Chimera Linux
A BSD/LLVM distro from scratch
Who?

- Developer at Igalia @ WebKit team
- FOSS contributor since 2007
- FreeBSD user since 2009
- POWER architecture maintainer at Void Linux
- Amateur gamedev
What is Chimera Linux?

- A new Linux distribution
- Not too interesting?
- Created from scratch
- FreeBSD-based userland
- Built fully with LLVM/Clang
What is Chimera Linux?

- Musl libc
- Rolling release
- Highly portable (ppc64le, aarch64, x86_64...)
- Bootstrappable
- Custom source package build system
What is Chimera Linux?

- Apk-tools for binary packages
- Lightweight service management (dinit)
- General purpose, graphical desktop
- Wayland, PipeWire, curated main/ repository
- Avoid legacy stuff when possible
Motivations

- Have some fun and start a community
- Explore alternatives, increase diversity
- Improve software portability
- Make a distro I will be happy with
- Learn from others’ mistakes, fix them
Motivations

- Prove Linux != GNU/Linux
- LLVM is a great toolchain
- Sanitizers, exploit mitigations (CFI)
- Actual usable LTO (ThinLTO)
- Superior cross-compiling support
Principles

- Not minimalist/"suckless"
- Not reactionary
- Not traditionalist
- If it sucks, get rid of it
- Systemd is not the root of all evil
Principles

- But technical debt is
- Be strict by default; enforce best practices
- There should be one obvious way to do it
- Be self-sustaining; good tooling is important
- Portability is not a joke
Principles

- Neither is bootstrapping
- It should probably not be written in shell
- Good things should be easy to do
- Bad things should be a pain in the ass
- Documentation is important
Principles

- Opinionated development is good
- Fun environment is an important thing
- There is no fun without good community
- At least not in the long term
- Meritocracy and technical-only spaces are crap
Early history

- Idea: ~2015 (Linux with BSD userland)
- Development start: May 2021
- Early June: first working prototype, <50 templates
- Reimagination of xbps-src from Void
- Using GCC, GNU userland, xbps
Early history

- Cports: “ports” tree with integrated build system
- Written in Python (package templates too)
- Designed to be fast (no buildsystem overhead)
- Namespaces for strict sandboxing
- Fully unprivileged
Example template

pkgname = "libpng"
pkgver = "1.6.37"
pkgrel = 0
build_style = "gnu_configure"
hostmakedepends = ["pkgconf"]
makedepends = ["zlib-devel"]
pkgdesc = "Library for manipulating PNG images"
maintainer = "q66 <q66@chimera-linux.org>"
license = "Libpng"
url = "http://www.libpng.org/pub/png/libpng.html"
source = f"${SOURCEFORGE_SITE}/libpng/libpng-{pkgver}.tar.xz"
sha256 = "505e70834d35383537b6491e7ae8641f1a4bed1876dbfe361201fc80868d88ca"

def post_install(self):
    self.install_license("LICENSE")

@subpackage("libpng-devel")
def _devel(self):
    return self.default_devel()

@subpackage("libpng-progs")
def _progs(self):
    return self.default_progs()
First steps

- Drop coreutils and related
- Project: bsdutils; bare and semi-functional
- Help port remaining tools for coreutils parity
- Additional new ports:
  - diff, grep, sed, ed, patch, m4, gzip
Porting BSD tools

- Code surprisingly clean and portable
- A few BSDisms shared between all code
- No REG_STARTEND in musl...
- tail(1): replace kqueue with inotify
- Otherwise pretty uneventful
Dropping xbps

- Original plan: use FreeBSD’s pkg
- Turned out not entirely ready for our use
- Alpine Linux’s apk proved a great fit
- Quick integration; robust by mid July
- Custom package generation code for now
Coreutils out, apk in

- Mid June: GNU userland mostly gone
- Still using GCC (and xbps for now)
- Late mid June: initial code for apk generation
- Early late June: xbps + coreutils out, apk fully in
- Why apk?
Benefits of apk-tools

- Lightweight but surprisingly elegant
- Transactional and constraint-based
- Robust dependency solver
- Support for triggers, virtual packages
- But about that GCC...
Dropping GCC

- Fairly easy: not much packaged yet
- Add standard LLVM build
- Tell it use its own runtime instead of GCC’s
- Recompile everything with it; fix errors (few)
- Finally remove GCC packaging (end of June)
Cross-compiling with GCC

- Target triplets and cross-toolchains
- Build: machine compiler runs on
- Host: machine compiled stuff runs on
- Target: machine compiled stuff works for
- Host == target most of the time
Cross-compiling with GCC

- Separate cross-toolchain for each host
- Binutils (linker, assembler, etc)
- Build “freestanding” compiler
- Build libc (musl) with it
- Build “final” compiler
Cross-compiling with GCC

- That means many compilers, one per target
- Tell builds what to use: e.g. `<triplet>`-gcc
- Kind of clunky and complicated
- Requires very specific infra
- Can we make it better?
Cross-compiling with LLVM

- One compiler for everything!
- One build of Clang targets every target
- Still need a “cross-toolchain”
- Compiler-rt builtins, musl, libunwind, libc++
- Slightly tricky to build
Cross-compiling with LLVM

- Compiler-rt needs libc headers to build
- Temporary header-only musl
- Build compiler-rt core bits: static only
- Add into sysroot; build musl with them
- Build libunwind (with --unwindlib=none)
Cross-compiling with LLVM

- Build libc++abi (with -nostdlib CXXFLAGS)
- Build libc++ (same)
- Build rest of compiler-rt (sanitizers)
- Tell Clang the -target + --sysroot
- Every arch built at once
Integrating into cbuild

- Separate host and target build dependencies
- Install host deps normally + cross-toolchain
- Install target dependencies with apk
- Treat cross sysroot as target apk’s root
- Dummy base package to provide virtual root
Integrating into cbuild

- In theory that is everything
- In practice various workarounds are needed
- Mostly handled transparently in cbuild
- Ensure correct flags are exported, etc.
- Proper handling of host/target build profiles
Improving cbuild

- Run unit tests for everything out of box
- Sandboxing: disallow network access
- Sandboxing: read-only container root
- Sanitize container environment
- Prevent breakout to outside system
First boot

```
... Output of boot process ...
```

Chimera Linux @ FOSDEM 2022
First boot

- Early October
- Linux kernel 5.14/15; some build workarounds
- Finally added init system and set it up
- Added initramfs generator – initramfs-tools
- Ported from Debian; surprisingly easy
Towards graphical system

- Increasing packaging pace
- Weston compositor by early November
- In cbuild: added update-check
- Checks upstreams for new versions
- Automatic; crucial for keeping things updated
Yes, it runs DOOM
DOOM

- Early December; milestone achieved
- Also turned on systemwide LTO
- Added GRUB bootloader
- Added PipeWire sound server
- User services
User services

- Automatic user instances of dinit via PAM
- Packages can install user services
- D-Bus session bus: not ad-hoc anymore
- Also PipeWire, WirePlumber
- Future expansion
Xorg
Xorg

- Added mid December
- More packaging: Gtk+3
- Desktops: Enlightenment, pekwm
- Basic input, video drivers
- Rust toolchain
Rust

- Needed for librsvg
- Major bootstrap pains
- Official x86_64-musl binaries sort of worked
- Symlink libgcc_s.so to libunwind.so
- Enough to recompile for our environment
Rust

- Major patching needed
- Added custom vendor triplets
- Generated custom bootstrap binaries
- Cross-compiled for ppc64le, aarch64
- Bootstrapped natively using those
Multimedia
Multimedia

- Made it just before year end
- Complete build of ffmpeg and dependencies
- Mpv media player
- Almost usable desktop system?
- Still need a web browser
Packaging a browser

- OpenSSL bump: 3.0.1
- Initial pieces of GNOME
- WebKitGTK and dependencies
- Epiphany
- Total template count: 500+
Packaging a browser
Future

- Migrate to apk-tools 3.x
- Release official binary repositories (and set up CI)
- Package GNOME desktop
- Extend user services framework further
- Migrate my own systems to it
Future

- Improve system documentation
- Enable Clang CFI, UBSan on eligible targets
- Investigate a _FORTIFY_SOURCE_ implementation
- Integrate non-stub locale support
- Integrate non-stub utmp/wtmp?
Thanks for listening!

- https://chimera-linux.org
- https://github.com/chimera-linux
- #chimera-linux @ OFTC (irc.oftc.net)
- #chimera-linux:matrix.org