Gentoo

Linux for experts

Onkobu Tanaake

2019-06-15T23:00:00

I've tried many Linux distributions, from early Mandrake to free SuSE and ran Gentoo for many years since its early days. Then I shifted my priorities from building packages to having a life. Recently I returned to Gentoo with a strategy and clear motivation: no clutter, no Canonical and no systemd. After half a year I'm up and running again. A big desktop machine works from time to time through the large tasks requiring processing power. For everyday work I use a laptop, Lenovo G505s and a mini-ITX platform hosts all the important services like a backup RAID, web server, file/ media server and databases (SQLite, PostGreSQL), Gentoo binary package host. The following paragraphs outline the foundation of my motivation, to invest a little more time again, now more efficiently.

First the hard facts:

- OpenRC instead of Systemd
- free from Pulseaudio, pure ALSA
- binary packages re-distributable, needs compatible architecture
- limited hardware support/ hand pick
- Better interoperability, clipboard (Ctrl-C/V), X-clipboard (selection only)
- Better control over features with USE-variable, libraries to be used, versions
- requires distcc/ detailed knowledge to use an old AMD Turion laptop
- Kernel compilation is expert level (NVidia, ICE1712)
- Carefully partitioning, even separate /usr, isolation and separate growth, tracking
- On-Off-relationship, mature tools, more reliable than in 2006
- free of clutter (3D, shine, search bar, Network Manager, zero redundancy, no background tasks like thumbnails)
- Scanner, Printer, drawing tablet, touch screen, video acceleration, ICE1712 audio, cryptography
- AppArmor in enforcing mode on all machines
- iptables using mainly -t mangle -A PREROUTING

- Secured Kernel and system according to checksec Guide [https://www.proteansec.com/forensics/gentoo-hardening-part-3-using-checksec-2/] and GitHub [https://github.com/slimm609/checksec.sh]
- Ecryptfs for disk encryption without auto-unlock through PAM
- SSH, GPG and OpenSSL for as much as possible
- No use of root user anywhere, Gentoo is quite complete with groups and dedicated users for services

I use hardware as long as I can. With Gentoo I can run a small XFCE-based system in 2019 on a laptop from 2005 (HP Pavillion ze6000). It doesn't compile larger packages very well like Libreoffice or a decent web browser. But I can make use of distributed compiling utilizing a larger machine. Delivering binary packages, even cross-compiled ones, is another alternative.

One of the downsides is the need for expert knowledge. This in turn requires decent amounts of time to gain this knowledge. Using a computer means more to me than just saving personal letters into my personal directory. I need folding marks, date of writing the letter as well as my adress/ header/ footer setup correctly. The easiest approach for this was LateX/ the dinbrief-template – nothing less.

Some might think »Why is this simple?« John Maeda already wrote »simple does not mean easy to understand«. I'd like to add easy to understand for everybody. Writing a few E-mail responses doesn't require in depth knowledge of an operating system or web technology. Anyone can click any attachment and follow the simple steps to earn a lot of money by simply sending personal bank account details. And after the macro was executed a reboot leaves the »that'll do«-user with an encrypted harddrive and the request to pay a decent amount of crypto currency to get a key to unlock holiday pictures and adult content.

I could also take a more beginner friendly Linux version like Ubuntu or even Debian, maybe a Red Hat product like CentOS. But I like it raw and without feature creeping assistants or super services. To run a desktop machine it is not necessary to pay attention to the dependencies of services beyond the run levels of Init-V-system. I'm also of the opinion, that it is neither a good idea to copy other OS' desktop clutter like ballooning messages, auto-mounts with auto-executes or desktop search. Have a set of sharp tools doing one thing and doing it good.

It is like taking a good photo. The process doesn't happen in editing. You can't put in what isn't there. Taking good raw material makes the later stages to the final product much easier. Thus I don't need setuid-bits in mount scripts but can make use of FUSE and/ or fstab-entries. So I can make use of half of the RAM as in-memory-disk much faster than any SSD and speeding up anything from database queries to build processes. And for years I could re-purpose the entire swap-partition as suspend-to-disk-memory. My systems never swap anything out, they're too slim.

So what helps me a lot to get along:

- Use piping, console, command line tools as much as possible

- Hand pick hardware, find out about the circuitry and linux support
- Take one challenge at a time, like get the system booting first, then install packages according to needs, optimize kernel, finally hardening
- Learn to embrace the limits, command line players, MuPDF or ImageMagick are great tools
- If there's a gap, fill with magic first and come up with serious engineering later
- When I feel comfortable I introduce a level of difficulty
- If it's not self explaining from the very first moment it has to pretty darn necessary or at least has to read from STDIN