

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.



Systemd

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



Systemd

	sysvinit	Upstart	systemd		sysvinit	Upstart	systemd
Interfacing via D-Bus	no	yes	yes	Automatic serial console handling	no	no	yes
Shell-free bootup	no	no	yes	Unique Machine ID handling	no	no	yes
Modular C coded early boot services included	no	no	yes	Dynamic host name and machine meta data handling	no	no	yes
Read-Ahead	no	no	yes	Reliable termination of services	no	no	yes
Socket-based Activation	no	no	yes	Early boot /dev/log logging	no	no	yes
Socket-based Activation: inetd compatibility	no	no	yes	Minimal kmsg-based syslog daemon for embedded use	no	no	yes
Bus-based Activation	no	no	yes	Respawning on service crash without losing connectivity	no	no	yes
Device-based Activation	no	no	yes	Gapless service upgrades	no	no	yes
Configuration of device dependencies with udev rules	no	no	yes	Graphical UI	no	no	yes
Path-based Activation (inotify)	no	no	yes	Built-In Profiling and Tools	no	no	yes
Timer-based Activation	no	no	yes	Instantiated services	no	yes	yes
Mount handling	no	no	yes	PolicyKit integration	no	no	yes
fsck handling	no	no	yes	Remote access/Cluster support built into client tools	no	no	yes
Quota handling	no	no	yes	Can list all processes of a service	no	no	yes
Automount handling	no	no	yes	Can identify service of a process	no	no	yes
Swap handling	no	no	yes	Automatic per-service CPU cgroups to even out CPU usage between them	no	no	yes
Snapshotting of system state	no	no	yes	Automatic per-user cgroups	no	no	yes
XDG_RUNTIME_DIR Support	no	no	yes	SysV compatibility	yes	yes	yes
Optionally kills remaining processes of users logging out	no	no	yes	SysV services controllable like native services	yes	no	yes
Linux Control Groups Integration	no	no	yes	SysV-compatible /dev/initctl	yes	no	yes
Audit record generation for started services	no	no	yes	Reexecution with full serialization of state	yes	no	yes
SELinux integration	no	no	yes	Interactive boot-up	no	no	yes
PAM integration	no	no	yes	Container support (as advanced chroot() replacement)	no	no	yes
Encrypted hard disk handling (LUKS)	no	no	yes	Dependency-based bootup	no	no	yes
SSL Certificate/LUKS Password handling, including Plymouth, Console, wall(1), TTY and GNOME agents	no	no	yes	Disabling of services without editing files	yes	no	yes
Network Loopback device handling	no	no	yes	Masking of services without editing files	no	no	yes
binfmt_misc handling	no	no	yes	Robust system shutdown within PID 1	no	no	yes
System-wide locale handling	no	no	yes	Built-in kexec support	no	no	yes
Console and keyboard setup	no	no	yes	Dynamic service generation	no	no	yes
Infrastructure for creating, removing, cleaning up of temporary and volatile files	no	no	yes	Upstream support in various other OS components	yes	no	yes
Handling for /proc/sys sysctl	no	no	yes	Service files compatible between distributions	no	no	yes
Plymouth integration	no	yes	yes	Signal delivery to services	no	no	yes
Save/restore random seed	no	no	yes	Reliable termination of user sessions before shutdown	no	no	yes
Static loading of kernel modules	no	no	yes	utmp/wtmp support	yes	yes	yes
Automatic serial console handling	no	no	yes	Easily writable, extensible and parseable service files, suitable for manipulation with enterprise management tools	no	no	yes



Systemd

- **Systemd command features**
 - 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`



Systemd

- **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**

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)

Restart settings/Exit causes	no	always	on-success	on-failure	on-abnormal	on-abort	on-watchdog
Clean exit code or signal		X	X				
Unclean exit code		X		X			
Unclean signal		X		X	X	X	
Timeout		X		X	X		
Watchdog		X		X	X		X



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!!**

