packaging helper tools: extend dh_installinit
- make service, invoke-rc.d and update-rc.d systemd-aware
- keep enabled/disabled state in sync between different init systems
- systemd-to-sysvinit converter (maybe)
- packages can be updated one by one
So you want to add systemd support to your package? Great! Here's what you should know
simple, declarative text files
location: `/etc,/run,/lib)/systemd/system`
packages should use `/lib/systemd/system`
ownen provided by upstream
`pkg-config --variable systemd/systemunitdir systemd`
`./configure --with-systemd/systemunitdir=/lib/systemd/system`

service types ⁴

- Type=
  - forking: PIDFile=/var/run/foo.pid
  - dbus: BusName=org.foo.MyService
  - simple: default
  - oneshot: often used with RemainAfterExit=true
  - notify: requires support by the daemon via sd_notify()

- LSB/SysV services use Type=forking, RemainAfterExit=true, GuessMainPID=no

⁴http://0pointer.de/public/systemd-man/systemd.service.html
service activation

- different triggers to start a service:
  - D-Bus activation
  - socket activation
  - hardware hotplug event (udev)
  - timer based activation
  - target(runlevel) based activation

---

5http://0pointer.de/public/systemd-man/daemon.html
services are started on demand

dbus-daemon forwards start requests to systemd for D-Bus system services

corresponding systemd service is set via SystemdService=

D-Bus services can be enabled/disabled by using an Alias as SystemdService

currently 47 packages shipping 70 D-Bus system services

11 of them have a native systemd service, still lots of low hanging fruit
org.freedesktop.UPower.service ⁶

[D-BUS Service]
Name=org.freedesktop.UPower
Exec=/usr/lib/upower/upowerd
User=root
SystemdService=upower.service
(SystemdService=dbus-org.freedesktop.UPower.service)

⁶/usr/share/dbus-1/system-services/org.freedesktop.UPower.service
[Unit]
Description=Daemon for power management

[Service]
Type=dbus
BusName=org.freedesktop.UPower
ExecStart=/usr/lib/upower/upowerd

[Install]
WantedBy=graphical.target
(Alias=dbus-org.freedesktop.UPower.service)
socket activation

- .service and .socket file names need to match
- hook up the .socket file in socket.target.wants to activate the socket on boot
- systemd will setup the socket and start the service on demand handing over the socket to the daemon process
- daemon needs to support that
- great for lazily starting services but even better for avoiding explicit dependencies
- not a panacea for all types of services though

---

8[http://0pointer.de/public/systemd-man/systemd.socket.html](http://0pointer.de/public/systemd-man/systemd.socket.html)
avoid starting long running processes (via RUN+=) from udev rules files (udev will kill non-forking processes after a timeout)

tag devices and activate the relevant target/service via TAG+=“systemd”,
ENV={SYSTEMD_WANTS}=“foo.target”

predefined targets: bluetooth, smartcard, sound, printer

hook up your service in those targets via WantedBy

---

http://0pointer.de/public/systemd-man/udev.html
bluez - sysvinit only

```
1 ACTION=="add", SUBSYSTEM=="bluetooth",
2 RUN+="/lib/udev/bluez-udev --udev"
3 ACTION=="change", SUBSYSTEM=="bluetooth",
4 RUN+="/lib/udev/bluez-udev --udev"
```
bluez - systemd and sysvinit

```
1  TEST="/sys/fs/cgroup/systemd", GOTO="bluetooth_end"
2  ACTION="add", SUBSYSTEM="bluetooth",
    RUN="/lib/udev/bluez-udev --udev"
3  ACTION="change", SUBSYSTEM="bluetooth",
    RUN="/lib/udev/bluez-udev --udev"
4  LABEL="bluetooth_end"
5
6  SUBSYSTEM="bluetooth", TAG="systemd",
7    ENV{SYSTEMD_WANTS}="bluetooth.target"
```
[Unit]
Description=Bluetooth service
After=syslog.target

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/sbin/bluetoothd -n
StandardOutput=syslog

[Install]
WantedBy=bluetooth.target
SysV init scripts and native services

- can be shipped alongside in the package
- a native service overrides LSB/SysV service given the names match: /etc/init.d/foo → foo.service
- if names don’t match, use an alias or blacklist
- most .service files need to be enabled explicitly
- don’t run systemctl enable in postinst, just ship the symlinks in the package until we have the necessary tooling
create run time directories or files for services

- location: `{/etc,/run,/usr/lib}`/tmpfiles.d/
- packages should use `/usr/lib/tmpfiles.d/`
- duplicated in a lot of SysV init scripts, some only exist for that very purpose
- useful outside of systemd, so we plan to suggest that as a general mechanism
- extend dh_installinit to create the necessary maintainer scripts code

---

<table>
<thead>
<tr>
<th>Type</th>
<th>Path</th>
<th>Mode</th>
<th>UID</th>
<th>GID</th>
<th>Age</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>/run/lock</td>
<td>1777</td>
<td>root</td>
<td>root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>/run/lock/subsys</td>
<td>0755</td>
<td>root</td>
<td>root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>/run/lock/lockdev</td>
<td>0775</td>
<td>root</td>
<td>root</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
tips on writing SysV init scripts

- avoid custom targets/actions
- include /lib/lsb/init-functions, right at the beginning
- avoid sleeps on stop, use start-stop-daemon --retry instead
- restart should be the equivalent to stop + start, don’t do any magic in between
- if your service doesn’t support reload, don’t map it to restart, use force-reload instead
- keep your help message up-to-date, especially wrt reload
- avoid Debian specific config files like /etc/default/$package
- especially, avoid enable/disable flags, use proper interfaces like update-rc.d
conclusion

- wait until jessie is open for development and we have the necessary tools in place (we will announce that in time)
- half-assed systemd support is worse than no support
- talk to us if you have questions, especially if you have a non-trivial service
- your feedback and support is most welcome
- don’t be afraid
resources

- http://www.freedesktop.org/wiki/Software/systemd
- http://0pointer.de/public/systemd-man/
- http://wiki.debian.org/systemd
- IRC: #debian-systemd on irc.debian.org