Chimera Linux

A BSD/LLVM distro from scratch

Who?

- FOSS contributor since 2007
- On a break from work
- FreeBSD user since 2009
- Former Void Linux developer
- Jack of all trades

Chimera Linux

- New Linux distribution (mid 2021)
- General purpose, from scratch
- FreeBSD core tools, LLVM toolchain, musl libc
- Rolling release
- Highly portable to many architectures

Quick history

- Started in early-mid 2021
- Cbuild: meta-build system for packages
- Void Linux developer at the time
- Quick distro written around cbuild
- GCC, GNU userland, xbps package manager

Quick history

- Some 50 package templates
- Not bootable
- Capable of building itself
- Aiming to fix shortcomings of xbps-src
- Performance and correctness

Quick history

```
[    0.402241] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[    0.417580]  sda: sda1
[    0.427952] sd 0:0:0:0: [sda] Attached SCSI disk
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Running /scripts/local-premount ... done.
Warning: fsck not present, so skipping root file system
[    0.587452] EXT4-fs (sda1): INFO: recovery required on readonly filesystem
[    0.587515] EXT4-fs (sda1): write access will be enabled during recovery
[    0.607723] usb 1-1: new high-speed USB device number 2 using xhci_hcd
[    0.625602] EXT4-fs (sda1): recovery complete
[    0.641871] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
[  OK  ] early-filesystems
[  OK  ] early-udevd
[  OK  ] early-console
[  OK  ] early-hwclock
[  OK  ] early-udev-trigger
Checking root file system (^C to skip)...
fsck from util-linux 2.37.2
fsck: error 2 (No such file or directory) while executing fsck.ext4 for /dev/sda1
[  OK  ] early-root-fsck
[  OK  ] early-root-rw
[  OK  ] early-static-devnodes
[  OK  ] early-udev-settle
fsck from util-linux 2.37.2
[  OK  ] early-aux-fsck
[  OK  ] early-aux-filesystems
[  OK  ] early-rcboot

Chimera 5.14.14-foo-2 (chimera) (hvc0)

chimera login: root
Password:
# uname -a
Linux chimera 5.14.14-foo-2 #1 SMP ppc64le
# grep -i gcc /boot/config-$(uname -r)
CONFIG_GCC_VERSION=0
# grep -i clang /boot/config-$(uname -r)
CONFIG_CC_VERSION_TEXT="clang version 12.0.0"
CONFIG_CC_IS_CLANG=y
CONFIG_CLANG_VERSION=120000
#
```

Quick history

- Switch to LLVM, FreeBSD tools, etc.
- Gradual packaging expansion
- Iterative enhancements
- Bootable, self-bootstrapping system in late 2021
- GNOME desktop in early 2022

Quick history

Quick history

- Talk at FOSDEM 2022
- Many refinements during 2022
- Security hardening
- Development of fresh solutions
- Aiming for alpha in February/March 2023

Why?

- Unhappiness with existing systems
- Make a well-rounded, practical OS
- Improve software in general
- Make full use of LLVM (sanitizers, LTO...)
- Prove Linux != GNU/Linux ;)

Why?

- Have some fun
- Internet people said I couldn’t
- Build a good community
- Make a good system for myself
- Make a good system for other people

General principles

- Projects with 1 goal are doomed to fail
- Pointless dogmas are doomed to fail
- However, beware of scope creep
- Opinionated development is good
- Software quality matters, community more so

General principles

- Fun is good, try to keep it that way (for everyone)
- FOSS projects are social spaces
- Reactionaries and toxic people make a toxic community
- Try hard to keep them out
- Attempt not to stray towards pointless elitism

Technical principles

- Be strict by default and avoid technical debt
- There should be one obvious way to do it
- Be simple, but not too simple
- Hardening is important
- Strive for portability

Technical principles

- Good tooling is important
- Being self-sustaining is important
- Infrastructure should be easy to replicate
- Be able to bootstrap from source
- (keep in mind it’s a means to an end though)

Technical principles

- If it's written in shell, it likely should not be
- Make it easy to do right, and hard to do wrong
- Documentation matters
- Systemd's not good, but it brought necessary change
- Develop good solutions and keep head out of the sand

BSD system development

- Entire system in a single tree
- Lots of components sharing a repository
- Complete OS capable of boot
- Third party software distributed through ports
- Some third party components in base (toolchain...)

What makes a Linux distro?

- A collection of software from different parties
- Linux kernel as the base layer (always)
- Userland tooling (often GNU), libc (often GNU)
- Toolchain to build all this (often GNU)
- Service manager and aux tooling (often systemd)

What makes a Linux distro?

- Tied together with a package manager
- Linux + GCC + glibc + coreutils + … => GNU/Linux
- Distros ensure these components all work together
- Test many different configurations
- “We don’t break userspace” – Linux kernel

Sorting out the toolchain

- LLVM/Clang is pretty seamless nowadays
- Most Linux systems => GCC provides runtime
- LLVM comes with its own (compiler-rt)
- Used in Chimera (no libgcc)
- Musl used for libc (from the start)

Sorting out the toolchain

- GNU binutils complements GCC (linker etc.)
- In Chimera, LLD from LLVM is used everywhere
- Elftoolchain provides other tooling (+ libelf)
- This is also used by FreeBSD
- LLVM also provides the tools (e.g. llvm-readelf)

Core userland

- GNU coreutils, findutils, diffutils, grep, sed, util-linux...
- Busybox often used on non-GNU distros (e.g. Alpine)
- Overly spartan functionally, code is not very good
- Other alternatives are usually worse
- FreeBSD to the rescue

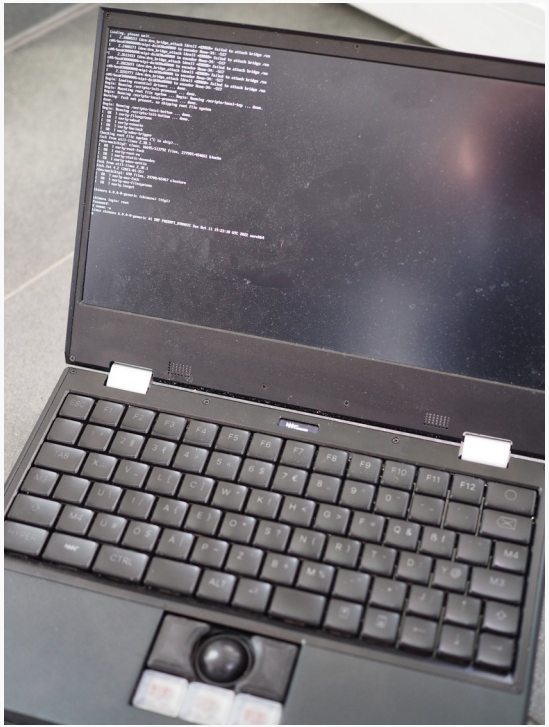
Core userland

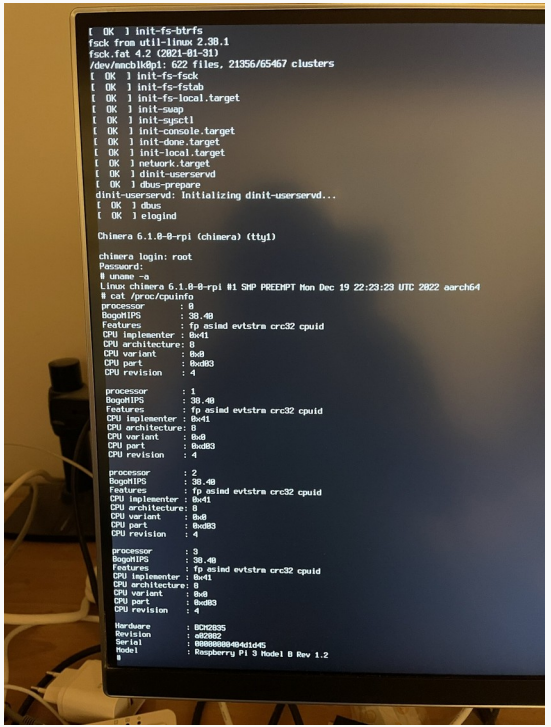
- Found third party port: bsdutils
- Incomplete, bare, not quite ready
- Helped complete it, reach parity with coreutils
- Ported many other tools to expand coverage
- Result: chimerautils

Core userland

- Replaces not just GNU tooling
- All-in-one userland package (simplified bootstrap)
- Lean enough for bare envs (e.g. initramfs)
- Powerful enough for interactive use
- Easy hardening, help break up monoculture

Linux kernel





Linux kernel

- Clang-compatible these days
- Patches to support BSD utilities, Elftoolchain libelf
- LLVM_IAS did not work on some archs until recently
- E.g. ppc64le needed binutils because of that
- Clang assembler used everywhere now

CKMS

- Distros usually use DKMS to build out of tree modules
- Massive bash script, unfitting functionality
- Created CKMS (Chimera Kernel Module System)
- DKMS-like functionality, but lightweight and robust
- Privilege separation, packaging integration...

Package manager

- FreeBSD pkg considered
- Not quite in the shape I’d like
- Alpine Linux’s apk proved excellent
- Eventual transition to apk-tools 3.x (new generation)
- Full cbuild integration in summer 2021

Service management

- Many options evaluated (runit, s6, openrc, ...)
- Dinit was chosen
- First distribution to use it as a default
- Lean but powerful: supervising, dependency-based
- A good base for a solid service infrastructure

Service management

- Custom suite of core Dinit services
- Full support for fine-grained service targets
- First-class support for user services
- Goal: all long-running processes should be services
- Integration with session tracking

Turnstile

- Linux mostly uses systemd-logind for session tracking
- Elogind exists as a “standalone” version
- Ripped out of systemd and lots of things stubbed out
- (E)logind tracks when users log in and out
- Done with a daemon and a PAM module

Turnstile

- This+seat management is used by desktops
- With systemd, logind also spawns user systemd
- This manages services running in user session
- Elogind cannot do this (functionality stubbed out)
- New solution developed

Turnstile

- Turnstile aims to eventually fully replace elogind
- Manage user instances of Dinit
- Don’t manage seats (work in tandem with libseat)
- Provide a library alongside daemon
- Agnostic API (for either logind or turnstiled)

Turnstile

- Manage Dbus session bus as a user service
- Result: a single session bus per user (like systemd)
- Fixed session bus address (/run/user/$UID/bus)
- Limitless potential for user services (Dbus activation?)
- Currently also used for e.g. PipeWire/WirePlumber

Cbuild

- Build system for cports (our “ports tree”)
- Written in Python (standard library only)
- Builds package templates (which are Python scripts)
- Generates apk repositories
- High performance, but also strict and secure

Cbuild

```
pkgname = "crispy-doom"
pkgver = "5.11.1"
pkgrel = 0
build_style = "gnu_configure"
hostmakedepends = ["automake", "pkgconf"]
makedepends = [
    "sdl-devel", "sdl_mixer-devel", "sdl_net-devel",
    "libsamplerate-devel", "libpng-devel"
]
pkgdesc = "Limit-removing enhanced-resolution Doom source port"
maintainer = "q66 <q66@chimera-linux.org>"
license = "GPL-2.0-or-later"
url = "https://github.com/fabiangreffrath/crispy-doom"
source = f"{url}/archive/{pkgname}-{pkgver}.tar.gz"
sha256 = "7c5bb36393dec39b9732e53963dadd6bcc3bd193370c4ec5b1c0121df3b38faa"
# FIXME int
hardening = ["!int"]

def pre_configure(self):
    self.do("autoreconf", "-if")
```

Cbuild

- Everything builds in a simple container (build root)
- Minimized Chimera system (with build deps installed)
- Unprivileged and fully sandboxed (Linux namespaces)
- Build container is read-only, no network access
- No access to the outside system by the build

Cbuild

- Templates are declarative (ideally just metadata)
- Transparent cross-compiling support
- Clean handling of common build systems
- Strict, mandatory linting hooks, unit tests run OOTB
- Bulk build support, update-check, ...

Build flags and hardening

- All the basic stuff (FORTIFY, PIE, stack canaries, etc.)
- System-wide LTO (ThinLTO)
- System-wide UBSan subset by default
- Protected against signed integer overflows
- CFI (Control Flow Integrity) for many packages

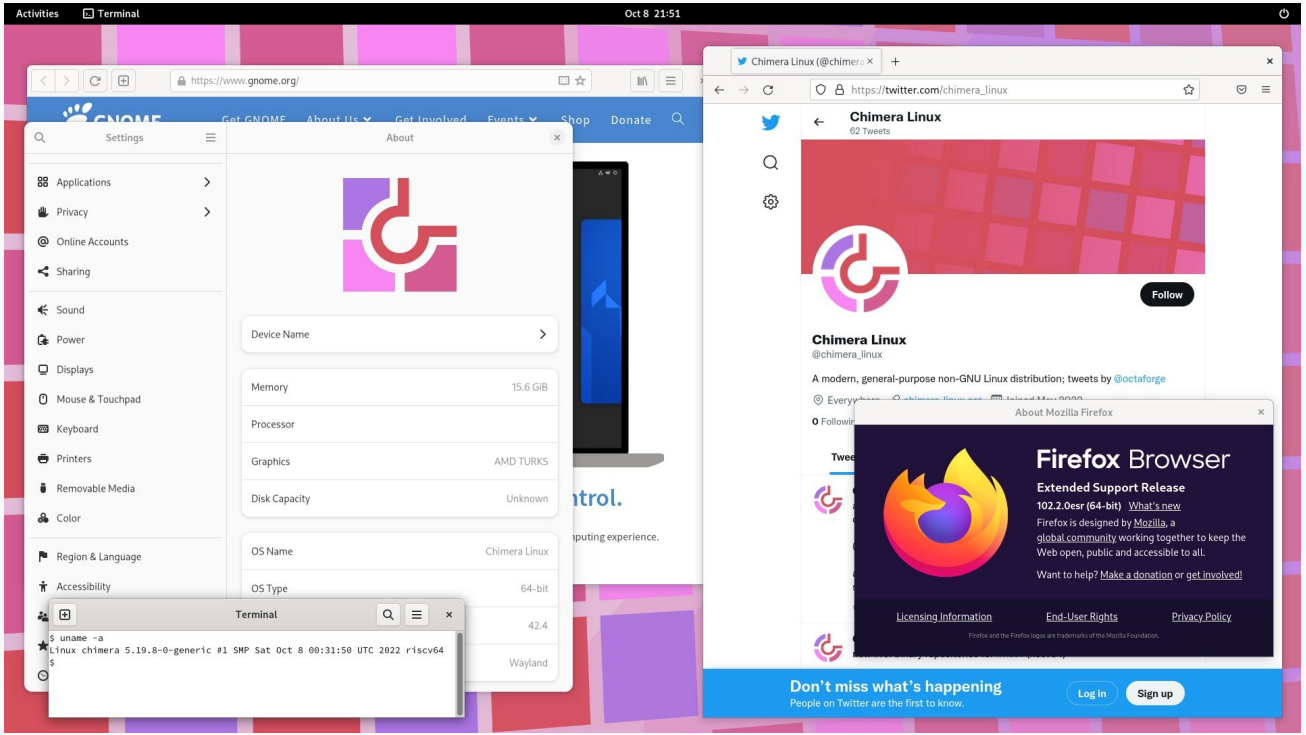
Scudo allocator

- We now use LLVM's Scudo hardened allocator
- Also used by e.g. Android by default
- Replaces musl mallocng
- Significantly better multithread performance
- GWP-ASan support eventually

Other core things

- Some tooling taken from Debian
- Initramfs-tools for initramfs generation
- Console-setup for console configuration
- Cryptsetup scripts for encrypted drives
- GRUB bootloader, ZFS support (incl. root), etc.

Chimera desktop



Chimera desktop

- First added: Weston and Gtk
- Expansion: Xorg, Enlightenment/EFL, pekwm
- Multimedia stack (FFmpeg, GStreamer)
- GNOME desktop (default) added in spring 2022
- Web browsers (Epiphany, Firefox), games (DOOM)

Towards alpha

- To be released in late February to early March
- World rebuild
- Automatic build infrastructure
- Clean up hardening fallout
- Update every template

After alpha

- Expected to take 6-12 months
- Add libgcc compatibility shim (binary compatibility)
- Add support for D-Bus activation
- Investigate additional hardening
- Documentation, locale support, etc.

Conclusion

- Currently nearing usability
- Suitable for early adopters by March
- Get all major changes done by beta
- Continue packaging software
- Cooperate with upstreams

Thanks for listening!

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