Google Summer of Code '25 Proposal

Debian - Autopkgtests for the rsync package

Alex - alex@puer-robustus.eu

Abstract

Extend the Debian rsync package autopkgtest suite by running more upstream tests, setting up end-to-end tests with a proper ssh connection, and adding new tests for so far uncovered cases.

Version 0.2

Application tasks

The application tasks for this project include, amongst others, providing a log of the `sbuild` run with `autopkgtests` enabled. You can find mine here.

This is the result of running `sbuild -n > /tmp/rsync_sbuild.log` (`sbuild` version `0.88.4~bpo12+2`) with this config based on `origin/debian/master` in my local clone of the `rsync` Debian source package repository.

As my `sbuild` log shows, `lintian` complains about an outdated FSF address in the `debian/copyright` file (I assume this is due to the `fail-on warning` settings in my `sbuild` config for `lintian`). My MR on salsa.debian.org replaces the physical FSF address with a link to the relevant web page. While this silences `lintian`, I am not fully confident that it is correct for me to adjust license file information like this. But given that the FSF does not have a physical office anymore, it might make sense to merge this commit? Regardless, this change should have zero influence on the tests run by `autopkgtest`.

After adding the `rsync -h autopkgtest` test config as shown in this draft MR, running `sbuild` with the above configuration gives the following log (lintian still fails because I have commited the copyright file adaptation onto a separate branch).

Goals

The general goal of this project is to add `autopkgtest` test configurations to the `rsync` Debian package so as to increase its coverage. Since the upstream tests are also run

as part of the existing `rsync autopkgtest` configuration, the question of where those additional tests should live (upstream vs. Debian) arises.

I believe all tests concerning the behavior of the application *as such* should ideally[1] be parts of the upstream source. The earlier tests catch regressions, the lower is the cost/effort/time investment necessary to fix their underlying cause and prevent negative repercussions on users.

However, even in an ideal setting, there are tests which distribution package maintainers should care about more than upstream developers: those tests which check the integration of distribution package X with all of its dependencies as provided by the distribution.

In addition to giving maintainers of package X peace of mind, setting up tests which check the functioning of the package X, also reassure maintainers of all of X's dependencies, as regressions in new versions of those packages can be caught this way.

Against that backdrop and in the context of the Google Summer of Code (GSoC) '25 program at Debian and the project "Autopkgtests for the rsync package", I therefore propose to

1. Configure the Debian `rsync autopkgtest` run such that most (if not all) upstream tests are run. Currently, some of the upstream tests are skipped; mostly for lack of root rights (compare this Debian CI test run, lines 846-900):

```
59s ./runtests.sh running in
    /tmp/autopkgtest-lxc.vcttef7a/downtmp/build.Pzh/src
59s    rsync_bin=/usr/bin/rsync
59s    srcdir=/tmp/autopkgtest-lxc.vcttef7a/downtmp/build.Pzh/src
59s    TLS_ARGS= -l -L
59s    testuser=debci
59s    os=Linux ci-086-4624a191 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC
    Debian 6.1.129-1 (2025-03-06) x86_64 GNU/Linux
59s    preserve_scratch=no
59s    scratchbase=/tmp/autopkgtest-lxc.vcttef7a/downtmp/build.Pzh/src/testtmp
60s PASS    00-hello
61s SKIP    acls-default (I don't know how to use your setfacl command)
62s SKIP    acls (I don't know how to use setfacl or chmod for ACLs)
64s PASS    alt-dest
65s PASS    atimes
66s PASS    backup
67s PASS    batch-mode
70s PASS    chgrp
71s PASS    chmod-option
72s PASS    chmod-temp-dir
```

[1] I am aware that writing tests is not necessarily the favorite passtime of developers and relying solely on upstream tests comes with risks.

```
73s PASS    chmod
74s SKIP    chown (Can't chown (probably need root))
75s SKIP    crtimes (Rsync is configured without crtimes support)
76s PASS    daemon-gzip-download
77s PASS    daemon-gzip-upload
78s PASS    daemon
79s PASS    delay-updates
80s PASS    delete
81s SKIP    devices (Rsync needs root/fakeroot for device tests)
82s PASS    dir-sgid
83s PASS    duplicates
85s PASS    exclude-lsh
88s PASS    exclude
89s PASS    executability
90s PASS    files-from
92s PASS    fuzzy
93s PASS    hands
94s PASS    hardlinks
97s PASS    itemize
98s PASS    longdir
100s PASS    merge
101s PASS    missing
102s PASS    mkpath
103s SKIP    protected-regular (Can't chown (probably need root))
105s PASS    relative
107s PASS    safe-links
108s PASS    ssh-basic
109s PASS    symlink-ignore
110s PASS    trimslash
111s PASS    unsafe-byname
112s PASS    unsafe-links
113s PASS    wildmatch
114s SKIP    xattrs (Unable to set an xattr)
114s ------------------------------------------------------------
114s ----- overall results:
114s       36 passed
114s       7 skipped
```

To run most/all upstream tests requires *either*

1. running the existing `upstream-tests` script as root (undesirable), *or*
2. splitting the existing `upstream-tests` script into one which runs tests without root privileges and another one for those with, *or*
3. keeping `upstream-tests` untouched and adding a new script which only

executes those upstream tests with root rights which require those (combining the `autopkgtest needs-root` restriction with the `whichtests=` param of `runtests.sh`).

I prefer the last approach as it ensures all possible future upstream tests (without root requirements) are also run without further intervention by the Debian `rsync` package maintainers.

2. Setup end-to-end `autopkgtest`s with a running ssh server and verify that rsync transfers over ssh work as expected since the upstream "only" mocks the ssh connection. I expect this to be the biggest chunk of work for this project as I am unfamiliar with running a) openssh servers b) in a CI environment.
3. If time permits, create additional tests for cases which are not yet covered by upstream nor by Debian (as part of Debian or directly upstream, as decided with the Debian maintainers on a case by case basis).

over the course of a 90h (small) GSoC project.

About me

Professionally, I am a backend developer at a financial institution in Europe. I have gotten the permission from my employer to temporarily reduce my workload/take leave to accomodate a GSoC participation, if necessary, so I can assure Debian that I will be able to dedicate the alloted time to the project.

Despite being an avid FLOSS enthusiast and user, I have not contributed to FLOSS projects outside of personal ones[2]. Applying to GSoC organizations whose software I use personally is a good way of rectifying that grievance, I think, and overcome the barriers of entry to contributing to well established projects with the guidance of a mentor.

When it comes to Linux and Debian experience, I am running Debian (and various derivatives of it) on my personal computers and have had to configure web services on Debian servers at work. So while I can navigate Linux systems, configure simple `systemd` units, and write basic shell scripts, I have no packaging knowledge whatsoever; but, I am very much willing to learn that as extending my Linux skill set is a priority to me (I've passed the LPIC 101-500 test last year and am planning to finish the certification by passing the LPIC 102-500 test later this year).

As a developer I have had to setup CI pipelines which ran tests and generated documentation but never in the context of packaging 3rd party software for Debian (with the Debian-specific toolchain) or any other distribution.

[2]You can find my personal projects at Codeberg and Sourcehut.