Pratham K

prathamIN@proton.me | blog | @git-bruh

Technical Skills

Build Systems: Cargo, CMake, Meson, Autotools, Scons, Make, Waf
Developer Tools: Git, Docker, Podman, Github Actions, Shellcheck
Debugging Tools: GDB, Strace, Uftrace
Languages: Bash / POSIX sh, C, C++, Python, Rust, TypeScript
Language Tools: Clang Format, Clang Tidy, Sanitizers (ASAN, UBSAN, TSAN)
Linux Kernel APIs: Landlock, Seccomp, User Namespaces
Miscellaneous: CI/CD, Cross Compiling, Linux Kernel Configuration, Systems Administration

Open Source Experience

Package Maintainer and Core Team Member 2022 – Present

KISS Linux Community

- Participated in the packaging and regular maintenance of large software packages such as Compiler Toolchains, Browsers like **Firefox** and **Chromium**, tools like **Docker** etc. involving working with various build systems, communicating with package maintainers from other distributions, and patching software to be more portable, allowing usage on esoteric systems using **musl libc**, pure **Clang/LLVM** toolchains, etc.
- Contributed various bug fixes to the shell-based **KISS** package manager, involving the **ldd**-based dynamic dependency detector and the package alternatives system: kiss-community/kiss
- Lead the development of various infrastructure automation related projects to reduce the scope of human error, save time, and facilitate reproducibility: kiss-community/maintainer-utils
  - Wrote a multi-stage bootstrap script to build the rootfs tarball from scratch in an unprivileged fashion using **unshare** and **bubblewrap**
  - Wrote various helper scripts for maintainers to check package versions against the latest upstream versions on **repology**, helping semi-automate the package maintenance process

Open Source Contributor 2022 – Present

Various Contributions To Projects Used In Personal FOSS Endeavours

- Fixed an undefined behaviour bug in **Chromium** that caused certain pages to crash on **GCC**-built binaries: chromium/#4546610
- Made various contributions to a terminal UI rendering library, **termbox2**, including portability-related bug fixes, and features like ANSI escape sequence parsing for mouse events: termbox/termbox2
- Wrote a detailed report including findings from commit bisection regarding a bug in the **virglrenderer** backend used by **Qemu** that caused Wayland applications inside the guest machine to crash, helping uncover a bug in **Mesa**: virglrenderer/#291
- Contributed a Python script to the DevelopersIndia community for creating weekly job posts on the subreddit, using **PRAW** and **FeedParser** for consuming an RSS feed, and a simple JSON store for persistence: deviras/#9
- Uncovered a small bug related to improper **libCURL** usage in **Flatpak** and **OSTree** that led to crashes under certain circumstances: flatpak/#5074, ostree/#2706

Contractual Projects

libCURL Cross Build | *C, Rust, Docker, CMake, Cargo, Android NDK* Feb 2023

- Developed an end-to-end build pipeline for cross compiling the **CURL** library with it's rust bindings to various architectures targeting the Android platform
- Used **cross-rs** to setup the **Android NDK** toolchain images serving as a base for the **Dockerfile**
- Wrote a **Dockerfile** to build the **CURL** library, along with transitive dependencies such as **Brotli**, **Nghttp2** and **BoringSSL**, involving debugging build failures, backporting upstream patches, and figuring out appropriate build options for various target architectures

Kaldi ASR Client | *C++, Python, CMake, GRPC, NVIDIA Triton* Dec 2022

- Developed a client library in C++ for performing audio inferences on WAV files with the **NVIDIA Triton Inference Server**, using **GRPC** for communication
- Wrote Python bindings for the same using the **ctypes** module and handled potential issues such as exception handling and propagation of signals such as **SIGINT** back to Python on the C++ side

- Developed an end-to-end build pipeline for installing build dependencies on the host, building unpackaged C++ libraries such as **Kaldi** from source using **CMake** and integrating the build artifacts along with Python bindings into a Python wheel
- Resolved various issues arising during the build process due to build system quirks such as non-relative RPATHs being set in shared libraries, preventing their relocation to other paths, resolved with tools like **patchelf**
- Wrote a simple daemon script using **Netcat** to provision Triton server instances on-demand in an effort to work around a memory leak bug which was extensively documented and reported upstream: kaldi/#4814
- Documented the entire end-to-end deployment process in detail including building libraries, configuring the Triton server, and using the Python bindings

Personal Projects

S6 Scripts | *POSIX sh, execline, s6-rc, mdev* Feb 2023 - Present

- Ported KISS Linux's init scripts from **BusyBox runit** to the **s6-rc** service manager for better reliability, logging and dependency management
- Wrote system initialization scripts performing various tasks such as mounting pseudo-filesystems, loading kernel modules, coldplugging devices, along with service definitions for daemons like **dhcpcd**, having a clean and explicit dependency graph allowing maximum parallelization during the boot process

Landbox | *C, Make, Linux Syscalls* Oct 2022 - Present

- Explored the Linux Landlock API as a contender to User Namespaces for the purpose of filesystem sandboxing by developing a small CLI program inspired by **bubblewrap**, having the ability to restrict read, write and execute permissions for specified paths
- Wrapped the kernel-level Landlock API in a reusable helper library, featuring runtime checks to detect available Landlock features, wrappers for making syscalls, along with helper constants correlating traditional file permissions (read, write, execute) to Landlock rules

Matrix TUI | *C, Meson, cJSON, libCURL, lmdb, termbox2* Aug 2021 - Present

- Developed a minimal TUI for the Open Source Matrix communications protocol, interacting with the REST APIs laid down in the specification
- Wrote various TUI widgets from scratch, including scrollable and auto resizing input fields, treeviews, message views etc., putting general-purpose & reusable parts of the code into a sub-library, termbox-widgets
- Wrote another sub-library, libmatrix, wrapping the Matrix REST APIs with **libCURL** and **cJSON**, providing a clean library interface exposing structs and enums corresponding to event types, iterators over the received events, and helpers for event creation and HTTP requests
- Implemented an asynchronous UI with constraints such as handling input while simultaneously receiving events from producer threads by effectively leveraging queues, threads, pipes, signals, and the poll() syscall
- Implemented an efficient key-value event store with the **LMDB** database and designed clean abstractions over it to facilitate serialization of events received from the server
- Prevented memory safety bugs by writing extensive unit tests with the **Unity** test framework and integrating them with **ASAN** (Address Sanitizer) and **TSAN** (Thread Sanitizer)
- Used **Meson** as the build system, leveraging features like subprojects, and integrated **clang-format** for code formatting and **clang-tidy** for static analysis

Matrix Discord Bridge | *Python, Bottle, Sqlite3, Urllib3, Websockets* Nov 2020 - Present

- Developed a simple, self-hosting friendly bridge to share communications across Discord channels and Matrix rooms with a 1:1 mapping of features like replies, emotes, and mentions
- Used the Python **Websockets** library to interact with the Discord API and handle various quirks such as heartbeats and disconnects within the constraints of the event loop
- Implemented a webhook endpoint with the **Bottle** framework to receive events from the Matrix home server, along with **Sqlite3** for persisting relevant data

Education

Manipal University Jaipur Part Time, Remote

Bachelor of Computer Applications *2023 – Present*