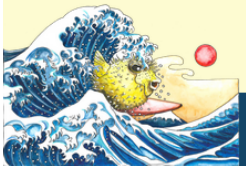
Why Rewrite Fw_update(8)?

Andrew Hewus Fresh andrew@afresh1.com

- afresh1@openbsd.org
- https://bsd.network/@AFresh1



Why rewrite fw_update(8)?

Andrew Hewus Fresh andrew@afresh1.com

- afresh1@openbsd.org
- https://bsd.network/@AFresh1

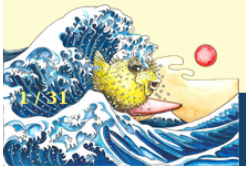
The OpenBSD mindset, and how fw_update came into being and evolved to what it is now.
Synopsis:
A trip into the history of L<fw_update(8)>, why it exists, and how OpenBSD prioritizes user experience.
Bio:
Andrew has been using OpenBSD for over 20 years and afresh1@openbsd.org for about ten now and in that time has contributed at least a week worth of effort to the project. He primary keeps L<perl(1)> up to date in the base system and maintains a few ports to have something to test those perl updates. He hasn't used OpenBSD professionally since before he got his account, but continually wishes other things were as nice to use. He has also restarted and has been organizing the BSD Pizza Night in Portland, OR since 2014, shortly after moving there.

Bad User Experience

We had some bad UX

After installing a new laptop, X doesn't start on first boot.

Yuck!



Bad User Experience

We had some bad UX

After installing a new laptop, X doesn't start on first boot.

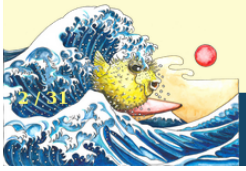
Yuck!

During installation, the installer asks if you want to start X on boot. After reboot, X doesn't start.
Why am I involved at all?
My main laptop is either my 2011 x220 or maybe a 2013 t440s. So, I didn't really hit it, but it is certainly a thing that new users would hit.
Theo was annoyed that after installing a new laptop he had to reboot before X would start.
But I thought, "how hard could it be?"
add slide -- Not because they are easy, but because we thought they would be easy

Why Was This The Experience?

Several reasons:

- Manufacturers are cheap.
- Manufacturers have different priorities for licensing than OpenBSD.
- OpenBSD cares a lot about licensing.



Why was this the experience?

Several reasons:

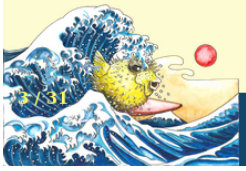
- Manufacturers are cheap.
- Manufacturers have different priorities for licensing than OpenBSD.
- OpenBSD cares a lot about licensing.

Manufacturers figured out they could save money by not including storage
They now make the OS load it onto the device
Does it really save enough money for it to be worth the hassle?
I suppose it's better than when the firmware couldn't be upgraded.
Now they can put distribution licenses on the firmware.
That didn't previously matter, the firmware shipped on a chip on the hardware and very few people thought about it.
Now that it's no longer part of the hardware, for a while people really cared.
Some folks only use 802.11b, Fossy had a special network.
OpenBSD cares a lot about licensing but hardware manufacturers don't.
So we have a conflict.

OpenBSD Is A Stickler For Licenses

```
$ head -4 src/gnu/README | tail -2
This directory contains software that is Gigantic and Nasty but
Unavoidable.
```

https://www.openbsd.org/policy.html



OpenBSD is a stickler for licenses

```
$ head -4 src/gnu/README | tail -2 This directory contains software that is Gigantic and Nasty but Unavoidable.
```

https://www.openbsd.org/policy.html

I got several email from folks patiently explaining that L<perl(1)> doesn't use (only) the GPL and does't belong in src/gnu.
I was glad when we redefined what the "GNU" subdirectory contains.
We have some things that have "unacceptable" licenses in base, but there are some licenses that haven't ever been (CDDL, GPLv3).
Even in src/gnu there are limits to what can be imported.
For example, no new GPL licensed code will be included. Updates to existing GPLv2 code notwithstanding.
Some examples of things in this directory, other than perl(1) are cvs(1), Clang/LLVM, and several GPLv2 versions of gcc for use on platforms that llvm doesn't support.

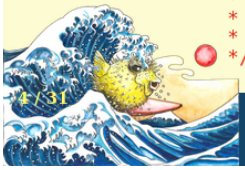
OpenBSD's Prefered License

```
$ cat /usr/share/misc/license.template
Below is an example license to be used for new code in OpenBSD,
modeled after the ISC license.

It is important to specify the year of the copyright.  Additional years
should be separated by a comma, e.g.
    Copyright (c) 2003, 2004

If you add extra text to the body of the license, be careful not to
add further restrictions.

/* Copyright (c) YYYY YOUR NAME HERE <user@your.dom.ain>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */
```



OpenBSD's prefered license

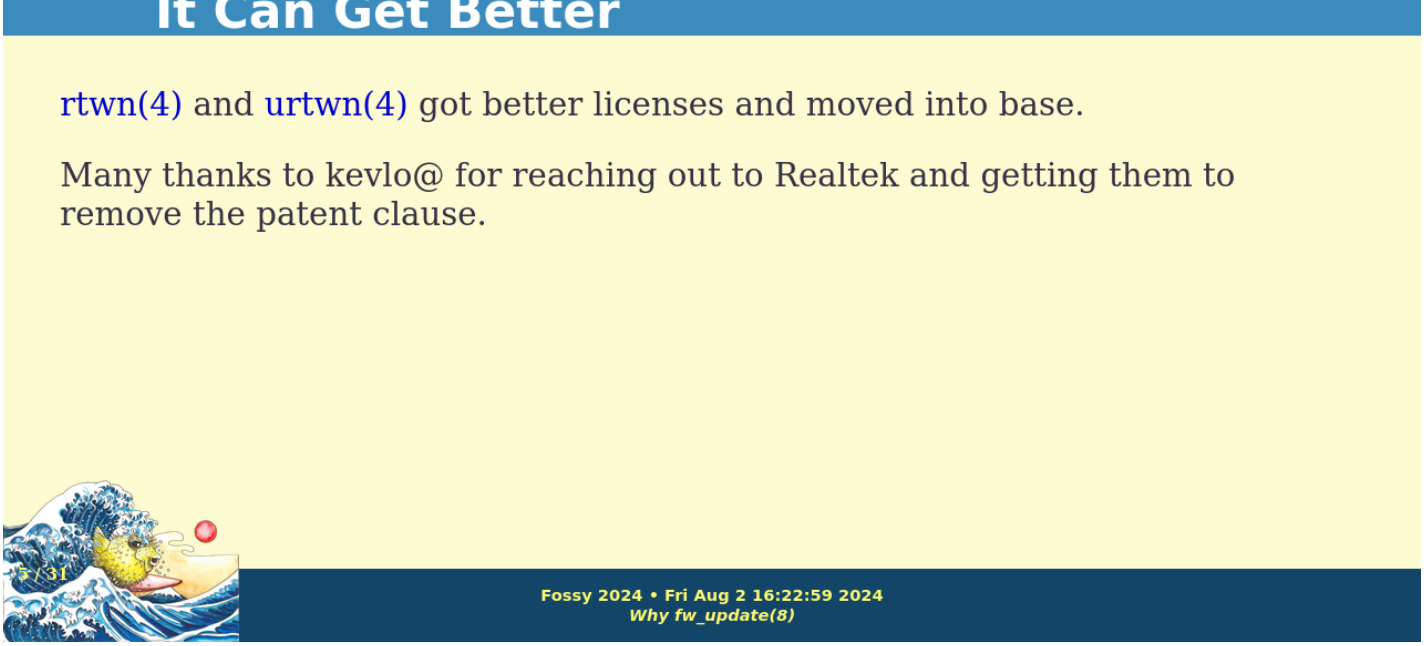

```
$ cat /usr/share/misc/license.template
Below is an example license to be used for new code in OpenBSD,
modeled after the ISC license.

It is important to specify the year of the copyright.  Additional years
should be separated by a comma, e.g.
    Copyright (c) 2003, 2004

If you add extra text to the body of the license, be careful not to
add further restrictions.

/* Copyright (c) YYYY YOUR NAME HERE <user@your.dom.ain>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */
```


OpenBSD has its own WiFi stack.
Although we do use the same DRM (Direct Rendering Manager) as Linux, as it has an acceptable license.
A lot of firmware has a license that can be included in OpenBSD's base system, even though it isn't "open source" and maybe the source isn't available.
Mostly OpenBSD cares that it can be distributed freely.
Some firmware doesn't have a license that allows distributing as freely as the base system needs.
Many of those licenses allow distributing them less freely, and that firmware is available separately.
Mostly wifi and graphics cards, but Intel chipsets too.



It can get better

rtwn(4) and urtwn(4) got better licenses and moved into base.

Many thanks to kevlo@ for reaching out to Realtek and getting them to remove the patent clause.

L<Old license|https://raw.githubusercontent.com/fuzeman/linux-firmware/fuzeman/master/LICENCE.rtlwifi_firmware.txt>
No more patent clause
L<New License|https://raw.githubusercontent.com/openbsd/src/master/sys/dev/microcode/rtwn/rtwn-license>
L<kevlo commit adding realtek firmware|https://github.com/openbsd/src/commit/2f22c03877562cecfe7a5d75957f3a51d58a0c46>
L<removing the firmware from fw_update|https://github.com/openbsd/src/commit/3a48df72c0eb61b630f78ada5f42d49c3fc03964>
Took a bit of work to figure out how to remove the old one without a "quirks" entry. The perl package tools use a special "quirks" package to handle these special cases, like things moving into the base system.

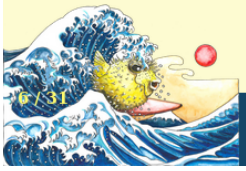
History

We still needed a way to have working hardware.

But how?

> We want to make available source code that anyone can use for ANY PURPOSE, with no restrictions.

-- https://www.openbsd.org/goals.html



History

We still needed a way to have working hardware.

But how?

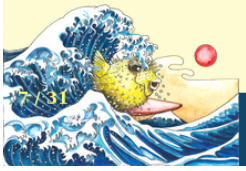
> We want to make available source code that anyone can use for ANY PURPOSE, with no restrictions.

-- https://www.openbsd.org/goals.html

We still wanted to be able to run OpenBSD on modern hardware.
People need to be allowed to use OpenBSD to build atomic bombs to be dropped on Austrailia or building baby mulching machines.
But even if they maybe get a more limited choice in wifi cards and don't have accelerated video, the rest of us don't need to suffer.

In The Beginning There Was Nothing

If the firmware license wasn't compatible, the hardware was unsupported.



In the beginning there was Nothing

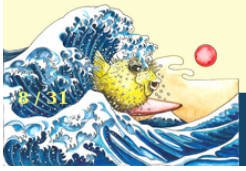
If the firmware license wasn't compatible, the hardware was unsupported.

If the license isn't acceptable, OpenBSD doesn't "use it anyway".
So the hardware just isn't supported and this leads to suffering.

Packaged Firmware

You had to know that your hardware needed firmware, and then install it.

This didn't last long.



Packaged Firmware

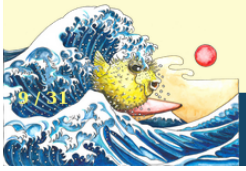
You had to know that your hardware needed firmware, and then install it.

This didn't last long.

https://github.com/openbsd/ports/commit/bbee873c5e9375a679caef0e7c2fcc92705b21a4
The first step was making the firmware more easily available, without putting bad licenses in base.
But you had to know you needed it, and figure out what you needed.

Wrapper That Scanned Dmesg

The first **fw_update**.



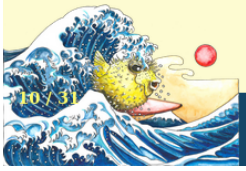
Wrapper that scanned dmesg

The first `fw_update`.

Shortly after, for the next UX improvement halex@ added a small wrapper around C<pkg_add> that did a simple update/install detection based on a hardcoded driver list. It scanned the dmesg to figure out which firmware you need and install it for you.
https://github.com/openbsd/src/commit/3e6d54bfa152b6fb8b908119b1ffa4982b1a01bf
Before my rewrite, it was incorrecly C<fw_update(1)> in section 1.
The initial fw_update was run during L<rc.firsttime(8)>, which meant firmware automatically installed on the first boot of the system.
This led to the issue with having to reboot before X would work.

Perl Wrapper

A better experience.



Fossy 2024 • Fri Aug 2 16:22:59 2024
Why fw_update(8)

Perl wrapper

A better experience.

The OpenBSD pkg tools are written in perl(1). This is because perl is good at dealing with the mess humans make, like versions and dependencies. It's entirely possible, but highly improbable that will ever change.
Then espie@ rewrote it in perl in 2015, embedded the logic into the the package system itself and replaced the shell script to provide a better user experience.
https://github.com/openbsd/src/commit/8205cf658d53e4ab271515bd1c17cba153c85bae
It still scanned a fairly small list of drivers, but the experience improved.
Although most folks never interacted with it directly, and this meant even less of that.
This was the same nice UX as the package tools now.

Initial Implementation

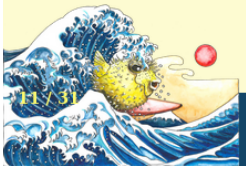
Started putting firmware into OpenBSD packages in 2011.

2011 - halex@ wrote a shell wrapper around **pkg_add** .

2015 - espie@ replaced it with a more tightly integrated perl version.

2022 - afresh1@ replaced it with a shell script.

2036 - ???



Initial implementation

Started putting firmware into OpenBSD packages in 2011.

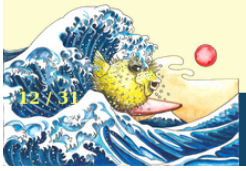
2011 - halex@ wrote a shell wrapper around `pkg_add`.

2015 - espie@ replaced it with a more tightly integrated perl version.

2022 - afresh1@ replaced it with a shell script.

2036 - ???

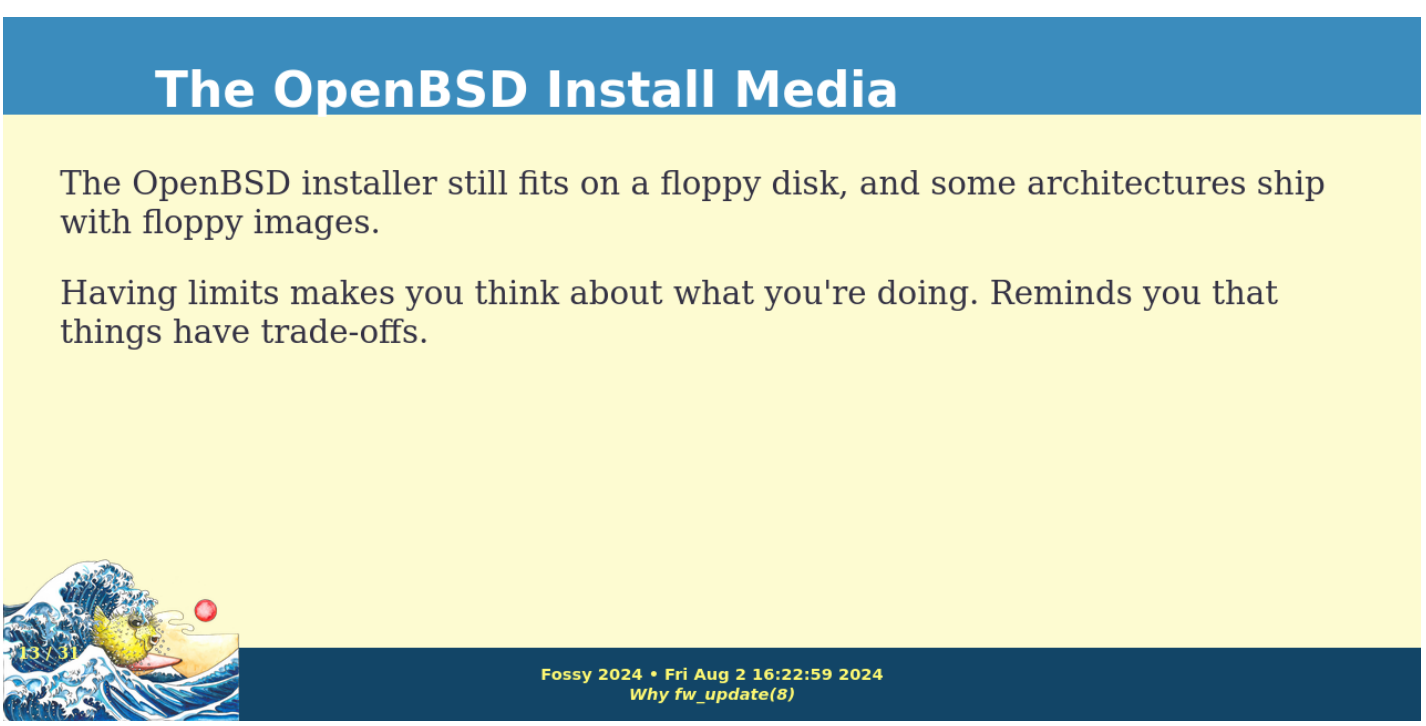
As you see, shortly before the end of the 32bit time_t, we're due for another rewrite.



Fixing the experience further

The fix for not having firmware on the first boot is to install the firmware when installing the system.

Things continually get nicer and more polished. So what are we polishing now?
Since we aren't allowed to include the firmware in the base system, and we need it available on the first boot, we need to install it between installing the system and rebooting.
Kernel?
Boot Loader (UEFI)?
Which means no place better than the installer.



The OpenBSD install media

The OpenBSD installer still fits on a floppy disk, and some architectures ship with floppy images.

Having limits makes you think about what you're doing. Reminds you that things have trade-offs.

While the installer is kinda a "live system", it's really just enough to do the install.
Since the installer includes everything necessary to install the system, that generally means that a motivated individual can recover a broken system using those same tools. There are some things that are mostly there to support recovery.
However, the install media doesn't fit L<perl(1)>.
We can't trust that the installer kernel will run the perl that was just installed.
So, we aren't able to use the perl fw_update, which means we had to reboot onto the running system before we could install the firmware.
Choosing not to add complexity where it isn't needed and keeping these limitations is one of the reasons OpenBSD has the best installer of any OS.
On the other side, if you're not careful, you end up with an installer that doesn't work over a serial console.
Or takes more than 10 minutes and a couple dozen questions to install a system.

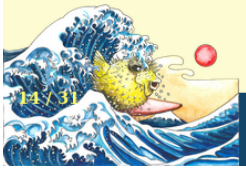
Knobs Are For Knobs

One of my favorite things about OpenBSD.

```
# OpenBSD
    ls [-1AaCcdFfgHhikLlmnopqRrSsTtux] [file ...]

# FreeBSD
    ls [-ABCFGHILPRSTUWZabcdfghiklmnopqrstuwxy1,] [--color=when] [-D format]
            [file ...]

# gnu ls doesn't even try to list the options in the synopsis
```



Fossy 2024 • Fri Aug 2 16:22:59 2024
Why fw_update(8)

---

knobs are for knobs

One of my favorite things about OpenBSD.

```
# OpenBSD ls [-1AaCcdFfgHhikLlmnopqRrSsTtux] [file ...] # FreeBSD ls [-ABCFGHILPRSTUWZabcdfghiklmnopqrstuwxy1,] [--color=when] [-D format] [file ...] # gnu ls doesn't even try to list the options in the synopsis
```

Side trip

One of my favorite things about OpenBSD, they think about whether a feature outweighs the complexity. Even better, they remove complexity where they can.

But lots of things either never become a sysctl, or over time that sysctl gets removed because the system auto-tunes or just one setting was "the best" for most users.

Making the interface simpler improves the user experience.

Bluetooth for example, or apache -> nginx -> httpd, or sudo -> doas.

(thanks brynet@!)

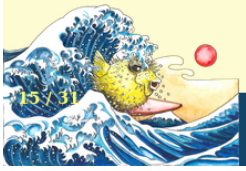
Another good example is the recent addition of adding full-disk-encryption support to the installer. While you have been able to use FDE for many years now, there were some manual steps necessary to enable it. Recently kn@ added first passphrase based FDE and then keydisk based. Keydisk based does still require you to prepare the keydisk manually. The amount of time spent discussing the questions was significantly more than the time discussing the actual code.

OpenBSD's Not GNU!

.Bl -enum produces ordinal numbers, not cardinal numbers;

patch from Jan Stary Ehans at stare dot czE.

We are both confident that the practical consequences of this
documentation bug are limited since you are not supposed to commit
manual pages containing infinite numbers of list items in the first
place (remember, OpenBSD's not GNU!) - but correctness is a virtue
in documentation nonetheless.

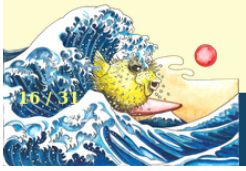


OpenBSD's not GNU!

.Bl -enum produces ordinal numbers, not cardinal numbers; patch from Jan Stary Ehans at stare dot czE. We are both confident that the practical consequences of this documentation bug are limited since you are not supposed to commit manual pages containing infinite numbers of list items in the first place (remember, OpenBSD's not GNU!) - but correctness is a virtue in documentation nonetheless.
A recent schwarze@ commit implies this attitude of simplicity.
You are not supposed to commit [...] infinte numbers of [...] items [...] but correctness is a virtue.
https://marc.info/?l=openbsd-cvs&m=171623026621515&w=2 https://github.com/openbsd/src/commit/5a50f35104472ad54fdd464b6f5a3220b218a011
OpenBSD is not here to be everything for everyone, instead it tries to stay as simple as possible while meeting the developers needs.

Packages Are Pretty Simple

Packages are nearly just tarballs.



Packages are pretty simple

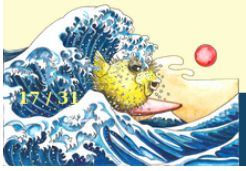
Packages are nearly just tarballs.

Back on track.
Fortunately, the packaging format for OpenBSD packages, which is used by the firmware packages, is fairly simple.
Nearly just tarballs.

This Is Harder Than It Looks

The installer only has limited tools available,

but we can also expect a "single user".



This is harder than it looks

The installer only has limited tools available,

but we can also expect a "single user".

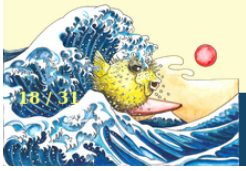
Unfortunately, not quite as simple as just downloading and extracting a tarball.
As you may have inferred, up until this point, the perl package tools were the only thing that actually dealt with /var/db/pkg.
But at least in the installer we don't need to deal with multiuser.
So some things are easier in the installer?

Limited Tooling

Some things we do have on the install media are OpenBSD's ksh(1), ftp(1) which happens to also be a simple http(s) client, and ed(1).

As well as a few other things.

The OpenBSD installer itself is written using these tools.



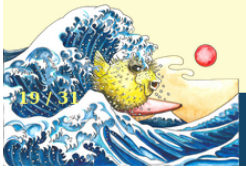
Limited tooling

Some things we do have on the install media are OpenBSD's ksh(1), ftp(1) which happens to also be a simple http(s) client, and ed(1).

As well as a few other things.

The OpenBSD installer itself is written using these tools.

Some things that haven't been removed, and are actually part of that install media gives us some impressive capabilities.
The OpenBSD installer is written with these tools that fit on a floppy disk.



Enough tooling

At least with a couple of limitations.

One: that none of the firmware would have any dependencies.

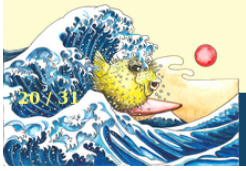
Two: there would only be a single version of any firmware available for a release at any time.

This gives us enough features that we are able to implement a very simple package system client. Fortunately, we were able to declare two things that made this possible.
No dependency handling, and no parsing version numbers.
In practice this means that only one firmware per driver in the SHA256.sig file.
If the version doesn't match, install the other one.
And we don't need 99%+ of the complexity of a package manager.
Protip, never write a package manager, the long tail of complexity is fractal.

So What Do We Have Now?

- `/usr/sbin/fw_update`
- `/usr/share/misc/firmware_patterns`
- `http://firmware.openbsd.org/firmware/${HTTP_FWDIR}/SHA256.sig`

And the firmware packages that live there.



So what do we have now?

- /usr/sbin/fw_update
- /usr/share/misc/firmware_patterns
- http://firmware.openbsd.org/firmware/${HTTP_FWDIR}/SHA256.sig

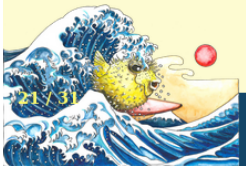
  And the firmware packages that live there.

Because the actual L<fw_update(8)> is a shell script, which is just a plain text file, we also don't have to include it on the install media, but instead can use the ksh from the install media to interpret the fw_update script that we installed. That means we don't have to worry as much about the size of the script. Which means we can use a bit of a nicer style than the very terse style used in the actual install scripts.

Pre-Compiled Firmware_patterns

The **firmware_patterns** are generated when building a release(8).

This allows generating patterns for all the graphics cards.

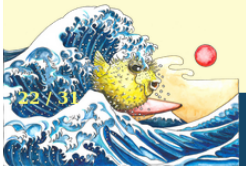


Pre-compiled firmware_patterns

The `firmware_patterns` are generated when building a release(8).

This allows generating patterns for all the graphics cards.

The final thing we did to make this work in the installer, where not all drivers are compiled into the kernel, to also include /usr/share/misc/firmware_patterns in the base sets. This maps a set of patterns to match from the L<dmesg(8)> that map to which driver, and therefore which firmware, we should install.
This is generated during build by a C program that knows how to query the drivers and see how they may appear in the dmesg.
From C<dev/pci/pcidevs_data.h> and the specific C<*_devlist.h> files from the different vendors.
Thanks to deraadt@ and rnagy@ for much of that.



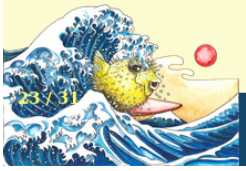
ksh(1)

An amazing amount of features, but not too many.

You can actually do bitwise calculations in it, I have written a script to do IPv6 address manipulation in pure ksh.
The firmware_patterns are actually fnmatch(3) globs so that ksh can match them directly without exec'ing an external program and match faster.
Still can't pass around arbitrary strings.
Another limitation, is no networking. (Should a shell really be able to do that?) (I'm sure networking in a shell is just a ghost story to scare me)

Ftp(1)

Actually an http(s) client.



ftp(1)

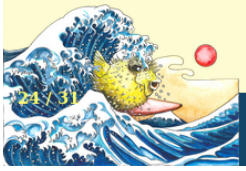
Actually an http(s) client.

Well, at least an C<HTTP GET> over TLS client.
Not the ability to POST over https. That went away when we lost lynx.
I have a helper to use L<nc(1)> with L<HTTP::Tiny(3p)>. OpenBSD's nc (really libreSSL's nc, which OpenBSD is upstream for) supports tls. Perhaps in perl 5.42 we will have a TLS library shipped with L<perl(1)>.
Although ftp(1) is using pledge(2) and unveil(2), it is still an older codebase that talks to the network. There is an unfinished http(1) client written by sunil@ that gets passed around on the lists from time to time, but the issues haven't been worked out so it hasn't been committed. So, we drop privileges when running ftp(1).
https://marc.info/?l=openbsd-tech&m=143315041829204&w=2

Su(1) Vs Doas(1)

In the installer, doas(1) is very limited.

Outside the installer, it has too many features, so we use su(1).



su(1) vs doas(1)

In the installer, doas(1) is very limited.

Outside the installer, it has too many features, so we use su(1).

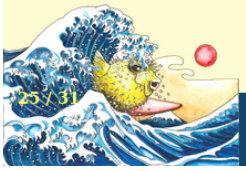
We actually use both to _drop_ privileges, since L<fw_update(8)> runs as root.
L<su(1)> isn't available on the install media.
On the install media, doas(1) is compiled down to a limited featureset that works more like su.
We can downloaded file as a lower privilege user before handling it as root.
Then we can verify is signed and trusted.

Signify(1) + Sha256(1)

Firmware packages are validated using signify(1) to sign a list of sha256(1) checksums in a **SHA256.sig** file.

```
 $ cat /etc/signify/openbsd-75-base.pub
untrusted comment: openbsd 7.5 base public key
RWRGj1pRpprAfgeF/rgld4ubduChLvTkigA1Zj7WLDsVA4qfYSWOEI8q

$ cat /etc/signify/openbsd-75-fw.pub
untrusted comment: OpenBSD 7.5 firmware public key
RWQ6EsXr4NMYvyLICug3dLHfmbpXlVasF1jbt3GVNQsosgB5+PgaufBu
```



signify(1) + sha256(1)

Firmware packages are validated using signify(1) to sign a list of sha256(1) checksums in a `SHA256.sig` file.

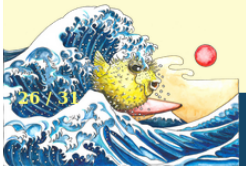
```
$ cat /etc/signify/openbsd-75-base.pub untrusted comment: openbsd 7.5 base public key RWRGj1pRpprAfgeF/
rgld4ubduChLvTkigA1Zj7WLDsVA4qfYSWOEI8q $ cat /etc/signify/openbsd-75-fw.pub untrusted comment: OpenBSD 7.5 firmware public key
RWQ6EsXr4NMYvyLICug3dLHfmbpXlVasF1jbt3GVNQsosgB5+PgaufBu
```

Normally we can use C<signify -C>, but that functionality isn't available in the installer.
So we use both tools to make sure the firmware we are going to install matches the checksum that was cryptographically signed before installing.
You can get the base public key in many places, and use it to verify the installer image.
Public key for base is in the installer, we install the firmware public key in the newly installed system.
As well as the keys for the _next_ release.
This is nearly the same validation that packages use.
Once validated, we can extract the firmware and register it with the pkg db.

Ed(1) -- The Standard Editor

"If you don’t know ed, you’re [...] Deficient."

-- https://mwl.io/tools/ed



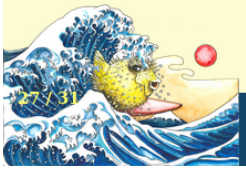
ed(1) -- The standard editor

"If you don’t know ed, you’re [...] Deficient."

-- https://mwl.io/tools/ed

One of the most useful things for system recovery in the installer is L<ed(1)>.
This is only of the things that used to be only for recovery.
But we script it to modify the C<+CONTENTS> of the package.

Well, only when not running in the installer.



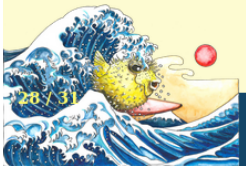
perl(1)

Well, only when not running in the installer.

When running multi-user, we need to be more careful.
We use perl to lock the package database when there could be multiple processes modifying it.

Fw_update(8) In The Installer!

We can now install firmware in the installer!



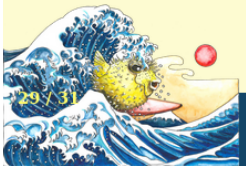
fw_update(8) in the installer!

We can now install firmware in the installer!

With all that, we are able to run L<fw_update(8)> during install.
We actually run it a lot, just in case:
During install or upgrade, if using L<sysupgrade(8)> we run it before rebooting, after install in L<rc.firsttime(8)>.

X Starts On First Boot

You no longer need to reboot to get into X.



X starts on first boot

You no longer need to reboot to get into X.

And with all that, nearly 850 lines of ksh, you no longer have to reboot after install before X will start.
Since OpenBSD 7.1.
The FAQ does have a section on adding extra firmware to the install image, if you need firmware for your wireless card to actually do the install.
https://www.openbsd.org/faq/faq4.html#WifiOnly

Thanks

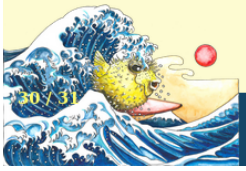
Andrew Hewus Fresh **andrew@afresh1.com**

- **afresh1@openbsd.org**
- https://bsd.network/@AFresh1

Artwork by Luc Houweling. I stole it from the 7.1 artwork.

Go buy a shirt: https://openbsdstore.com

And donate: https://openbsd.org



Thanks

Andrew Hewus Fresh andrew@afresh1.com

- afresh1@openbsd.org
- https://bsd.network/@AFresh1

Artwork by Luc Houweling. I stole it from the 7.1 artwork.

Go buy a shirt: https://openbsdstore.com

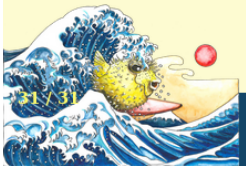
And donate: https://openbsd.org

BSD Stuff Around Portland

https://bsd.pizza

https://pdxlinux.org -- PLUG

https://calagator.org



BSD Stuff Around Portland

https://bsd.pizza

https://pdxlinux.org -- PLUG

https://calagator.org

If you are around Portland, OR, hit me up. We do a social meetup every month. We just get together to talk about BSD stuff over pizza and beer, or $food and $drink of your choice.
Michael Dexter organizes the PLUG (Portland Linux/Unix Group) every month for talks and more structured things.
The meetings for both get posted on calagator.org