GPL-3-free replacements of coreutils

Contents

Due to the nature of Apertis and its target markets there are licensing terms that are problematic[1] and that forces the project to look for alternatives packages. The `coreutils` package is good example of this situation as its license changed to GPLv3 and as result Apertis cannot provide it in the `target` repositories and images. The current solution of shipping an old version which precedes the license change is not tenable in the long term, as there are no upgrades with bugfixes or new features for such important package.

This situation leads to the search for a drop-in replacement of `coreutils`, which need to provide compatibility with the standard GNU `coreutils` packages. The reason behind is that many other packages rely on the tools it provides, and failing to do that would lead to hard to debug failures and many custom patches spread all over the archive. In this regard the strict requirement is to support the features needed to boot a target image with ideally no changes in other components. The features currently available in our `coreutils-gplv2` fork are a good approximation.

Besides these specific requirements, the are general ones common to any Open Source Project, such as maturity and reliability. Particularly important aspects are also the available community support, the development process and user adoption.

As a summary, bellow is the list of attributes

- License suitable for inclusion in Apertis
- Compatible with GNU `coreutils`
- Support for the features needed to boot a target image

[1] https://em.pages.apertis.org/apertis-website/policies/license-expectations/

- User adoption
- Community support
- Long term solution

Coreutils GPLv2

Currently Apertis provides `coreutils-gplv2`, with the following features

```
[ base64 basename cat chgrp chmod chown chroot cksum comm cp csplit cut date dd
df dir dircolors dirname du echo env expand expr factor false fmt fold groups
head hostid id install join link ln logname ls md5sum md5sum.textutils mkdir
mkfifo mknod mktemp mv nice nl nohup od paste pathchk pinky pr printenv printf
ptx pwd readlink rm rmdir seq sha1sum sha224sum sha256sum sha384sum sha512sum
shred shuf sleep sort split stat stty sum sync tac tail tee test touch tr true
tsort tty uname unexpand uniq unlink users vdir wc who whoami yes
```

Alternatives

In order to perform a comparison among different projects this section list different projects and metrics of each them. These metrics are quantitative ones, which can obtain from the Git log, and qualitative that can be derive from the first ones. The value of showing all these metrics is to allow non-technical users to clearly understand the comparison.

uutils-coreutils

Link: https://github.com/uutils/coreutils
Language: Rust
License: MIT
GNU compatibility: High (it is the project goal)
User adoption: Low
Completeness: Missing 14 commands
Started: 2013
Developers in last year: 40
Commits in last year: 885
Project status: Very active
Community support: High
Maturity: Medium

Pros

- High GNU compatibility
- High community support
- High community impact
- Portability in mind
- Ongoing development

- Implemented in a modern memory safe language
- Interest from Debian developers

Cons

- Missing commands and features
- Not used in production environments
- Depends on many Rust crates, which may not all be already available in Debian

Notes

- Semi-done: `cp expr install ls more od printf sort split tail test date join df`
- To do: `chcon csplit dd numfmt pr stty`
- Missing compared to coreutils-gplv2: `csplit dd dir pr stty vdir`
- Builds successfully on Apertis using the available Rust compiler
- Initial tests for basic features were successful
- A Debian Developer already ran some tests booting a Debian graphical session with GNOME using uutils-coreutils[2]

BSDutils

Link: https://github.com/dcantrell/bsdutils
Language: C
License: BSD
GNU compatibility: Low (project is only a port of OpenBSD compatible with Linux)
User adoption: Very low
Completeness: Missing 25 commands, long options unsupported, other differences Started: 2019
Developers in last year: 1
Commits in last year: 86
Project status: Active
Community support: Low (base project high)
Maturity: Medium (base project high)

Pros

- Linux support
- Based on OpenBSD, which is a mature project

Cons

- Missing commands and features
- Not fully compatible with GNU as it is a port from OpenBSD
- Low community support for the port itself
- Not used in production environments

[2] https://sylvestre.ledru.info/blog/2021/03/09/debian-running-on-rust-coreutils

- Original project only supports OpenBSD, Linux support added in a low activity fork
- Requires libbsd-dev

Notes

- This project is a port of tools from OpenBSD to have an BSD-licensed and lightweight replacement of GNU coreutils
- Provides a set of scripts to import new OpenBSD versions and a set of patches to be applied and provide Linux compatibility
- In order to upstream contributions might need to be done to this specific project or to OpenBSD
- Missing from coreutils-gplv2: `base64 cksum dir dircolors hostid link md5sum md5sum.textutils od pathchk pinky ptx seq sha1sum sha224sum sha256sum sha384sum sha512sum shred shuf sum tac tail unlink vdir`

Busybox

Link: https://busybox.net/
Language: C
License: GPLv2
GNU compatibility: High (compatibility in mind but a subset of features)
User adoption: Very high
Completeness: Commands with limited features
Started: 1999
Developers in last year: 27
Commits in last year: 299
Project status: Very active
Community support: High
Maturity: High

Pros

- High GNU compatibility
- High community support
- Very low footprint
- Already part of Apertis

Cons

- Supports a subset of features

Nbase

Link: https://github.com/cheusov/nbase
Language: C
License: BSD
GNU compatibility: Low (project is only a port of NetBSD compatible with Linux)

User adoption: Very low
Completeness: Missing 33 commands
Started: 2015
Developers in last year: 1
Commits in last year: 119
Project status: Active
Community support: Low
Maturity: Medium

Pros

- Linux support
- Based on NetBSD, which is a mature project

Cons

- Missing commands and features
- Not fully compatible with GNU as it is a port from NetBSD
- Low community support
- Not used in production environments
- Requires NetBSD make, mk-configure, libbsd
- Original project only supports NetBSD, Linux support added in a low activity fork

Notes

- This project is a port of tools from NetBSD compatible with other Unix like systems
- Missing from coreutils-gplv2: `[ base64 chgrp chown chroot dir dircolors factor groups hostid install link md5sum md5sum.textutils od pathchk pinky ptx readlink sha1sum sha224sum sha256sum sha384sum sha512sum shred shuf sum tac unlink users vdir who whoami`

FreeBSD

Link: https://github.com/freebsd/freebsd/tree/master/bin
Link: https://github.com/freebsd/freebsd/tree/master/usr.bin
Language: C
License: FreeBSD
GNU compatibility: Very low
User adoption: High
Developers in last year: 72 (on usr.bin)
Commits in last year: 423 (on usr.bin)
Project status: Active
Community support: High
Maturity: High

Pros

- High community support

Cons

- Missing commands and features
- No Linux support
- No GNU compatibility

Sbase and Ubase

Link: https://gitlab.com/garbeam/src/-/tree/master/bin/sbase
Link: https://gitlab.com/garbeam/src/-/tree/master/bin/ubase
Language: C
Project status: Inactive, no activity since 2016
Community support: None

Pros

- Linux support

Cons

- Project inactive

Heirloom

Link: https://en.wikipedia.org/wiki/Heirloom_Project
Link: https://wiki.archlinux.org/index.php/Heirloom
Language: C
Project status: No activity since 2007
Community support: None

Pros

- Linux support

Cons

- Project inactive

Replacement: uutils-coreutils

Based on the above comparison the best option is `uutils-coreutils`, since it is the only one with the explicit goal of providing a fully compatible alternative to GNU `coreutils`, and it has a good community support which most probably will continue and improve in the future. The main risk is the current low user adoption and the lack of usage in production scenarios. It is worth to mention that the main license used in the project is MIT but further analysis needs to be done to confirm the licensing of all the used dependencies.

These risks enumerated will be handled by the testing and migration in order to provide a reliable approach.

As it has been mentioned the license used is MIT, and detailed information about its dependencies can be found in the FOSSA analysis[3]. Unfortunately, this report is not reliable since it shows several incorrect dependencies.

The following list shows the dependencies as reported by `cargo`

Package	License
ansi_term	MIT
arrayvec	MIT OR Apache-2.0
autocfg	MIT OR Apache-2.0
backtrace-sys	MIT OR Apache-2.0
bitflags	MIT OR Apache-2.0
bit-set	MIT OR Apache-2.0
bit-vec	MIT OR Apache-2.0
blake2-rfc	MIT OR Apache-2.0
byteorder	Unlicense OR MIT
cfg-if	MIT OR Apache-2.0
chrono	MIT OR Apache-2.0
constant_time_eq	CC0-1.0
data-encoding	MIT
dunce	CC0-1.0
either	MIT OR Apache-2.0
failure	MIT OR Apache-2.0
fake-simd	MIT OR Apache-2.0
fnv	MIT OR Apache-2.0
fs_extra	MIT
glob	MIT OR Apache-2.0
half	MIT OR Apache-2.0
hex	MIT OR Apache-2.0
ioctl-sys	MIT OR Apache-2.0
isatty	MIT OR Apache-2.0
maybe-uninit	MIT OR Apache-2.0
md5	MIT OR Apache-2.0
num-integer	MIT OR Apache-2.0
onig	MIT
onig_sys	MIT
pkg-config	MIT OR Apache-2.0
platform-info	MIT
ppv-lite86	MIT OR Apache-2.0
rand_chacha	MIT OR Apache-2.0
rand_pcg	MIT OR Apache-2.0
rust-ini	MIT
semver	MIT OR Apache-2.0

[3] https://app.fossa.io/projects/git%2Bgithub.com%2Fuutils%2Fcoreutils?ref=badge_large%22

Package	License
semver-parser	MIT OR Apache-2.0
sha1	BSD-3-Clause
sha2	MIT OR Apache-2.0
sha3	MIT OR Apache-2.0
smallvec	MIT OR Apache-2.0
strsim	MIT
syn	MIT OR Apache-2.0
synom	MIT OR Apache-2.0
synstructure	MIT
tempfile	MIT OR Apache-2.0
term_grid	MIT
termsize	MIT
term_size	MIT OR Apache-2.0
thread_local	MIT OR Apache-2.0
typenum	MIT OR Apache-2.0
unix_socket	MIT OR Apache-2.0
vec_map	MIT OR Apache-2.0
wild	MIT OR Apache-2.0
winapi-util	Unlicense OR MIT
xattr	MIT OR Apache-2.0

Testing

In order to confirm the missing features/commands in the `uutils-coreutils` which are required by Apertis a testing needs to be performed. The steps proposed are:

- Run initial tests on target images
  - Test booting standard target images
  - Test installing/removing packages
- Run current `coreutils-gplv2` test plan with `uutils-coreutils`
- Run `uutils-coreutils` as default on development environments
- Make `uutils-coreutils` and all the Rust crates it depends on available in Debian
- Provide long-term maintenance of the new packages in Debian as well
  Note that some effort is being driven by `uutils-coreutils` community to use the `coreutils` test case to generate a report for the still missing features. This will be a nice to have feature but it is more than it is actually required for this stage.

Initial test and results

As part of an initial test using `uutils-coreutils` the following steps have been taken

- Replace utilities from `coreutils-gplv2` with the ones provided by `uutils-coreutils`
- Boot target image without issues
- Reinstall package `libc6` without issues

These initial results are promising, however more detailed tests should be planned and executed to spot potential issues.

Migration

As of `v2022dev3`, `uutils-coreutils` is the default `coreutils` implementation shipped on the Apertis reference images for devices, while GNU coreutils remains in use in the package building pipelines and on the SDK images. This work involved importing Debian's `rust-coreutils`[4] package along with its dependencies.

The upstream Debian package isn't aimed at replacing `coreutils` yet, meaning some packaging changes were necessary to get Apertis images to build using `rust-coreutils` only:

- Change the package's priority from `optional` to `required`: this ensures `debootstrap` will pick up this package when bootstrapping the system
- Add Conflicts/Breaks/Provides/Replaces relationships so all dependencies are satisfied and conflicting packages (such as `coreutils-gplv2`) cannot be installed
- Install the binaries to `/bin` and `/usr/bin` instead of `/usr/libexec/rust-coreutils`

Additionally, a few patches were necessary to implement missing command-line options and most of them are in the process of being upstreamed. Some of our patches still require more work, especially regarding SELinux-related options: for now, we made sure the corresponding command-line options would be recognized, but the associated behavior isn't implemented yet. Since SELinux is not used in Apertis this was enough to ensure images could be built successfully, but these patches are not suitable for upstreaming. Implementing full SELinux support is not currently in the scope of Apertis, maintaining the downstream patches until upstream implements proper support is not going to be particularly problematic.

`coreutils-gplv2` have been consequently removed from Apertis `v2022dev3` and later repository.

[4] https://tracker.debian.org/pkg/rust-coreutils