Pid Eins

レナート
لينارت

POSTED ON SA 26 JANUAR 2013

## The Biggest Myths

Since we first proposed systemd for inclusion in the distributions it has been frequently discussed in many forums, mailing lists and conferences. In these discussions one can often hear certain myths about systemd, that are repeated over and over again, but certainly don't gain any truth by constant repetition. Let's take the time to debunk a few of them:

1. **Myth: systemd is monolithic.**

   If you build systemd with all configuration options enabled you will build 69 individual binaries. These binaries all serve different tasks, and are neatly separated for a number of reasons. For example, we designed systemd with security in mind, hence most daemons run at minimal privileges (using kernel capabilities, for example) and are responsible for very specific tasks only, to minimize their security surface and impact. Also, systemd parallelizes the boot more than any prior solution. This parallization happens by running more processes in parallel. Thus it is essential that systemd is nicely split up into many binaries and thus processes. In fact, many of these binaries[1] are separated out so nicely, that they are very useful outside of systemd, too.

A package involving 69 individual binaries can hardly be called *monolithic*. What is different from prior solutions however, is that we ship more components in a single tarball, and maintain them upstream in a single repository with a unified release cycle.

2. **Myth: systemd is about speed.**

   Yes, systemd is fast (A pretty complete userspace boot-up in ~900ms, anyone?), but that's primarily just a side-effect of doing things *right*. In fact, we never really sat down and optimized the last tiny bit of performance out of systemd. Instead, we actually frequently knowingly picked the slightly slower code paths in order to keep the code more readable. This doesn't mean being fast was irrelevant for us, but reducing systemd to its speed is certainly quite a misconception, since that is certainly not anywhere near the top of our list of goals.

3. **Myth: systemd's fast boot-up is irrelevant for servers.**

   That is just completely not true. Many administrators actually are keen on reduced downtimes during maintenance windows. In High Availability setups it's kinda nice if the failed machine comes back up really fast. In cloud setups with a large number of VMs or containers the price of slow boots multiplies with the number of instances. Spending minutes of CPU and IO on really slow boots of hundreds of VMs or containers reduces your system's density drastically, heck, it even costs you more energy. Slow boots can be quite financially expensive. Then, fast booting of containers allows you to implement a logic such as socket activated containers, allowing you to drastically increase the density of your cloud system.

   Of course, in many server setups boot-up is indeed irrelevant, but systemd is supposed to cover the whole range. And yes, I am aware that often it is the server firmware that costs the most time at boot-up, and the OS anyways fast compared to that, but well, systemd is still supposed to cover the whole range (see above...), and no, not all servers have such bad firmware, and certainly not VMs and containers, which are servers of a kind, too.[2]

4. **Myth: systemd is incompatible with shell scripts.**

   This is entirely bogus. *We* just don't use them for the boot process, because we believe they aren't the best tool for that specific purpose, but that doesn't mean systemd was incompatible with them. You can easily run shell scripts as systemd services, heck, you can run scripts written in *any* language as systemd services, systemd doesn't care the slightest bit what's inside your executable. Moreover, we heavily use shell scripts for our own purposes, for installing, building, testing systemd. And you can stick your scripts in the early boot process, use them for normal services, you can run them at latest shutdown, there are practically no limits.

5. **Myth: systemd is difficult.**

   This also is entire non-sense. A systemd platform is actually much simpler than traditional Linuxes because it unifies system objects and their dependencies as systemd units. The configuration file language is very simple, and redundant configuration files we got rid of. We provide uniform tools for much of the configuration of the system. The system is much less conglomerate than traditional Linuxes are. We also have pretty comprehensive documentation (all linked from the homepage) about pretty much every detail of systemd, and this not only covers admin/user-facing interfaces, but also developer APIs.

   systemd certainly comes with a learning curve. Everything does. However, we like to believe that it is actually simpler to understand systemd than a Shell-based boot for most people. Surprised we say that? Well, as it turns out, Shell is not a pretty language to learn, it's syntax is arcane and complex. systemd unit files are substantially easier to understand, they do not expose a programming language, but are simple and declarative by nature. That all said, if you are experienced in shell, then yes, adopting systemd will take a bit of learning.

   To make learning easy we tried hard to provide the maximum compatibility to previous solutions. But not only that, on many distributions you'll find that some of the traditional tools will now even tell you -- while executing what you are asking for -- how you could do it with the newer tools instead, in a possibly nicer way.

Anyway, the take-away is probably that systemd is probably as simple as such a system can be, and that we try hard to make it easy to learn. But yes, if you know sysvinit then adopting systemd will require a bit learning, but quite frankly if you mastered sysvinit, then systemd should be easy for you.

6. **Myth: systemd is not modular.**

   Not true at all. At compile time you have a number of `configure` switches to select what you want to build, and what not. And we document how you can select in even more detail what you need, going beyond our configure switches.

   This modularity is not totally unlike the one of the Linux kernel, where you can select many features individually at compile time. If the kernel is modular enough for you then systemd should be pretty close, too.

7. **Myth: systemd is only for desktops.**

   That is certainly not true. With systemd we try to cover pretty much the same range as Linux itself does. While we care for desktop uses, we also care pretty much the same way for server uses, and embedded uses as well. You can bet that Red Hat wouldn't make it a core piece of RHEL7 if it wasn't the best option for managing services on servers.

   People from numerous companies work on systemd. Car manufactureres build it into cars, Red Hat uses it for a server operating system, and GNOME uses many of its interfaces for improving the desktop. You find it in toys, in space telescopes, and in wind turbines.

   Most features I most recently worked on are probably relevant primarily on servers, such as container support, resource management or the security features. We cover desktop systems pretty well already, and there are number of companies doing systemd development for embedded, some even offer consulting services in it.

8. **Myth: systemd was created as result of the NIH syndrome.**

This is not true. Before we began working on systemd we were pushing for Canonical's Upstart to be widely adopted (and Fedora/RHEL used it too for a while). However, we eventually came to the conclusion that its design was inherently flawed at its core (at least in our eyes: most fundamentally, it leaves dependency management to the admin/developer, instead of solving this hard problem in code), and if something's wrong in the core you better replace it, rather than fix it. This was hardly the only reason though, other things that came into play, such as the licensing/contribution agreement mess around it. NIH wasn't one of the reasons, though...[3]

9. **Myth: systemd is a freedesktop.org project.**

   Well, systemd is certainly hosted at fdo, but freedesktop.org is little else but a repository for code and documentation. Pretty much any coder can request a repository there and dump his stuff there (as long as it's somewhat relevant for the infrastructure of free systems). There's no cabal involved, no "standardization" scheme, no project vetting, nothing. It's just a nice, free, reliable place to have your repository. In that regard it's a bit like SourceForge, github, kernel.org, just not commercial and without over-the-top requirements, and hence a good place to keep our stuff.

   So yes, we host our stuff at fdo, but the implied assumption of this myth in that there was a group of people who meet and then agree on how the future free systems look like, is entirely bogus.

10. **Myth: systemd is not UNIX.**

    There's certainly some truth in that. systemd's sources do not contain a single line of code originating from original UNIX. However, we derive inspiration from UNIX, and thus there's a ton of UNIX in systemd. For example, the UNIX idea of "everything is a file" finds reflection in that in systemd all services are exposed at runtime in a kernel file system, the `cgroupfs`. Then, one of the original features of UNIX was multi-seat support, based on built-in terminal support. Text terminals are hardly the state of the art how you interface with your computer these days however. With systemd we brought native multi-seat support back, but this time with full support for today's hardware, covering graphics, mice, audio, webcams and more, and all that fully automatic, hotplug-capable and without configuration. In fact the design of systemd as a suite of integrated tools that each have their individual purposes but when used together are more than just the sum of the parts, that's pretty much at the core of UNIX philosophy. Then, the way our project is handled (i.e. maintaining much of the core OS in a single git repository) is much closer to the BSD model (which is a true UNIX, unlike Linux) of doing things (where most of the core OS is kept in a single CVS/SVN repository) than things on Linux ever were.

Ultimately, UNIX is something different for everybody. For us systemd maintainers it is something we derive inspiration from. For others it is a religion, and much like the other world religions there are different readings and understandings of it. Some define UNIX based on specific pieces of code heritage, others see it just as a set of ideas, others as a set of commands or APIs, and even others as a definition of behaviours. Of course, it is impossible to ever make all these people happy.

Ultimately the question whether something is UNIX or not matters very little. Being technically excellent is hardly exclusive to UNIX. For us, UNIX is a major influence (heck, the biggest one), but we also have other influences. Hence in some areas systemd will be very UNIXy, and in others a little bit less.

11. **Myth: systemd is complex.**

    There's certainly some truth in that. Modern computers are complex beasts, and the OS running on it will hence have to be complex too. However, systemd is certainly not more complex than prior implementations of the same components. Much rather, it's simpler, and has less redundancy (see above). Moreover, building a simple OS based on systemd will involve much fewer packages than a traditional Linux did. Fewer packages makes it easier to build your system, gets rid of interdependencies and of much of the different behaviour of every component involved.

12. **Myth: systemd is bloated.**

    Well, *bloated* certainly has many different definitions. But in most definitions systemd is probably the opposite of bloat. Since systemd components share a common code base, they tend to share much more code for common code paths. Here's an example: in a traditional Linux setup, sysvinit, start-stop-daemon, inetd, cron, dbus, all implemented a scheme to execute processes with various configuration options in a certain, hopefully clean environment. On systemd the code paths for all of this, for the configuration parsing, as well as the actual execution is shared. This means less code, less place for mistakes, less memory and cache pressure, and is thus a very good thing. And as a side-effect you actually get a ton more functionality for it...

As mentioned above, systemd is also pretty modular. You can choose at build time which components you need, and which you don't need. People can hence specifically choose the level of "bloat" they want.

When you build systemd, it only requires three dependencies: glibc, libcap and dbus. That's it. It can make use of more dependencies, but these are entirely optional.

So, yeah, whichever way you look at it, it's really not *bloated*.

13. **Myth: systemd being Linux-only is not nice to the BSDs.**

    Completely wrong. The BSD folks are pretty much uninterested in systemd. If systemd was portable, this would change nothing, they still wouldn't adopt it. And the same is true for the other Unixes in the world. Solaris has SMF, BSD has their own "rc" system, and they always maintained it separately from Linux. The init system is very close to the core of the entire OS. And these other operating systems hence define themselves among other things by their core userspace. The assumption that they'd adopt our core userspace if we just made it portable, is completely without any foundation.

14. **Myth: systemd being Linux-only makes it impossible for Debian to adopt it as default.**

    Debian supports non-Linux kernels in their distribution. systemd won't run on those. Is that a problem though, and should that hinder them to adopt system as default? Not really. The folks who ported Debian to these other kernels were willing to invest time in a massive porting effort, they set up test and build systems, and patched and built numerous packages for their goal. The maintainance of both a systemd unit file and a classic init script for the packaged services is a negligable amount of work compared to that, especially since those scripts more often than not exist already.

15. **Myth: systemd could be ported to other kernels if its maintainers just wanted to.**

    That is simply not true. Porting systemd to other kernel is not feasible. We just use too many Linux-specific interfaces. For a few one might find replacements on other kernels, some features one might want to turn off, but for most this is nor really possible. Here's a small, very incomprehensive list: `cgroups`, `fanotify`, `umount2()`, `/proc/self/mountinfo` (including notification), `/dev/swaps` (same), `udev`, `netlink`, the structure of `/sys`, `/proc/$PID/comm`, `/proc/$PID/cmdline`, `/proc/$PID/loginuid`, `/proc/$PID/stat`, `/proc/$PID/session`, `/proc/$PID/exe`, `/proc/$PID/fd`, `tmpfs`, `devtmpfs`, capabilities, namespaces of all kinds, various `prctl()s`, numerous `ioctls`, the `mount()` system call and its semantics, `selinux`, `audit`, `inotify`, `statfs`, `O_DIRECTORY`, `O_NOATIME`, `/proc/$PID/root`, `waitid()`, `SCM_CREDENTIALS`, `SCM_RIGHTS`, `mkostemp()`, `/dev/input`, ...

    And no, if you look at this list and pick out the few where you can think of obvious counterparts on other kernels, then think again, and look at the others you didn't pick, and the complexity of replacing them.

16. **Myth: systemd is not portable for no reason.**

    Non-sense! We use the Linux-specific functionality because we need it to implement what we want. Linux has so many features that UNIX/POSIX didn't have, and we want to empower the user with them. These features are incredibly useful, but only if they are actually exposed in a friendly way to the user, and that's what we do with systemd.

17. **Myth: systemd uses binary configuration files.**

    No idea who came up with this crazy myth, but it's absolutely not true. systemd is configured pretty much exclusively via simple text files. A few settings you can also alter with the kernel command line and via environment variables. There's nothing binary in its configuration (not even XML). Just plain, simple, easy-to-read text files.

18. **Myth: systemd is a feature creep.**

Well, systemd certainly covers more ground that it used to. It's not just an init system anymore, but the basic userspace building block to build an OS from, but we carefully make sure to keep most of the features optional. You can turn a lot off at compile time, and even more at runtime. Thus you can choose freely how much feature creeping you want.

19. **Myth: systemd forces you to do something.**

    systemd is not the mafia. It's Free Software, you can do with it whatever you want, and that includes not using it. That's pretty much the opposite of "forcing".

20. **Myth: systemd makes it impossible to run syslog.**

    Not true, we carefully made sure when we introduced the journal that all data is also passed on to any syslog daemon running. In fact, if something changed, then only that syslog gets more complete data now than it got before, since we now cover early boot stuff as well as STDOUT/STDERR of any system service.

21. **Myth: systemd is incompatible.**

    We try very hard to provide the best possible compatibility with sysvinit. In fact, the vast majority of init scripts should work just fine on systemd, unmodified. However, there actually are indeed a few incompatibilities, but we try to document these and explain what to do about them. Ultimately every system that is not actually sysvinit itself will have a certain amount of incompatibilities with it since it will not share the exect same code paths.

    It is our goal to ensure that differences between the various distributions are kept at a minimum. That means unit files usually work just fine on a different distribution than you wrote it on, which is a big improvement over classic init scripts which are very hard to write in a way that they run on multiple Linux distributions, due to numerous incompatibilities between them.

22. **Myth: systemd is not scriptable, because of its D-Bus use.**

    Not true. Pretty much every single D-Bus interface systemd provides is also available in a command line tool, for example in `systemctl`, `loginctl`, `timedatectl`, `hostnamectl`, `localectl` and suchlike.

You can easily call these tools from shell scripts, they open up pretty much the entire API from the command line with easy-to-use commands.

That said, D-Bus actually has bindings for almost any scripting language this world knows. Even from the shell you can invoke arbitrary D-Bus methods with dbus-send or gdbus. If anything, this improves scriptability due to the good support of D-Bus in the various scripting languages.

23. **Myth: systemd requires you to use some arcane configuration tools instead of allowing you to edit your configuration files directly.**

    Not true at all. We offer some configuration tools, and using them gets you a bit of additional functionality (for example, command line completion for all settings!), but there's no need at all to use them. You can always edit the files in question directly if you wish, and that's fully supported. Of course sometimes you need to explicitly reload configuration of some daemon after editing the configuration, but that's pretty much true for most UNIX services.

24. **Myth: systemd is unstable and buggy.**

    Certainly not according to our data. We have been monitoring the Fedora bug tracker (and some others) closely for a long long time. The number of bugs is very low for such a central component of the OS, especially if you discount the numerous RFE bugs we track for the project. We are pretty good in keeping systemd out of the list of blocker bugs of the distribution. We have a relatively fast development cycle with mostly incremental changes to keep quality and stability high.

25. **Myth: systemd is not debuggable.**

    False. Some people try to imply that the shell was a good debugger. Well, it isn't really. In systemd we provide you with actual debugging features instead. For example: interactive debugging, verbose tracing, the ability to mask any component during boot, and more. Also, we provide documentation for it.

    It's certainly well debuggable, we needed that for our own development work, after all. But we'll grant you one thing: it uses different debugging tools, we believe more appropriate ones for the purpose, though.

26. **Myth: systemd makes changes for the changes' sake.**

    Very much untrue. We pretty much exclusively have technical reasons for the changes we make, and we explain them in the various pieces of documentation, wiki pages, blog articles, mailing list announcements. We try hard to avoid making incompatible changes, and if we do we try to document the why and how in detail. And if you wonder about something, just ask us!

27. **Myth: systemd is a Red-Hat-only project, is private property of some smart-ass developers, who use it to push their views to the world.**

    Not true. Currently, there are 16 hackers with commit powers to the systemd git tree. Of these 16 only six are employed by Red Hat. The 10 others are folks from ArchLinux, from Debian, from Intel, even from Canonical, Mandriva, Pantheon and a number of community folks with full commit rights. And they frequently commit big stuff, major changes. Then, there are 374 individuals with patches in our tree, and they too came from a number of different companies and backgrounds, and many of those have way more than one patch in the tree. The discussions about where we want to take systemd are done in the open, on our IRC channel (`#systemd` on freenode, you are always weclome), on our mailing list, and on public hackfests (such as our next one in Brno, you are invited). We regularly attend various conferences, to collect feedback, to explain what we are doing and why, like few others do. We maintain blogs, engage in social networks (we actually have some pretty interesting content on Google+, and our Google+ Community is pretty alive, too.), and try really hard to explain the why and the how how we do things, and to listen to feedback and figure out where the current issues are (for example, from that feedback we compiled this lists of often heard myths about systemd...).

    What most systemd contributors probably share is a rough idea how a good OS should look like, and the desire to make it happen. However, by the very nature of the project being Open Source, and rooted in the community systemd is just what people want it to be, and if it's not what they want then they can drive the direction with patches and code, and if that's not feasible, then there are numerous other options to use, too, systemd is never exclusive.

One goal of systemd is to unify the dispersed Linux landscape a bit. We try to get rid of many of the more pointless differences of the various distributions in various areas of the core OS. As part of that we sometimes adopt schemes that were previously used by only one of the distributions and push it to a level where it's the default of systemd, trying to gently push everybody towards the same set of basic configuration. This is never exclusive though, distributions can continue to deviate from that if they wish, however, if they end-up using the well-supported default their work becomes much easier and they might gain a feature or two. Now, as it turns out, more frequently than not we actually adopted schemes that where Debianisms, rather than Fedoraisms/Redhatisms as best supported scheme by systemd. For example, systems running systemd now generally store their hostname in `/etc/hostname`, something that used to be specific to Debian and now is used across distributions.

One thing we'll grant you though, we sometimes can be smart-asses. We try to be prepared whenever we open our mouth, in order to be able to back-up with facts what we claim. That might make us appear as smart-asses.

But in general, yes, some of the more influental contributors of systemd work for Red Hat, but they are in the minority, and systemd is a healthy, open community with different interests, different backgrounds, just unified by a few rough ideas where the trip should go, a community where code and its design counts, and certainly not company affiliation.

28. **Myth: systemd doesn't support `/usr` split from the root directory.**

Non-sense. Since its beginnings systemd supports the `--with-rootprefix=` option to its `configure` script which allows you to tell systemd to neatly split up the stuff needed for early boot and the stuff needed for later on. All this logic is fully present and we keep it up-to-date right there in systemd's build system.

Of course, we still don't think that actually booting with `/usr` unavailable is a good idea, but we support this just fine in our build system. This won't fix the inherent problems of the scheme that you'll encounter all across the board, but you can't blame that on systemd, because in systemd we support this just fine.

29. **Myth: systemd doesn't allow your to replace its components.**

    Not true, you can turn off and replace pretty much any part of systemd, with very few exceptions. And those exceptions (such as journald) generally allow you to run an alternative side by side to it, while cooperating nicely with it.

30. **Myth: systemd's use of D-Bus instead of sockets makes it intransparent.**

    This claim is already contradictory in itself: D-Bus uses sockets as transport, too. Hence whenever D-Bus is used to send something around, a socket is used for that too. D-Bus is mostly a standardized serialization of messages to send over these sockets. If anything this makes it more transparent, since this serialization is well documented, understood and there are numerous tracing tools and language bindings for it. This is very much unlike the usual homegrown protocols the various classic UNIX daemons use to communicate locally.

Hmm, did I write I just wanted to debunk a "few" myths? Maybe these were more than just a few... Anyway, I hope I managed to clear up a couple of misconceptions. Thanks for your time.

**Footnotes**

[1] For example, `systemd-detect-virt`, `systemd-tmpfiles`, `systemd-udevd` are.

[2] Also, we are trying to do our little part on maybe making this better. By exposing boot-time performance of the firmware more prominently in systemd's boot output we hope to shame the firmware writers to clean up their stuff.

[3] And anyways, guess which project includes a library "lib*nih*" -- Upstart or systemd?[4]

[4] Hint: it's not systemd!

Category: projects

← BACK TO INDEX