SYSADMIN

System administration technologies brought to you from the coalface of Linux.

Jonathan Roberts dropped out of an MA in Theology to work with Linux. A Fedora advocate and systems administrator, we hear his calming tones whenever we're stuck with something hard.

This is the first issue of *Linux Voice*, and I'm so pleased to be contributing to it along with Graham, Andrew, Mike and Ben. It feels new and exciting, with all four of them having poured so much enthusiasm into the project, it can only work out well.

When Graham asked me to write the first two pages of this new sysadmin section, I started looking around for ideas, and I noticed that in the same way *Linux Voice* represents a kind of transformation of a well loved thing, so Linux itself seems to be undergoing a similar transformation.

New technologies, like systemd (and its many associated subprojects, including journald and logind), btrfs, cgroups and kdbus are slowly replacing older technologies and approaches that many people have long assumed to be synonymous with Linux.

Many sysadmins have been ignoring these new technologies, in part because of all the controversies surrounding them, and in part because of inertia. But the times, they are a changing.

Something better change

With the announcement of Red Hat Enterprise Linux 7 Beta, with the emerging plans in Debian to replace its init system, and openSuse having seriously debated btrfs as its default filesystem, the time has come for sysadmins to start taking these technologies seriously, as the next round of major distro releases are certainly going to include some combination of them.

So, a new and exciting magazine and a new and exciting set of technologies – seems like the perfect chance to take a closer look. Look out for my coverage of some of these new technologies over the next few issues of Linux Voice.

systemd

Don't fear change

The init replacement for RHEL 7 and SUSE Enterprise Linux 12.

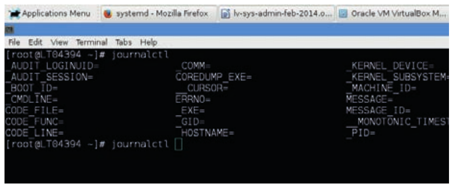
The arrival of a new Linux init system has been a long time coming. It was back in 2006 that Upstart was introduced to Ubuntu, and around the same time that Fedora and others also started experimenting with new init systems. The reasons then are much the same as the reasons now – sysvinit is old and doesn't do everything a modern distribution needs it to. More specifically:

- **sysvinit** can't take account of hot-pluggable hardware devices and filesystems, such as network mounts or USB sticks.
- **sysvinit** doesn't provide sufficient supervision of processes, allowing double forked processes to become orphaned.
- **sysvinit** can't parallelise boot services effectively, so it is slow.
- **sysvinit** startup scripts are difficult to write, difficult to debug and can't easily be shared between distributions – the Sendmail init script is over 1,000 lines long!

Systemd fixes these problems and introduces a number of new features that make the case for it even more compelling. Rather than explaining in great detail how systemd works or how it fixes these problems (there's plenty of information on that in **http://0pointer.de/blog/projects/systemd.html**), we're going to take a look at a few key features of systemd that might make sysadmins look forward to systemd, rather than dread having to learn a new tool.

Configuration file format

As mentioned above, in sysvinit systems, configuration of services was complex and error-prone. They were usually configured through a combination of arcane Bash scripts in **/etc/init.d** and some environmental settings in **/etc/sysconfig** or **/etc/defaults**. These init scripts often did awful amounts of work, such as echoing service status to the console and managing lock files, which were repeated in almost every init script.



Most of systemd's tools feature tab-completed sub-commands, which is indicative of the efort that's gone into making it a pleasure to use.

Systemd removes the need for much of the complexity in these init scripts by handling service status echoes and suchlike itself. This means it can switch complex procedural Bash code for a clear, declarative configuration file. For example, here's the configuration for the syslog service on my Fedora system:

```
[Unit]
Description=System Logging Service
[Service]
EnvironmentFile=-/etc/sysconfig/rsyslog
ExecStart=/sbin/rsyslogd -n $SYSLOGD_OPTIONS
Sockets=syslog.socket
StandardOutput=null
[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

All of the configuration options available in these files are extremely well documented (systemd as a whole has some of the best docs around) – see **man systemd.unit** or **man systemd.service** for details.

What's more, if you had to modify a sysvinit file, you'd have to be careful when it came to package upgrades etc that your changes wouldn't get overwritten. With systemd, unit files get packaged into **/usr/lib/systemd/system**, but if you want to replace the default with your own, you can put them in **/etc/systemd/system** and whatever is there will take precedence over the defaults.

You can even include other unit configuration files in yours, so you can easily extend the default configuration:

```
include /usr/lib/systemd/system/nfs-secure.service
#extra conf goes here
```

Resource controls

Why would you want to extend a service configuration like that? Well, systemd launches all processes inside their own cgroup (and all processes spawned from this end up in the same cgroup – this is also useful as it stops double forking processes from orphaning themselves), so you can take advantage of this to use cgroups to limit the resources that each process (and its child processes) can consume.

Systemd not only makes this possible by the way it spawns processes, but it also makes it easy by exposing many of the most common bits of functionality in configuration directives. For instance, you could limit the amount of CPU a process gets by dropping in a new unit configuration file to **/etc/systemd/system** and adding:

```
[Service]
CpuShares=200
```

By default, systemd gives all processes (well, cgroups), an equal share of the processor (1024). By setting **CpuShares** to 200, you're restricting this process to about 20% of CPU time. What's more, this isn't applied just to the parent process but to all child processes. So if you have Apache running with many hundreds of spawned CGI processes, this would restrict all of those processes to about 20% of CPU time.

With the configuration file in place, you'd just need to tell systemd to reload it, with **systemctl daemon-reload**, and then restart the service, with **systemctl restart httpd.service**, for example.

You can also set memory limits (MemoryLimit) and IO limits (BlockIOWeight). See **man systemd.resource-control** for further details. There are also any number of security settings that can be put in the configuration files like this. For example, you can restrict a service from accessing a particular device, make individual directory trees inaccessible or read-only, create a private **/tmp** directory for a service or even stop a service, and all its child processes, from accessing the network. In the example below, we've configured a service to have a private **/tmp** directory. See how simple it is:

```
[Service]
PrivateTmp=yes
```

Journal

Another aspect of systemd is that it collects all output from processes it starts – whether that's through **syslog()** calls, messages emitted to STDOUT or STDERR, initial RAM disk or kernel messages. It does this through one of its components, journald. To see the contents of the logs, you can just type **journalctl** as root and you'll get the results displayed, just as if you were looking at the contents of **/var/log/messages** or similar.

This default view gives you some simple improvements over the traditional techniques, however. Error and higher priority messages are in red, notice and warning are bold, timestamps are in your local timezone. These are fairly cosmetic improvements. What sets journald apart is that the logs are kept on disk in a binary format, which means that the journal entries can be indexed on all fields, making them quick to search and easy to filter. For example:

```
journalctl PRIORITY=7 –since=yesterday
```

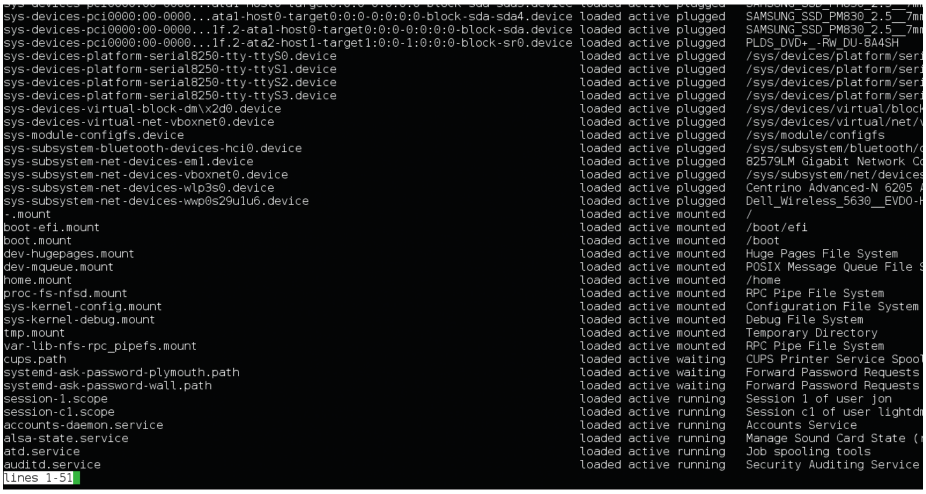
Will show all messages of debug priority received by the journal since yesterday. If you tried to do this with standard syslog messages or the like, you'd have to concoct your own grep or awk command, or hook it in to a system like Logstash or Splunk.

There are loads of fields on which you can filter that come direct from the messages themselves, as well as a lot of metadata that the journal inputs in to each log message itself, including SELinux context, hostname, transport etc. To see the full details, you can read **man systemd.journal-fields**. Journalctl even features tab completion of possible field names, so you can get a quick look too by typing

```
journalctl <tab><tab>.
```

There are many other great features in systemd that, if you take the time to look around, will make your life as a sysadmin better. We hope this article has at least given you the motivation to take a closer look.

> **"Another aspect of systemd is that it collects all output from processes that it starts."**



```
sys-devices-pci0000:00-0000...ata1-host0-target0:0:0-0:0:0:0-block-sda-sda4.device loaded active plugged   SAMSUNG_SSD_PM830_2.5__7mm
sys-devices-pci0000:00-0000...1f.2-ata1-host0-target0:0:0-0:0:0:0-block-sda.device loaded active plugged   SAMSUNG_SSD_PM830_2.5__7mm
sys-devices-pci0000:00-0000...1f.2-ata2-host1-target1:0:0-1:0:0:0-block-sr0.device loaded active plugged   PLDS_DVD+_-RW_DU-8A4SH
sys-devices-platform-serial8250-tty-ttyS0.device                                   loaded active plugged   /sys/devices/platform/seri
sys-devices-platform-serial8250-tty-ttyS1.device                                   loaded active plugged   /sys/devices/platform/seri
sys-devices-platform-serial8250-tty-ttyS2.device                                   loaded active plugged   /sys/devices/platform/seri
sys-devices-platform-serial8250-tty-ttyS3.device                                   loaded active plugged   /sys/devices/platform/seri
sys-devices-virtual-block-dm\x2d0.device                                           loaded active plugged   /sys/devices/virtual/block
sys-devices-virtual-net-vboxnet0.device                                            loaded active plugged   /sys/devices/virtual/net/v
sys-module-configfs.device                                                         loaded active plugged   /sys/module/configfs
sys-subsystem-bluetooth-devices-hci0.device                                        loaded active plugged   /sys/subsystem/bluetooth/d
sys-subsystem-net-devices-em1.device                                               loaded active plugged   82579LM Gigabit Network Co
sys-subsystem-net-devices-vboxnet0.device                                          loaded active plugged   /sys/subsystem/net/devices
sys-subsystem-net-devices-wlp3s0.device                                            loaded active plugged   Centrino Advanced-N 6205 A
sys-subsystem-net-devices-wwp0s29u1u6.device                                       loaded active plugged   Dell_Wireless_5630__EVDO-H
-.mount                                                                            loaded active mounted   /
boot-efi.mount                                                                     loaded active mounted   /boot/efi
boot.mount                                                                         loaded active mounted   /boot
dev-hugepages.mount                                                                loaded active mounted   Huge Pages File System
dev-mqueue.mount                                                                   loaded active mounted   POSIX Message Queue File S
home.mount                                                                         loaded active mounted   /home
proc-fs-nfsd.mount                                                                 loaded active mounted   RPC Pipe File System
sys-kernel-config.mount                                                            loaded active mounted   Configuration File System
sys-kernel-debug.mount                                                             loaded active mounted   Debug File System
tmp.mount                                                                          loaded active mounted   Temporary Directory
var-lib-nfs-rpc_pipefs.mount                                                       loaded active mounted   RPC Pipe File System
cups.path                                                                          loaded active waiting   CUPS Printer Service Spool
systemd-ask-password-plymouth.path                                                 loaded active waiting   Forward Password Requests
systemd-ask-password-wall.path                                                     loaded active waiting   Forward Password Requests
session-1.scope                                                                    loaded active running   Session 1 of user jon
session-c1.scope                                                                   loaded active running   Session c1 of user lightdm
accounts-daemon.service                                                            loaded active running   Accounts Service
alsa-state.service                                                                 loaded active running   Manage Sound Card State (r
atd.service                                                                        loaded active running   Job spooling tools
auditd.service                                                                     loaded active running   Security Auditing Service
lines 1-51
```

sysctemctl enables a user to easily inspect what units (services, in systemd speak) are loaded on their system and what their current status is.

The other contender for the next generation of Linux init systems, Upstart, is the Ubuntu's choice and involved in a hotly contested debate in the Debian project over what they should choose.