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://sjoerd.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

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

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
- 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 bmake mk-configure libbsd-dev
- 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.

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 `uuutils-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 pacakgage `libc6` without issues

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

Migration

Since `coreutils-gplv2` is a base package special care should be taken. Also the fact that it is outdated adds additions possible security issues, which should be addressed in the short term.

The following guidelines will be followed to assure a smooth transition minimizing risks.

- Determine the list of tools supported and successfully tested provided by `uutils-coreutils`.
- Create a new package based on `uutils-coreutils` named `coreutils-uutils` with all the tools that are supported and successfully tested.
- For missing tools a replacement will be provided on case by case basis.

Due to the Apertis release flow[2] this process will start on development releases allowing any potential issue to be addressed before a stable point release, with the possibility of switching back to `coreutils-gplv2` if a proper fix cannot be implemented on time.

[2] https://sjoerd.pages.apertis.org/apertis-website/policies/release-flow/