Porting a Linux package manager and Build System for PacBSD

KnoxBUG September 2017

https://pacbsd.org

Who is Adam Jimerson?

- System Architect and Lead Developer at Utiliflex (writing software to control power grids)
- Google Developer Group (GDG) Gigcity Lead Community Organizer
- PacBSD Contributor/Developer

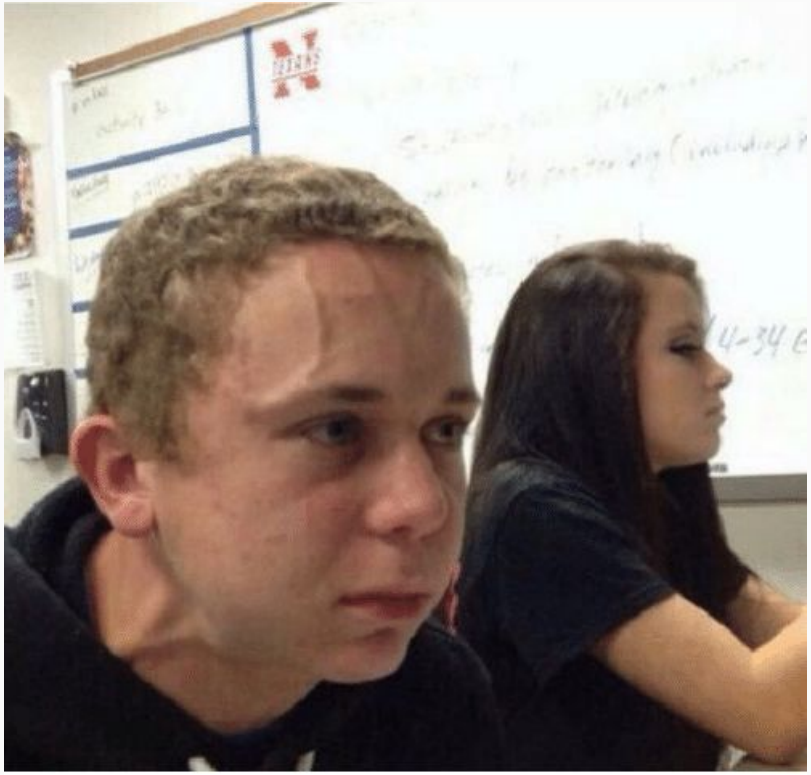
https://vendion.me vendion@gmail.com vendion@pacbsd.org

https://google.com/+AdamJimerson

https://pacbsd.org

Key takeaways

- A basic understanding of what PacBSD is
- Basic knowledge of the Arch Build System and FreeBSD ports tree
- Some knowledge of some of the differences and similarities between the BSDs and Linux
- A better appreciation for what package/port managers & maintainers do for you



When you use Arch Linux and you haven't told anyone in the last 10 minutes

https://pacbsd.org

Disclaimer: some or all of this talk may sound like Klingon or some other foreign language

I promise you it’s not

https://pacbsd.org

Qapla'
(Klingon word for “Success”)

https://pacbsd.org

What is PacBSD?

- Based on FreeBSD and Arch Linux
- Pacman as the package manager
- Choice of FreeBSD init or OpenRC init systems
- Choice of BSD or GNU core utilities*
- Hybrid of Arch Build System and Ports Tree to build packages
- Rolling Release cycle

* Currently the GNU core utilities are not built, due to missing tools but can be added in easily

Why FreeBSD?

- Subjectively better code quality and stability
- Native support for Jails and ZFS

Why Pacman and the ABS?

- Fast and simple package manager
- Built for rolling systems
- Shell scripts > Makefiles

https://pacbsd.org

A quick dive into the Arch Build System

- Made up of multiple repositories
  - Testing, Core, Extra, Community, Community-testing, and Multilib
- Each repository has a directory for each package in it
  - core/linux, extra/xorg, extra/wayland, extra/firefox, community/atom
- Each package must include a PKGBUILD shell script but may also include:
  - Any necessary patches, a “(pkgname).install” file, and/or pre/post transaction hooks for Pacman
- Supports split packages

A quick dive into the Arch Build system

- Provided tools:
  - Makepkg - Parses the PKGBUILD file and builds the package(s), also assists with keeping checksums current
  - Vercmp - Determines the relationship between two version numbers following Arch Linux's definition of version numbers E.g. 2:1.0-1 > 1:3.10-6
  - Pactree - Find and list all dependencies of a package, or all packages that depend on the given package (reverse dependencies)
  - Pacman - Provides a bunch of tools useful to users and developers alike

Example Arch Package

Click to Open

https://pacbsd.org

A quick dive into the Ports Tree

- Made up of software categories
  - Audio, Games, Security, Lang, Devel, German, Japanese, www, x11, etc
- Each category has a directory for each port in it
  - multimedia/vlc, www/firefox, editors/neovim, graphics/wayland
- Each port must include a Makefile, distinfo, pkg-descr, pkg-plist but may also include:
  - Any necessary patches under a sub-directory called “files”

A quick dive into the Ports Tree

- Makefile - Tells BSD's `configure` and `make` command how to make the port
  - This includes using GNU `configure` to generate a makefile for the port and/or swapping out for GNU `make`
- Distinfo - Contains checksums and file size of any files downloaded during build process
- Pkg-plist - Contains a list of files that will be installed by the port

Example Port Package

Click to Open

What's the point?

What PacBSD does to manage this

https://pacbsd.org

How we avoid being Frankenstein's monster

- Match package version with FreeBSD ports
  - Useful for things that tend to lag behind like display servers and graphics drivers due to lack of support from hardware manufacturers
    - Side note: when drm-next merge???
- Automatically apply any needed port patches while building
  - Not all port patches are relevant to us, like ones that change where things install to
- There are some options in the port's Makefile that we want to keep
  - Patch pacman adding PacBSD related options that can be used by `makepkg`

https://pacbsd.org

Automatically set options

- Fbsd10fix - Unescapes certain characters in files used by common tool chains (config.rpath/libpath, configure, libtool/libtool.m4, etc)
  - E.g. `s|freebsd\[\[123\]\]\*)|freebsd[[123]].*)|g`
- Set_compiler_clang - Build C/C++ files with Clang (LLVM) by default
- Apply_patches - Automatically applies any port patches found in the *files* directory if any
- Pathfix{,32} - Replaces FreeBSD path placeholders with a path

https://pacbsd.org

Optional options

- Libtoolfix - builds with the PacBSD libtool.mk file instead of upstream one
- Set_compiler_gcc - Build the package with GCC instead of Clang
- Dos2unix - Convert dos line endings to unix line endings
- Iconv{,32} - Converts codesets of files
- Gnu_configure - Use GNU configure when building

Example of a PacBSD PKGBUILD

Click to Open

https://pacbsd.org

A more interesting example

Bash PKGBUILD

https://pacbsd.org

Pac-build tool

- Wrapper around Arch's `makepkg` command
- Supports ZFS/UFS2 and Jails/chroot configurations
  - Handles creation and updating base jail or chroot environment for each architecture
- Allows each package to be built in a clean environment
  - ZFS:
    - Jails: Creates an additional ZFS dataset that sits on top the base jail dataset which contains changes during the build process
    - Chroot: Creates a ZFS snapshot before building are restores it after building
  - UFS: Recreates the build environment for each package for both Jails and chroot

https://pacbsd.org

Pac-build: Common tasks

- NullFS mounts host Pacman cache inside the Jail or chroot environment
- Creates a “builder” user and group in the base system to build as a non-privileged user
- Creates, updates, and removes build environments
- Makes it possible to get shell access to the base jail/chroot

Pac-build: Jails + ZFS

- Creates a base jail and ZFS dataset for each supported arch.
- Creates an additional dataset and snapshot for each package being built that builds on the base jail
  - If rebuilding packages couple options are available:
    - Start from scratch including installing dependencies
    - Don’t roll back package dataset to skip installing dependencies
- Con: requires manual clean up up datasets and snapshots
- Prefered configuration due to speed and flexibility

https://pacbsd.org

Pac-build: Chroots + ZFS

- Creates a chroot environment for each supported arch.
- Creates an additional dataset and snapshot (both datasets) for each package being built that builds on the base jail
  - If rebuilding packages couple options are available:
    - Restore base dataset from the snapshot previously taken to start from scratch
    - Don’t roll back base dataset to skip installing dependencies
- Con: requires manual clean up up datasets and snapshots
- Available fallback option to jails

https://pacbsd.org

Pac-build: Chroots/Jails + UFS2

- Creates a chroot environment/base jail for each supported arch.
- Recreates the base environment for each package being built
  - If rebuilding packages couple options are available:
    - Requires recreating the the base system to start from scratch
    - Skips the reinstall base step to keep current dependencies
- Much slower due to constantly having to recreate the base environment
- Available to users where ZFS is not an option or not familiar with ZFS

Thanks!

Adam Jimerson

PacBSD Developer/Contributor

System Architect - Utiliflex

vendion@pacbsd.org
www.pacbsd.org