Bitrig ports: BSD ports, packages, and Uncommon Operating Systems

AsiaBSDCon 2016

John C. Vernaleo, Ph.D.

jcv@bitrig.org

03/12/2016

A Complete System

BSD systems:

- Complete.
- Coherent.
- Each with a focus of its own.

Is that Enough?

Third Party Packages

- Most production systems require additional software.
- Desktop and laptop systems even more so.
- That is a big part of what made UNIX do so well.
- The reason GNU is UNIXy and not LISPy.

./configure; make; make install

- This almost works.
  - If you need a single package.
  - If you need something big (a web browser, half a dozen php versions, etc.).
- And you probably want security updates.
- Really, you probably want a number of programs which may depend on each other.

Ports and Packages

- 3rd party collection of packages.
- Provided by OS vendor.
- Every BSD has one.

Bitrig

A little on the OS I did this work on.

Bitrig pt. 2

Bitrig is a free, fast, and secure Unix-like Open Source operating system. It is available on current hardware platforms.

`https://bitrig.org/`

Bitrig pt. 3

- Forked from OpenBSD in 2012.
- First 1.0 Release in 2014.
  - But usable snapshots well before then.
  - Has been used in production and development environments.

Differences from OpenBSD

- git and github for version control.
- “Modern” development practices.
- clang as CC.
- Focus on amd64 and arm only.
- Other changes not too relevant for packages.

Bitrig Ports

- Based on OpenBSD ports
  - Makefiles
  - Perl for the tools
  - Patches
- Should seem very similar to OpenBSD.

But it never quite works out that way.

What does your software support?

- Everyone supports Linux.
- Maybe even OSX and FreeBSD.
- Isn't POSIX enough?

Less Common Systems

- The UNIX wars ended a long time ago.
- Do you need to support Solaris, IRIX, Unicos, etc.
- What about BSDs other than FreeBSD?
- Even worse for Plan 9, GNU/Hurd, anything really 'weird'.

Relative Number of Packages (early 2016)

OS	ports/packages
Bitrig	5,000
OpenBSD	9,451
NetBSD	14,132
MacPorts	16,500
FreeBSD	25,580
Debian	48,608

Why don't we all just use Debian?

- They have the most packages.
- Maybe the isn't the only factor to consider.
- And maybe you don't need all those packages.

autotools should help us

- Test for capabilities not names.
- But really just test for names.

Common Portability Issues

- OS name
- Compiler and Compiler Version
- Paths
- OS conventions
- Actual system changes and capabilities.

If you really want to fork...

Make the name *BSD.

Compilers

- clang/llvm vs gcc
- There is a lot of gcc only code out there.
  - Sometimes it is honestly gcc specific.
  - Usually it is non-standard C.
  - Can use the compiler the code wants.
  - Or can fix the code.
- FORTRAN hackers understood that there was more compiler years ago.
- C/C++ much less so.

The easy ones

- Interpreted languages tend not to be too bad.
- Probably why there are so many Perl, LaTeX, and font packages.
- The interpreter can be another story though (php, ruby, nodejs, and python, I'm looking at you).

Shared Packaging Systems

- Why don't we use pkgsrc or pkg_ng?
  - (or Gentoo ebuilds, or nix, or something else).
- pkgsrc already supports Bitrig and lots of other systems.

Shared Systems pt. 2

- Gets you far more committers.
- As is, packages require more people than kernel does.
- No need for messy merging.

Shared Systems pt. 3

- Doesn't solve the main problem.
- Less common systems will always get left behind.
- It does mean you get all sorts of nice features for free.

Other options

- Totally roll your own thing.
- Patchset on some other system
  - DragonflyBSD
- Use existing system.
  - To merge or not to merge.

And now you've hit a program that you need to fix

- Doesn't compile.
- Or worse, compiles and doesn't work.

Options

1. Maintain it yourself
2. Drop the package
3. Upstream it

Upstream Options

1. Accept it.
2. Reject it.
3. Ignore it.

Acceptance

- Best case.
- Nothing to maintain locally.
- This happens more often that you might expect.

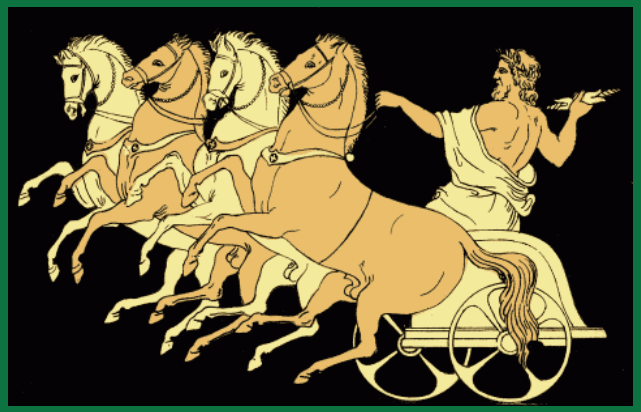
Most upstream authors happy for patches

- Extremely rare for portability to make software worse.
- Almost always leads to an actual increase in software quality.
- Anything that gets your software users is probably good.

But...

There is one class of software that can be an exception.

Scientific Software



- Modified and updated version of FORTRAN 77 NCSA release.
- ZEUS-MP v1.5.11
- `http://www.netpurgatory.com/zeusmp.html`
- A whole lot of #ifdefs

Rejection

- This seems to be the case people worry about.
- Doesn't happen that often.
- Might be something you can resolve.

Radio Silence

- Lots of abandoned projects out there.
- Sourceforge is full of them.
- Some are just unresponsive.

We still have options

1. Drop the program.
2. Maintain the patches.
3. Fork it.

Drop It

- Do you even really need the package?
- Is it worth having something that will never be updated again?
- But packagers like to have lots of packages.

Fork it

- The Github workflow kind of encourages this.
- You won't get updates easily.
- You might end up maintaining it forever.

Patches

- Just support the patches.
- This is the most common thing to do.
- But patches are very brittle.
- And someone not using your system doesn't get the benefits.

What Developers Can Do

Write portable code!

What you can do pt. 2

The world is not all Linux and OSX.

What you can do pt. 3

- Choose sensible defaults.
- Don't hard code compilers
- Even better, build with more than one compiler.
- Test on other systems.
- Maybe even automate that with something like Travis CI.
- Be receptive to downstream packagers.

Conclusions

- Third Party Packages are a big part of an OS (even a BSD).
- If you fix it, upstream it.
- Try to make it portable in the first place.