Distributing software on Linux

Valentin David

Bergen Open Source Conference 2024
November 7th

Table of Contents

1. Introduction
2. APIs and shared libraries
3. Packages are not for third party
4. Bringing your own userspace ABI
5. Conclusion

About me

- Work for Canonical
  - Ubuntu Core
  - Snapd
  - Snaps' core runtime
- GNOME foundation member
  - GNOME OS
  - Freedestkop SDK, the main Flatpak runtime
- My opinion. I do not represent Canonical or GNOME here.

Issue statement

- A developer wants to provide an (hopefully open source) application to Linux users.
- Nowadays, not all Linux users can build from source code.
- Linux as in Linux+GNU+FreeDesktop
- Focus on the C world. Everything else depends on it.

Distribute software, the old way

1. Release a source archive

2. Hope every distribution will make packages of it

Table of Contents

1. Introduction
2. APIs and shared libraries
3. Packages are not for third party
4. Bringing your own userspace ABI
5. Conclusion

Linux API

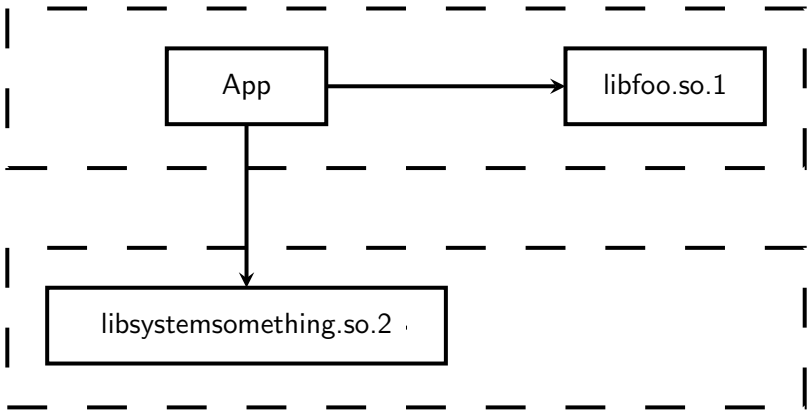
- system calls (Linux kernel)
- devices through ioctl (Linux kernel)
- file systems (procfs, sysfs, cgroups) (Linux kernel)
- sockets (AF_NETLINK, Linux kernel)
- C API
  - virtual system calls (Linux kernel)
  - system shared libraries
- IPC: dbus, varlink, wayland, etc.

About shared libraries

- PT_INTERP (e.g. /lib64/ld-linux-x86-64.so.2) (only on executable)
- Multiple DT_NEEDED
- Dynamic symbols table
- Relocation tables (to relocate the GOT)
- DT_SONAME, ABI "major" version (only on libraries)
- Symbol version tables (with definitions and requirements)

Shared library ABI versioning

System libraries, application libraries?



System libraries, application libraries?

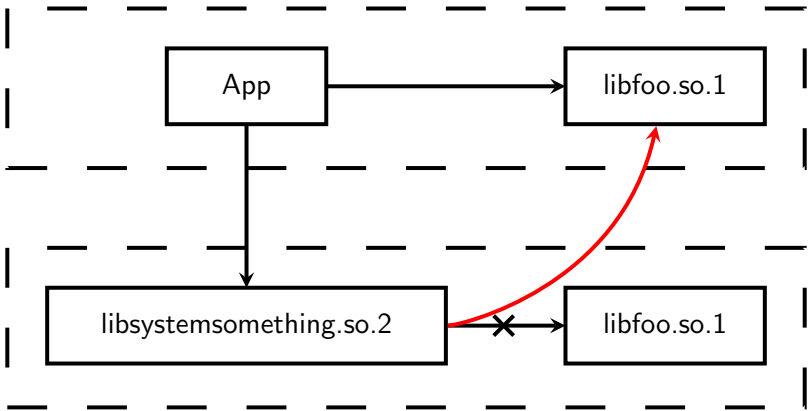


Table of Contents

1. Introduction
2. APIs and shared libraries
3. Packages are not for third party
4. Bringing your own userspace ABI
5. Conclusion

Packages

The good

- Dependencies install shared libraries
- Dependencies know about ABI versions

The bad (for third party)

- A package built on Ubuntu 24.04 might not work on 22.04
- A stale dependency can break upgrades
- A bad dependency can break your OS
- Not user friendly to install third party packages

Special mentions

- Ubuntu's Personal Package Archives (PPA)
- openSUSE's Open Build Service
- Nix

Table of Contents

1. Introduction
2. APIs and shared libraries
3. Packages are not for third party
4. Bringing your own userspace ABI
5. Conclusion

Doing it yourself

- Interpreter (ld−linux.so) is absolute path
- Glibc need to match interpreter (new libc with old ld-linux.so does not work)
- shared libraries backends for Vulkan, OpenGL, EGL, libdrm need updates every time a new GPU is released
- openssl, gnutls, nss need updated CA certificates
- security fixes
- Glibc's Name Service Switch is plugin based. Needed for DNS, user/group query, etc.

Open Container Initiative and Docker

- Good for microservices
- Docker has a popular registry
- Not practical for desktop applications
- One blob, so need updates
- Not usually installed by default

Systemd portable services

- No default “store”
- Good for services only
- Can split base from application

Flatpak

- Desktop applications
- Installed by default on Fedora, Endless, Mint, Elementary, GNOME OS...
- Default store `flathub.org`
- `https://docs.flatpak.org/en/latest/`

Snap

- Installed by default on Ubuntu
- Can do system and user services
- Can do desktop applications
- `https://snapcraft.io/docs/snapcraft-tutorials`

Table of Contents

1. Introduction
2. APIs and shared libraries
3. Packages are not for third party
4. Bringing your own userspace ABI
5. Conclusion

What to use

Maximize your audience: try to use most of them. It is less that there are distros.

Services OCI and Snap

Desktop Flatpak and Snap