Half an year on Alpine: just musl aside

2025-08-31 20:05:26

Warning: *Contains tongue-in-cheek language that might feel provocative if you have invested part of your identity into your software choices.*[1]

For roughly six months now, I've been running Alpine on my laptop as a daily driver.

Before that, I was running Void. I like how blazing fast Void is to start up and shut down, has an excellent package management system and its init system, runit, is beautifully integrated. Alpine also fits this description.

When I first learned about Void, I read it described itself as *"rolling but stable"*, meaning you get new package versions often, but not to the point of breakage. While I never had an upgrade break a system, it still bothered me to upgrade and suddenly have sound or Bluetooth not work anymore, or have the system Python version change frequently, leading to all sorts of complaints from this or that application I didn't even know was a Python program.

I wanted to go back to a more stable distribution, but I also wanted a simpler, faster init system than systemd. Alpine, which has fixed releases every six months, eventually rose as an alternative.

▶ Expand for an overly detailed rundown of how I narrowed down my options

Choosing a path forward

So, prompted by a charger dying in the beginning of this year, which put a machine out of work, I decided to take the opportunity to try another system with a slower release cycle and started weighing my options.

What remained in the end were Devuan, antiX and Alpine. And I briefly tried all three during this experiment before settling on Alpine.

Devuan and antiX do offer options besides SysV-style init, but they don't really integrate them very well in their systems. While with Void you will really use runit to manage any services installed through the distribution's package manager, because when a package has a relevant service to ship it will ship a runit service, Devuan and antiX have other init systems as *options* rather than *the* distribution's init system.

What you really get then is a mixture of runit/OpenRC and SysV-style init, leaning a lot more towards SysV-style and leveraging the runit/OpenRC ability to also run SysV-style init scripts.[2] If you like managing SysV-style init, that should not be a problem, although in that case you might as well get the SysV-style version then! Alpine, on the other hand, just like Void, has OpenRC as *the* way to run services.

Alpine it is

For starters, I wanted to learn a little bit more about this unknown-to-me init system, OpenRC, since I'd be spending some time setting up services with it.

OpenRC can be used alongside or in place of SysV-style init. I will not get into the different roles played by so-called init systems and process supervisors, but OpenRC provides `openrc-init`, which can replace `/sbin/init` for a fully OpenRC-powered system.

I could not find explicit information on whether this is the case with Alpine, but an actual inspection of the running system shows it is not. Mind you, even Alpine's `/sbin/init` is actually BusyBox:

```
% file $(which init)
/sbin/init: symbolic link to /bin/busybox
```

I then checked out BusyBox's manual page to try and understand more:

```
  Init is the first process started during boot. It never exits.  It
  (re)spawns children according to /etc/inittab.  Signals:

  HUP: reload /etc/inittab TSTP: stop respawning until CONT QUIT: re-
  exec another init USR1/TER, since I'd be spending some time setting up servi
  halt/reboot/poweroff/Ctrl-Alt-Del script
```

I knew Alpine heavily leaned on BusyBox, but didn’t realize it was to this extent. So, following this trail, there was Alpine’s `/etc/inittab`, where I found this:

```
  ::sysinit:/sbin/openrc sysinit
  ::sysinit:/sbin/openrc boot
  ::wait:/sbin/openrc default
```

Meaning, Alpine’s init is BusyBox’s init, which in turn calls on OpenRC to bring services up.

Working around musl…

Willing to give OpenRC a try, I had to clear up my only concern with Alpine: lack of compatibility with glibc software.

This took a long time to evaluate, because you need to actually try to do what you usually do in real situations to see if things will work out or not.

If you are questioning whether Alpine *can* reliably run a full desktop environment or window manager, play and record audio, share your screen, run GUI applications including the latest browsers, act as the host of virtualization and development environments, the answer is an unwavering *yes*.

If you are questioning whether on top of that Alpine can play DRM video and games (e.g., Netflix and Steam), the answer is *also* yes, but you’ll need a glibc layer around it. If you like Flatpak, that might actually be trivial.

But for more involved usage of software that requires glibc, in particular novel and experimental stuff you are compiling yourself or exploring and prototyping with, your mileage may vary. It is more likely that a vendor will *not* provide a musl build of their software.

Some stuff works just by installing `gcompat`, a compatibility layer that will make glibc software run as-is. Some however won't work with it.

Sometimes I'd just compile something myself, meaning I made my own musl-linked executable of something that was available only for glibc. That works, but compilation for musl is often less documented and may not be possible at all depending on whether or not musl provides all the needed features. The latter case was however quite rare in my experience.

Some other options are running things through Flatpak, or Nix, or Distrobox. But sometimes, if all failed, I just had to accept it wasn't possible. Or it would work and then crash when using a specific feature at runtime.

And that's great, having ways to circumvent musl. But you could also just… run stuff directly? It's nice to have options, but it's also a lot of friction. And sadly that is what ended up exhausting me from using Alpine as a daily driver.

I think that, if you have a very consistent usage of Alpine, where you are mostly doing the same thing and using the same tools, you could find a comfy workflow there. But I am extremely moved by experimentation and curiosity. And something tells me that I could devise a more fluid workflow to run my experiments and try different tools inside isolated glibc-friendly environments, but in the end it's still friction. And it's hard to shake the feeling that, if you were running a more compatible system to begin with, that wouldn't even be something to think about.

Getting software from the distribution's package manager is the way to go for numerous reasons. But sometimes that is just not an option. Sometimes the package is just nonexistent in the repositories. Sometimes you need to compile it yourself because you need a specific version. Or because you don't trust the provided pre-compiled binary. Sometimes you need a specific compilation with a specific feature. Sometimes you are in a hurry, or just curious to try something you can securely run in isolation.

So not being able to install it the conventional way is friction in itself. Having to compile something every time is extra friction. Having to recompile on upgrades is more friction. And to not be able to compile or run it even by compiling it yourself, that is just compounded friction with some frustration sprinkled on top.

… though I’d rather not

This is not *about* Alpine, but it’s also inseparable from Alpine.

Alpine has an excellent, fast package manager that never broke my setup. When things did break, it was able to fix them by itself. The six months release cycle is to me a really sweet spot at which to space versions — not too often, not too seldom. A lot of software is available. Developers are security conscious. It’s extremely lean on resources.

And in the spirit of being so lightweight, Alpine shines on virtualized and embedded systems. This focus on being small is why musl is a great fit, since it’s a lot lighter than glibc, among other advantages. For me though, while I appreciate the smaller size, it’s not that critical that I’d sacrifice compatibility and versatility over it.

So I have zero complaints about Alpine, musl aside. It is perfect in everything I signed up for when I decided to try it. It delivered on every feature and never lied to me. It never said to me it was going to be smooth and easy to run glibc programs. So it’s not so much that I didn’t like Alpine, it’s that I wish there were a glibc Alpine just like there is a glibc Void. But I’m well aware that’s not happening.

Choosing a path beyond

It was excellent to learn hands-on about how Alpine works and what it can offer. I learned a lot about not just its components, such as OpenRC and BusyBox, which were basically strangers to me, known only conceptually. Alpine also taught me a lot about all the things it *doesn’t* have.

Because, seriously, if you do a chroot installation of Alpine and don’t even run its setup scripts, you’ll end up with something so trimmed down you’ll wonder if you’re midway through LFS. You’ll have to work your way up to a usable system piece by piece, finding out what you need, why you need it and how to set it up. And that is a huge learning opportunity for those interested

For now though, I decided to backtrack a little bit and then fully reverse course if I can't find a working compromise. That means giving Void another try and, if stability keeps being a problem, go further back and land on Debian. If I shed the non-systemd preference aside, Debian would deliver on the stability and compatibility that I am looking for.

You see, I am not looking for the *next* distribution to try as if it were a hobby and I wanted to try novelty out of FOMO. This is actually a lot of work! I'd rather do other things *with* the operating system rather than install and configure it, as I mentioned before. And if at the other end of the bell curve it's Debian, whatever. Perfect is the enemy of good and being picky is exhausting.

I mentioned curiosity is a big factor in wanting to try things too. And frankly, after a while holding this preference for non-systemd operating systems and using systems without it, I might as well seize the opportunity to learn more about systemd as well. At my current internship, systemd is what all the distros we have in production are using. In future work, that is very likely to be the case too. So there is nothing to lose.

I really wish, though, that package managers were more capable of differentiating between security and feature upgrades. If they were, we could run a rolling distro in "Debian mode" at will. This thought may not sit well with some rolling distributions, say, in Arch Linux you have the notion that "Partial upgrades are unsupported". There may be the expectation that the whole repository at any given time is the end state for all machines subscribed to it, but if dependencies could also be at the version and not only package level, that could also be solved, though I won't undersell the consequences and added complexity in such a change.

Just like mail clients, all operating systems suck. Some just suck less for some specific use case.

---

1. Also, if you find out how to reclaim that identity, do share. Asking for a friend. ↩

2. It's no surprise a distribution's main init system has better support, as it sure is a lot of work for packagers to support and test multiple init systems. ↩