Systemd

Starting Linux systems with a change
Systemd

NEW
BETTER
FASTER
EASIER

Everyone is DOING it
The development team is LOVED
Systemd

OK. That last one is a bit of a stretch.

OK. Not even close.

OK. Some act like a$$holes.

It's not like Linus is always nice.
Key, I'm *f*cking tired of the fact that you don't fix problems in the code *you* write, so that the kernel then has to work around the problems you cause.

Greg - just for your information, I will *not* be merging any code from Kay into the kernel until this constant pattern is fixed.

This has been going on for *years*, and doesn't seem to be getting any better. This is relevant to you because I have seen you talk about the kdbus patches, and this is a heads-up that you need to keep them separate from other work. Let distributions merge it as they need to and maybe we can merge it once it has been proven to be stable by whatever distro that was willing to play games with the developers.

But I'm not willing to merge something where the maintainer is known to not care about bugs and regressions and then forces people in other projects to fix their project. Because I am *not* willing to take patches from people who don't clean up after their problems, and don't admit that it's their problem to fix.

Kay - one more time: you caused the problem, you need to fix it. None of this "I can do whatever I want, others have to clean up after me" crap.

Linus
Systemd

Old process - Sys V
Intermediate - Upstart
New - systemd
Systemd

• **Sys V**
  - Uses simple scripts
  - Requires a shell
  - Serial process
    • Each unit must finish before the next can start
    • Unrelated units “hang” while a broken start fails
  - 15+ years of experience
Systemd

• **Upstart**
  - Faster than sysV
  - Uses D-bus for internal communications
  - Parallel start up
Systemd

- **Systemd**
  - Highly parallel
  - Uses D-bus
    - Working on kdbus (kernel d-bus)
  - Works well with plymouth boot gui
  - Can restart failed unit
    - Critical unit can be watched and restarted
<table>
<thead>
<tr>
<th>Feature</th>
<th>sysvinit</th>
<th>Upstart</th>
<th>systemd</th>
<th>sysvinit</th>
<th>Upstart</th>
<th>systemd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfacing via D-Bus</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Shell-free bootup</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Modular C-coded early boot services included</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Read-Ahead</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Socket-based Activation</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Socket-based Activation; inetd compatibility</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Bus-based Activation</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Device-based Activation</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Configuration of device dependencies with udev rules</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Path-based Activation (inotify)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Timer-based Activation</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Mount handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>fsck handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Automount handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Swap handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Snapshotting of system state</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>XDG_RUNTIME_DIR Support</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Optionally kills remaining processes of users logging out</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Linux Control Groups Integration</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Audit record generation for started services</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>SELinux integration</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>PAM integration</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Encrypted hard disk handling (LUKS)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>SSL Certificate/LUKS Password handling, including Plymouth, Console, wall(1), TTY and GNOME agents</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Network Loopback device handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Udev handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>System-wide locale handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Console and keyboard setup</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Infrastructure for creating, removing, cleaning up of temporary and volatile files</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Handling for /proc/sys sysctl</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Plymouth integration</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Save/restore random seed</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Static loading of kernel modules</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Automatic serial console handling</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Systemd

- Systemctl
- Journalctl
Systemd

• **Systemctl**
  - Used to start, stop, restart and check status

• **Usage**
  - systemctl start foo.service
  - systemctl restart foo.service
  - systemctl status foo.service
Systemd

- LIVE EXAMPLE!!
  - systemctl status bluetooth.service -l
  - systemctl status bluetooth -l
    - Assumes .service
    - -l provides “long line” support
Systemd

- **Systemctl also used to make service run at boot time**
  - systemctl enable foo

- **Can list all available services**
  - systemctl (messy list but LOTS of data)
  - systemctl list-unit-files (easier to read)
Systemd

- Runlevels are dead
  - Long live run levels!

- Targets are new hotness
  - Rescue ~ single user
  - Multi-user ~ run level 3/networking, no X
  - Graphical ~ Xorg
  - User-definable
    - eg. system-update.target for safe package updates
      - Logs updates with journalctl
Systemd

• How to get a list of the targets
  – systemctl --type=target (shows loaded)
  – systemctl --type=target --all (shows all)

• How to find parts of a target
  – systemctl list-dependencies foo.target

• LIVE DEMO!!!
Systemd

• Changing runlevels targets
  – systemctl isolate foo.target
    • This will kill running services NOT in new target just like telinit foo would do.
    • This will also start additional services as required and expected.
  – For obvious reasons there will not be a live demo :-(
Systemd

• Set target at boot
  – Single user mode is rescue.target
    • Add to kernel line from grub(2):
      systemd.unit=rescue.target
  – Can also use:
    Systemd.unit=runlevel1.target
• **What hogs my startup time?**
  - Systemctl can display startup time data
    • Show aggregate start time
      systemd-analyze
    • Show sorted list of processes
      systemd-analyze blame

• **LIVE DEMO!!**
Systemd

• **systemctl-analyze as a Gantt Chart**
  
  systemctl-analyze plot > test.svg

• **LIVE DEMO!!**
Systemd

- **Verbose debugging of systemd from kernel boot**
  
  - Append to kernel boot line
    
    ```
    systemd.log_level=debug
    systemd.log_target=kmsg
    ```
Systemd

- Systemd equivalent of init files

```ini
file:///usr/lib/systemd/system/bluetooth.service
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/libexec/bluetooth/bluetoothd
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1
[Install]
WantedBy=bluetooth.target
Alias=dbus-org.bluez.service
```
Systemd

• **Automagic restart**
  - WatchdogSec
    • WatchdogSec=10 (seconds wait before action)
  - Restart
    • Restart=on-failure (any exit code other than 0)

<table>
<thead>
<tr>
<th>Restart settings/Exit causes</th>
<th>no</th>
<th>always</th>
<th>on-success</th>
<th>on-failure</th>
<th>on-abnormal</th>
<th>on-abort</th>
<th>on-watchdog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean exit code or signal</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unclean exit code</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unclean signal</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watchdog</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Systemd

• **Journalctl**
  - More data than can be imagined
  - Mostly very usable
  - Often an overload without filtering
  - Easy to filter
**Systemd**

- **Show boot times**
  - Once installed, a (fairly) permanent log of every time the system is booted is kept
    
journalctl -list-boots
  - The most recent boot number is always 0

- **Show logs from a particular command**
  
journalctl _COMM=sshd -b <foo>
  - Pressing <tab> after = will display a list
  - --boot <foo> will show only logs from a particular boot
  - -b <null> is current boot

**LIVE DEMO!!**
Systemd

- Journalctl.conf
  - Set size
  - Set retention time
  - Set location
  - Accept defaults (all are reasonable)

LIVE DEMO!!
Systemd

• A single way journalctl trumps syslog:
  - Show all <foo> records between two dates:
    `journalctl _COMM=<foo> --since “YYYY-MM-DD HH:MM:SS” --until “YYYY-MM-DD HH:MM:SS”`

• LIVE DEMO!!