debmans Documentation

Release 1.0.1.dev54+gcb69fe5

Antoine Beaupré

Sep 21, 2017

Contents

1 Usage **3**
- 1.1 Examples . . . 3

2 Installation **5**

3 Design **7**
- 3.1 Components . . . 7
- 3.2 Software evaluation . . . 9
- 3.3 Infrastructure . . . 13
- 3.4 Other implementations . . . 13

4 Contributing **15**
- 4.1 Source code . . . 15
- 4.2 Issues, feature requests and security . . . 15
- 4.3 Test suite . . . 15
- 4.4 Release process . . . 16

5 API **17**
- 5.1 Extractor . . . 17
- 5.2 Renderer . . . 19
- 5.3 Search . . . 22
- 5.4 Main entry point . . . 23
- 5.5 Logger . . . 23
- 5.6 Utilities . . . 23

6 Remaining work **25**
- 6.1 Blocking . . . 25
- 6.2 Important . . . 26
- 6.3 Nice to have . . . 26
- 6.4 Internal code tasks . . . 27

7 License **29**
- 7.1 GNU AFFERO GENERAL PUBLIC LICENSE . . . 29
- 7.2 Preamble . . . 29
- 7.3 TERMS AND CONDITIONS . . . 30
- 7.4 How to Apply These Terms to Your New Programs . . . 37

8 Acknowledgements 39

9 Indices and tables 41

Python Module Index 43

Note: This project is now **DEPRECATED**. It has been superseded by the debiman project which was written after debmans was published, but without the author being aware of its existence. The result is a go program that is actually quite similar in design to `debmans` but faster and more complete. It is now the tool behind manpages.debian.org.

No further changes will be done on the source code here, which will probably live here as a reference indefinitely. More details about this story are available in this blog post.

`debmans` is a tool to extract documentation (currently manpages) from Debian packages and transform them into HTML for viewing with a normal web browser.

This suite of tools is designed to run on manpages.debian.org, manpages.ubuntu.com, linux.die.net or similar services.

Contents:

CHAPTER 1

Usage

You can do a test run of this software against your local APT cache, like this:

```
$ ( cd /var/cache/apt/archives/ && dpkg-scanpackages . | sudo tee Packages > /dev/
↪null )
dpkg-scanpackages: warning: package tar (filename ./tar_1.27.1-2+b1_amd64.deb) is␣
↪repeat;
dpkg-scanpackages: warning: ignored that one and using data from ./tar_1.27.1-
↪2+deb8u1_amd64.deb!
$ debmans -m /var/cache/apt/archives -o html -v --progress extract render site serve
extracting files matching patterns: (u'man/(?:\\w+/)?man[1-9]/.+\\.[1-9]\\w*(?:\\.gz)?
↪$',) in mirror /var/cache/apt/archives
extracting packages  [####################################]  100%
completed inspection of 564 packages, extracted 559 files
received 559 paths from extractor
rendering manpages  [################################---]   92%  0d␣
↪00:00:02man2html: unable to open or read file man1/dmenu.1
rendering manpages  [####################################]  100%
looking for static files to render in *.mdwn
Serving HTTP on 0.0.0.0 port 8000...
```

Your rendered manpages will be available at <http://localhost:8000/>.

Note that you will need `dpkg-scanpackages` from the dpkg-dev package to run the above test. The warnings can be safely ignored. `debmans` doesn't normally require `dpkg-dev` in operation if you use a properly configured mirror.

Examples

A more normal run would be to run on a regular mirror. This will extract all manpages from the given mirror and convert to HTML:

```
debmans --mirror /srv/mirror --output html/ extract render
```

This can be ran repeatedly and only extracted manpages will be rendered.

Extraction only

You can also run the process step by step, for example, this will only extract the manpages:

```
debmans -m /srv/mirror -o html extract
```

Note that extract creates a cache in the output directory to avoid re-extracting already found archives. Render also uses a caching mechanism by not converting to HTML if the HTML version is newer than the manpage. Those mechanisms can be disabled with `--no-cache`.

```
debmans -m /srv/mirror -o html --no-cache extract
```

You can make a trial run with the `-n` flag and enable more verbose logging:

```
debmans -v -n -m /srv/mirror -o manpages extract
```

Rendering only

Rendering the pages can be done separately with:

```
debmans -o html render --srcdir manpages
```

This is especially important if the extraction process was interrupted, as only part of the manpages will be rendered on subsequent runs.

Manpages are rendered using the plugin defined by the `--plugin` argument. Arbitrary plugin can be loaded provided that they comply with the *debmans.renderer.ManpageRenderer* and *debmans.renderer.CommandRenderer* public APIs.

Static pages rendering

The templating engine is fairly simple, based on Jinja2, which uses standard `{{foo}}` placeholders. The template is in `debmans/templates/template.html` and should be following debian.org's graphic design conventions.

This should generate the markdown files:

```
debmans -o html site
```

Use `--prefix` if the directory is not at the root of the host.

CHAPTER 2

Installation

`debmans` can be installed through pip with:

```
pip install debmans
```

It requires the python-apt module to be installed, which can't be done through PIP, unfortunately, see issue #845330.

Source code can also be found on Gitlab with Git or as a tarball, or on Debian's collab-maint repository.

The source code is should be installed with:

```
./setup.py install
```

It can also be ran from the source tree directly with:

```
python -m debmans render
```

The dependencies are listed in the `setup.py` file.

CHAPTER 3

Design

This page explains the design principles and decisions in the project.

The *Minimum Viable Product* (MVP) for this project is a service that creates an HTML version of all the manpages of all the packages available in Debian, for all supported suites. Basic `whatis(1)` functionality is also expected.

`apropos(1)` functionality is considered extra that can be implemented later with already existing tools.

Components

The design is split those components which map to `debmans` subcommands:

1. `extract`: extracts manpages from Debian packages
2. `render`: renders manpages into HTML
3. `site`: render a static site into HTML
4. `index`: indexes HTML pages for searching (not implemented yet)
5. `search`: the search interface (not implemented, but there is a simple "jump" Javascript tool)

There is also a `serve` command which starts a local webserver to help with development.

See the *Remaining work* file for details about the missing bits.

Extract

This part fetches all manpages from the archive and stores them on disk. This makes them usable for tools like dman that browses remote webpages.

The layout options for where to store files were:

- Ubuntu: `$DISTRIB_CODENAME/$LOCALE/man$i/$PAGE.$i.gz` (see dman)
- original codebase: `"${OUTPUTDIR}/${pooldir}/${packagename}_${version}"` (from manpage-extractor.pl)

Ubuntu's approach was chosen to avoid bitrot and follow more closely the existing filesystem layout. It also happens to be easier to implement.

The extractor uses a cache to avoid re-extracting known manpages. We use the Ubuntu layout there as well (`$outputdir/$suite/.cache/$packagename_version`), which leads to bitrot, but at least it's constrained to a suite. This will be a problem for unstable, so maybe some garbage-collection may be necessary.

Render

This converts manpages to HTML so they are readable in a web browser. This is mostly about calling one of the *Manpage converters* and then embedding the output in one of the *Templating systems*.

A simple timestamp change on the template, source and target files makes sure the files that do not need to be refreshed are skipped.

Index

This indexes HTML pages in a search engine of some sort.

In the backend, something will need to index the manpages if only to implement `apropos(1)` functionality, but eventually also full-text search. This should be modular so that new backends can be implemented as needed.

For now, we are considering reusing the existing Xapian infrastructure the Debian project already has.

The indexer would:

1. run `omindex` on the HTML tree and create a database
2. process each locale separately so they are isolated (may be tricky for `LANG=C`) and that the right stemmer is called
3. need a CGI script (provided by the xapian-omega package) to query the database - the HTML output is generated based on templates, so presumably we could reuse the existing templates.

It is assumed that Xapian can deal with large datasets (10-50GB) considering how it is used in Notmuch (my own mailbox is around ~6GB) and lists.debian.org.

We put the command name in the page `<title>` tag, the short description in `<meta description="...">` and use the magic markers (`<!--htdig_noindex-->ignored<!--/htdig_noindex-->`) to make the indexer ignore redundant bits.

See the :ref'search-engines' for more information about the various search software evaluated and the web interface.

Search

The search interface itself would be a CGI or WSGI tool (if written in Python) interface that would hook into the webserver to perform searches.

Originally, only a browser-based Javascript search tool implemented basic `whatis(1)` functionality. It looked up the manpage using a XMLHttpRequest to see if the requested page exists and redirects appropriately. It didn't look at different locales yet.

A prototype of a CGI-based approach was then written in Flask. The search engine will look at the filesystem for a given pattern and, optionnally, section, suite and locale parameters. It doesn't show all suites by default and prefers to show all matching manpages, behaving like a *names-only* `apropos(1)`.

See the *Web frameworks* section for more information about the decision to use Flask..

This should be extended to a full search interface, using `omega`'s web interface or another pluggable interfaces.

Software evaluation

This section tries to (sometimes after the fact) justify choices made in certain dependencies and software technologies used in the project.

Manpage converters

Those are the known manpage converters:

- just the plaintext output of man wrapped in `<PRE>` tags (current design)
- man2html is an old C program that ships with a bunch of CGI scripts
- there's another man2html that is a perl script, but I couldn't figure out how to use it correctly.
- w3m has another Perl script that is used by the Ubuntu site
- roffit is another perl script. the version in Debian is ancient (2012) and doesn't display the `man(1)` synopsis correctly (newer versions from github also fail)
- pandoc can't, unfortunately, read manpages (only write)
- man itself can generate an HTML version with `man -Hcat man` and the output is fairly decent, although there is no cross-referencing
- mandoc also has HTML output and is packaged in Debian

The Makefile here tests possible manpage HTML renderers. Each is timed with `time(1)` to show its performance.

Package	Timing
roffit	`0.06user 0.00system 0:00.07elapsed 96%CPU (0avgtext+0avgdata 4852maxresident)k`
w3m	`0.26user 0.01system 0:00.19elapsed 137%CPU (0avgtext+0avgdata 5456maxresident)k`
man	`1.63user 0.17system 0:01.81elapsed 99%CPU (0avgtext+0avgdata 27268maxresident)k`
man2html	`0.00user 0.00system 0:00.01elapsed 61%CPU (0avgtext+0avgdata 1568maxresident)k`
mandoc	`0.00user 0.00system 0:00.01elapsed 57%CPU (0avgtext+0avgdata 2352maxresident)k`

Note: Those statistics were created with `debmans/test/converters/Makefile` in the source tree.

Here is how the actual output compares:

Package	Correctness
roffit	SYNOPSIS fails to display correctly
w3m	includes HTTP headers, links to CGI script, all pre-formatted, no TOC
man	TOC, no cross-referencing
man2html	includes HTTP headers, links to CGI script, index at the end
mandoc	customizable links and stylesheets, no index, can avoid `<body>` tags

`man2html` was originally chosen because it is the fastest, includes an index and is not too opiniated about how the output is formatted. Unfortunately, it would fail to parse a lot of manpages, like the ones from the `gnutls` project.

Then `w3m` was used as a fallback, even though it actually calls `man` itself to do part of the rendering. It required a bunch of hacks to fix the markup.

So then the `mandoc` package was used, and it was significantly faster. In the test corpos (452 manpages), mandoc would render all pages in 23 seconds, while w3m was 5 times slower.

Time for w3m:

```
77.29user 13.02system 1:26.04elapsed 104%CPU (0avgtext+0avgdata 63800maxresident)k
```

Time for mandoc:

```
15.70user 4.75system 0:26.40elapsed 77%CPU (0avgtext+0avgdata 55992maxresident)k
```

Templating systems

Templating is necessary to turn things into HTML consistently. We chose Jinja because it seemed lightweight and simple enough. It is used by Pelican, MoinMoin 2.0, Ansible and Salt. It also seemed to be a good middleground between arbitrary code execution and static templates in this comparison. Jinja is also used by the Flask framework which is close to the Click commandline framework we are already using.

Interesting alternatives include Mako, used by Reddit and Pyramid, which is itself part of the larger Pylons project. This all seems like too much to pull in.

Genshi was also evaluated summarily later on but discarded as I found the basic tutorial to be too complicated.

It used to be possible to change the location of the templates and static files, but that was removed during some refactoring, partly because of the lack of interest from Ubuntu to reuse the software for now. If we decide to support themes, it may be simpler to use flask-themes instead of implementing our own theming system, although that seems to be a bit heavy...

Web frameworks

Flask was chosen to build a quick CGI prototype because I wanted to experiment with it, but also because it was integrating with the templating system already chosen (Jinja) and the commandline framework tool (Click). It also competes reasonably with other Python frameworks in the techempower benchmarks. I have found the decorators in flask to be really intuitive and easy to use and while I had to bounce around for a while to merge the search engine with the regular static file server, in the end it turned out to be a simple implementation.

Other alternatives that were considered were:

- builtin HTTPSserver in Python: used at first, but did only static files
- builtin `cgi` module: wanted to try a framework instead of parsing CGI arguments and dealing with content-types by hand
- Pyramid: too complicated
- Bottle: interesting, looks simpler than Flask, if less popular?
- Django, Plone, etc: too much overhead

Search engines

Various search were evaluated:

- Xapian:
  - used by Notmuch, craigslist, search.debian.org, lists.debian.org, wiki.debian.org, old gmane.org
  - no web API, would need to index directly through the Python API

  - harder to use
  - no extra server necessary
  - internal knowledge already present in debian.org
  - bound to use their CGI interface (Omega)
  - written in C++
- Lucene / Solr:
  - requires another server and API communications
  - per-index user/password access control
  - mature solution that has been around for a long time
  - may support indexing HTML?
  - JSON or XML data entry
  - large contributor community
  - usually faster than ES
  - based on Apache Lucene
  - written in Java
  - used by Netflix, Cnet
- Elasticsearch:
  - requires a server
  - packaged in Debian
  - REST/JSON API
  - documentation may be lacking when compared so Solr, e.g. had to ask on IRC to see if `_id` can be a string (yes, it can)
  - Python API, but could also be operated with a Javascript library
  - may be easier to scale than Solr
  - performance comparable with Solr
  - was designed to replace Solr
  - supports indexing HTML directly by stripping tags and entities
  - based on Apache Lucene as well
  - written in Java as well
  - requires a CLA for contributing
  - used by Github, Foursquare, presumably new gmane.org
  - created in 2010, ~18 months support lifetime?
- whoosh: used by moinmoin 2.0
- sphinx: not well known, ignored
- mngosearch: considered dead, ignored
- homegrown:

  - Ubuntu's tool uses a simple Python CGI to search page names, and Google Search for full-text search
  - codesearch uses Postgresql
  - David's uses sqlite
  - Readthedocs has a custom-built Javascript-based search engine
  - we could use a simple Flask REST API for searches, but then the extractor (or renderer?) would need to write stuff to some database - sqlite reads fails when writing, so maybe not a good candidate?

Sources:

- http://stackoverflow.com/questions/10213009/solr-vs-elasticsearch
- http://solr-vs-elasticsearch.com/

Xapian examples

Index all documents in `html/`:

```
$ ominxed --url / --db search --mime-type=gz:ignore html/
11.03user 0.36system 0:12.27elapsed 92%CPU (0avgtext+0avgdata 29952maxresident)k
18056inputs+15688outputs (0major+7884minor)pagefaults 0swaps
```

`--url` is the equivalent of the renderer's `--prefix`, `--db` is a directory where the database will end up and `--mime-type` is to ignore raw manpages.

Second runs are much faster:

```
$ time omindex --url / --db search --mime-type=gz:ignore html/
0.01user 0.00system 0:00.02elapsed 88%CPU (0avgtext+0avgdata 4656maxresident)k
0inputs+0outputs (0major+300minor)pagefaults 0swaps
```

Display information about the search database:

```
$ delve search.db
UUID = 6fd4d4ab-2529-4d67-bbff-32b88fd888fa
number of documents = 452
average document length = 3976.96
document length lower bound = 78
document length upper bound = 243183
highest document id ever used = 452
has positional information = true
```

Example searches:

```
$ quest -d search.db man2html | grep ^url
url=/cache/man/fr/man1/man2html.1.html
url=/cache/man/man1/man2html.1.html
url=/cache/man/ro/man1/man2html.1.html
url=/cache/man/it/man1/man2html.1.html
url=/cache/man/el/man1/man2html.1.html
$ quest -d search.db setreg | grep ^url
url=/cache/man/man1/setreg.1.html
url=/cache/man/man1/mozroots.1.html
url=/cache/man/man1/chktrust.1.html
url=/cache/man/man1/certmgr.1.html
```

This would search only `<title>` fields: `--prefix=title:S 'title:foo'`.

Infrastructure

Debian.org

Note: This may be out of date. The purpose of this section is to explained how this is should be deployed on Debian's infrastructure.

At least the extractor and renderer would run on `manziarly`. The output would be stored on the static.d.o CDN (see below). parts 3 could be a separate (pair or?) server(s?) to run the search cluster.

In the above setup, `manziarly` would be a master server for static file servers in the Debian.org infrastructure. Files saved there would be rsync'd to multiple frontend servers. How this is configured is detailed in the static-mirroring DSA documentation, but basically, we would need to ask the DSA team for an extra entry for manpages.d.o there to server static files.

Gitlab vs Alioth

The project was originally hosted in the Collaborative Maintenance repositories, but those quickly showed their limitations, which included lack of continuous integration, issue tracking and automatic rendering of markdown files.

A project was created on Gitlab for this purpose, in anarcat's personnal repositories (for now). On Gitlab, the project "mirrors" the public git URL of the `collab-maint` repo. On collab-maint, there is a cronjob in my personnal account which runs this command to synchronize the changes from Gitlab at the 17th minute of the hour:

```
git -C /git/collab-maint/debmans.git fetch --quiet gitlab master:master
```

This was found to be the best compromise in adding the extra gitlab features while still keeping access threshold for Debian members low. Do note that there is no conflict resolution whatsoever on collab-maint's side, and the behavior of Gitlab in case of conflicts isn't determined yet. This may require manual fixing of merge conflicts.

Other implementations

There were already three known implementations of "man to web" archive generators when this project was started.

After careful consideration of existing alternatives, it was determined it was easier and simpler to write a clean-room implementation, based in part on the lessons learned from the existing implementations and the more mature `debsources` project.

Original manpages.d.o codebase

The original codebase is a set of Perl and bash CGI scripts that dynamically generate (and search through) manpages.

The original codebase extracts manpages with `dpkg --fsys-tarfile` and the tar `tar` commands. It also creates indexes using `man -k` for future searches. Manpages are stored in a directory for each `package-version` pair, so it doesn't garbage-collect disappeared manpages. It also appears that packages are always extracted, even if they had been parsed before.

The CGI script just calls `man` and outputs plain text wrapped in `<PRE>` tags without any cross-referencing or further formatting.

There is also a copy of the Ubuntu scripts in the source code.

Ubuntu

Ubuntu has their own manpage repository at https://manpages.ubuntu.com/. Their codebase is partly Python, Perl and Bash.

It looks like there's a bash ''and'' python implementation of the same thing. They process the whole archive on the local filesystem and create a timestamp file for every package found, which avoids processing packages repeatedly (but all packages from the `Packages` listing are `stat`'d at every run). In the bash version, the manpages are extracted with `dpkg -x`, in the Python version as well, athough it uses the `apt` python package to list files. It uses a simple regex (`^usr/share/man/.*\.gz$`) to find manpages.

It keeps a cache of the md5sum of the package in `"$PUBLIC_HTML_DIR/manpages/$dist/.cache/$name` to avoid looking at known packages. The bash version only looks at the timestamp of the file versus the package, and only checks at the modification *year*.

To generate the HTML version of the manpage, both programs use the `/usr/lib/w3m/cgi-bin/w3mman2html.cgi` shipped with the w3m package.

Seach is operated by a custom Python script that looks through manpages filenames or uses Google to do a full text search.

dgilman codebase

A new codebase written by dgilman is available in github. It is a simple Python script with a sqlite backend. It extracts the tarfile with `dpkg --fsys-tarfile` then parses it with the Python `tarfile` library. It uses rather complicated regexes to find manpages and stores various apropos and metadata about manpages in the sqlite database. All manpages are unconditionnally extracted.

OpenBSD

The OpenBSD project has a man.cgi(8) program that powers the whole application behind man.openbsd.org. It uses mandoc(1) to format manpages, as the manual system in OpenBSD has native support for HTML output. This is linked directly in the CGI, which is written in C.

FreeBSD

Wolfram Schneider

The man.freebsd.org site is powererd by a perl script written by Wolfram Schneider which parses the output of the `man(1)` command directly. See the help page for more information.

CHAPTER 4

Contributing

The debmans project welcomes contributions of all sort.

A broader discussion about the service is in the manpages.debian.org wiki page.

Source code

The source code should be available in this Gitlab repository. Gitlab was chosen because this project hopes to be reused by other Debian derivatives and we prefer to have a neutral space to develop the software. Gitlab also provides an issue queue to report a new issue.

Debian developers not desiring to use a semi-proprietary platform like Gitlab may still use the collab-maint repository, see CollaborativeMaintenance for more information on how to collaborate on those repositories. The two repositories are synchronized every hour.

Issues, feature requests and security

Patches can be sent by email (see below) or with Gitlab Merge Requests.

Security issues can be privately reported on Gitlab or directly to the author, at anarcat@debian.org. You can use this OpenPGP public key to encrypt messages for confidential security issues:

```
8DC9 01CE 6414 6C04 8AD5  0FBB 7921 5252 7B75 921E
```

Test suite

Make sure you run tests before you send a patch. Also add tests for new functionalities you add or for bugs you find. Tests are done with pytest:

```
py.test
```

Test coverage can be verified with:

```
py.test --cov debmans
```

Exact lines lacking test coverage are available in the HTML report that can be generated with:

```
py.test --cov debmans --cov-report html
```

The test suite requires the `dget` and `dpkg-scansources` commands from the devscripts package to build a test mirror. Network access is required if packages are not locally cached in `/var/cache/apt/archives` or where `dget` can find them (see `DGET_PATH` in the `dget(1)` manpage).

Release process

1. make sure tests pass (see above for details):

   ```
   py.test
   ```

2. create a signed and annotated tag:

   ```
   git tag -s x.y.z
   ```

3. build and test Python "wheel":

   ```
   python setup.py bdist_wheel
   sudo pip install dist/*.whl
   debmans --version
   debmans -m /var/cache/apt/archives -o html -v --progress extract render site serve
   sudo pip uninstall debmans
   ```

4. push commits and tags to the git repository:

   ```
   git push
   git push --tags
   ```

5. publish Python "wheel" on PyPI:

   ```
   twine upload dist/*
   ```

6. announce on debian-doc@lists.debian.org

CHAPTER 5

API

This is the API documentation of Debmans. It should be stable across major releases. See the *Design* document for more details about the design.

Extractor

The extractor processes Debian packages and extracts specific patterns into a target directory. It uses a cache file that is named according to the package name and version to avoid the costly operation of opening the same package file multiple times.

debmans.extractor.**SOURCES_COMP_FMTS = ['gz', 'bz2', 'xz']**
supported compression formats for Sources files. Order does matter: formats appearing early in the list will be preferred to those appearing later

class debmans.extractor.**PackageExtractor**(*regex=[]*, *root='.'*, *destdir=None*, *dryrun=False*)
extract certain files from debian packages

apt_cache = None
this takes one second to load, swallow the cost now instead of for every package

files = None
the files written during extraction

root = None
the root of the mirror to look files in

destdir = None
the directory where to write extracted files

dryrun = None
do not write anything if True

patterns = None
regex patterns of files to extract

regexes
compiled static cache of regex patterns

to regenerate this when patterns is changed, set `_regexes` to None.

write_file(*item*, *data*)
callback to actually write files in archive

this will check for the internal regex list and write the given file in *destdir*, creating missing directories as needed.

only the part that is matching the pattern is extracted, unless the pattern features a `path` group, in which case only *that* part is then extracted.

extract(*pkg*, *destdir=None*, *cache=True*)
extract matching patterns into destdir

Parameters

- **pkg** (*debian.deb822.Deb822*) – a package dictionnary with fields like Filename, Package and Version at least.
- **destdir** (*str or None to default to the path given in constructor*) – where to store the extracted files
- **cache** (*bool*) – if we should check and create the cache file (a PackageCacheFile)

Returns extracted files paths

Return type list

Raises ***PackageCorruptedError*** – if apt fails to extract the file

class debmans.extractor.**PackageCacheFile**(*destdir*, *pkg*)
a cache file to see if we have inspected a package before

this creates an empty named pkgname-version in the given destdir on create. there are also facilities to check existence.

it is assumed that if there is no version change, no change is required in the man pages as well.

it is not possible to atomically check existence just yet.

this will leave stray cache files behind.

filename
the full path to the cache file

exists()
if the cache file exists

create()
create the cache file with the given field as content

class debmans.extractor.**PackageMirror**(*path*)
inspect a Debian mirror for binary packages

this is a modified replica of debsources's SourceMirror class. Ideally, this would be merged back into the original class as a derivative.

packages
return the mirror packages as a set of `<package, version>` pairs

Note: This is just like calling *ls()*, except there is a cache to avoid calling it multiple times.

releases
list of releases in this repository

Returns (codename -> description) mappings. description is in the format `X.Y codename (stable)`, unless no matching Release file was found, in which case it can be just `codename`, which is taken from the *packages()* list of suites.

Return type dict

ls()
iterate over packages found in the mirror

this will yield `(suite, pkg)` pairs. the `suite` is determined by looking at the name of the 4th directory up from where the Packages file is located, as is standard in complete apt repositories. this may yield weird codenames when working with ad-hoc repositories as the chosen name may be a bit random depending on your directory structure.

Returns `(suite, pkg)` tuples for each package found.

Return type `pkg` is a deb822 fragment, `suite` is a string.

exception debmans.extractor.**DebmirrorError**
runtime error when using a local Debian mirror

exception debmans.extractor.**PackageCorruptedError**
runtime error when using a local Debian mirror

Note: the documentation for `click` functions is incomplete. they should actually be turned into usage page and manpages, see this issue for details.

---

debmans.extractor.**extract**()
extract manpages from Debian binary packages in mirror

iterate over all binary packages found in the mirror, and extract each included manpage to the output directory.

Renderer

The Renderer module takes care of turning extracted documentation into HTML format. It uses Jinja templates and simple timestamp-based caching.

class debmans.renderer.**JinjaRenderer**(*template*, *cache=True*, *dryrun=False*)
render Jinja templates using given parameters, caching and simulation

this is basically an extension of the Template class, but extended so we can easily pass paths (instead of strings) in and out.

Todo

we should probably have derived Template directly here.

---

template = None
template to use to render the data

cache = None
if we should check timestamps before writing

dryrun = None
if True, do not write

source = None
source file currently processed

generated_time()
handy function to add timestamp to footers

render(*target*, ***data*)
render template with given data

if `pageinfo` isn't provided in `data`, it is set to the output of *generated_time()*.

Parameters

- **target** (*str*) – path to the target file
- **data** (*dict*) – set of parameters passed to `render()`

uptodate(*target*)
check if the target file is newer than template

also checks the *source* attribute if it is set, which allos for subclasses to add a file to check.

class debmans.renderer.**MarkdownRenderer**(*template*, *cache=True*, *dryrun=False*)
render markdown source files with a jinja template

render(*source*, *target*, ***data*)
render the given source file

Parameters

- **source** (*str*) – path to the Markdown source file
- **target** (*str*) – passed to *JinjaRenderer.render()*

class debmans.renderer.**CommandRenderer**(*template*, *command=None*, *cache=True*, *dryrun=False*)
a simple template-based rendering system

a file is passed as an argument to a command and the output is written into the given template, in the *{{content}}* Jinja2 element.

this is meant to be subclassed in command-specific renderers.

those can also not even be command-based, as long as they have the following parameters:

- `pattern`: regular expression pattern for this class
- `render(source, target, **data)`: render the given source file into the target file, with the attached Jinja data. at least `content` is expected in there, but `description` and `title` are also encouraged, those should match the template.

postprocess(*data*)
modify the data sent to the template after execution

this allows subclasses to intervene between the command call and the render call.

by default does nothing

render(*source*, *target*, ***data*)
render the given source file using external command defined in constructor

does not call command in `dryrun` mode.

Todo

support %(target)s instead of standard output, if necessary?

Parameters

- **source** (*str*) – path to the source file
- **target** (*str*) – path to the output file
- **data** (*dict*) – remaining arguments passed as is to *JinjaRenderer.render()*

:raises *CommandRendererError*: if command fails to convert given page

exception debmans.renderer.**CommandRendererError**
error raised when man2html fails to render the manpage

class debmans.renderer.**ManpageRenderer**
abstract class to store the manpage regex pattern

pattern = '/(?:(?P<suite>\\w+)/)?(?P<path>man/(?:(?P<locale>\\w+)/)?man[1-9]/(?P<name>.+)\\.(?P<section>[1-9]\\w*)
default pattern for manpages

class debmans.renderer.**W3mRenderer**(*template*, *command=None*, *cache=True*, *dryrun=False*)
render manpages with w3m

command = '/usr/lib/w3m/cgi-bin/w3mman2html.cgi "quit=1&local=%(source)s"'
path to w3m converter

postprocess(*data*)
process w3m parser output

class debmans.renderer.**MandocRenderer**(*template*, *command=None*, *cache=True*, *dryrun=False*)
render pages with mandoc

Todo

this assumes cross-references are done with the `.Xr` macro, which is unfortunately not often the case in my tests. so some manual cross-ref will be required here.

Todo

croaks on the `kodi(1)` manpage, a weird redirect, which we should handle manually here. the fix, according to `mandoc(1)` is to chdir to the correct relative directory. looking at `zshall(1)`, `.so` looks like an "include" directive.

class debmans.renderer.**Man2htmlRenderer**(*template*, *command=None*, *cache=True*, *dryrun=False*)
render manpages with man2html

postprocess(*data*)
process man2html output

•it doesn't return proper exit codes, look for Status header instead. Anything 40X is bad.

•the title is in the `NAME` level two header (`<h2>`)

•keep only the inside of the `<body>` tag

- rewrite URLs to point to the right place
- remove attribution

debmans.renderer.**DefaultManpageRenderer**
quick switch to toggle default manpage rendering implementation

alias of *MandocRenderer*

debmans.renderer.**find_files**(*directory*, *patterns*)
look for file paterns in the given directory and return the right command to run

Todo

this may be slow in large directories and may be reimplemented with `os.scandir()` if we ever depend on Python 3.5 or later.

Returns `module`, `path` tuples

Return type list

debmans.renderer.**match_jobs**(*files*, *patterns*)
dispatch the right command for the matching pattern

Parameters

- **files** (*list*) – list of file paths to inspect
- **patterns** (*list*) – list of tuples (`cls`, `regex`). `regex` is a compiled regex patterns to match against the pathnames, `cls` is a CommandRenderer subclass to run

Returns `module`, `path` tuples

Return type list

debmans.renderer.**render**()
render documentation to HTML

this looks for patterns matching a certain regex in the given SRCDIR directory

Note: this assumes files have an extension that should be stripped. for manpages, this should generally be `.gz`. if manpages are not compressed, this will break section support.

Todo

document that compressed manpages are mandatory

debmans.renderer.**site**()
render the whole static site

Search

The search module takes care of indexing and searching manpages. It currently has very limited functionality.

Main entry point

The main entry point of `debmans` is in the `debmans.__main__` module. This is to make it possible to call `debmans` directly from the source code through the Python interpreter with:

```
python -m debmans
```

All this code is here rather than in `__init__.py` to avoid requiring too many dependencies in the base module, which contains useful metadata for `setup.py`.

This uses the `click` module to define the base command and options, which then get passed to subcommands through the `obj` parameter, see `pass_obj()` in the click documentation.

Logger

This is a simple helper module to configure the `logging` module consistently.

`debmans.logger.`**`setup_logging`**(*name='debmans'*, *level='info'*, *syslog=False*, *stream=None*)
setup logging module according to the arguments provided

Utilities

Those are various utilities reused in multiple modules that did not fit anywhere else. various utilities for debmans

`debmans.utils.`**`find_parent_module`**()
find the name of a the first module calling this module

if we cannot find it, we return the current module's name (__name__) instead.

`debmans.utils.`**`find_static_file`**(*path*)
locate a file in the distribution

this will look in the shipped files in the package

this assumes the files are at the root of the package or the source tree (if not packaged)

this does not check if the file actually exists.

Parameters **`path`** (*`str`*) – path for the file, relative to the source tree root

Returns the absolute path to the file

`debmans.utils.`**`mkdirp`**(*path*)
make directories without error

this is a simple wrapper around `os.makedirs()` to avoid failing if the directory already exists.

it also logs to the `DEBUG` `logging` facility when a directory is created.

CHAPTER 6

Remaining work

Those are the known issues and limitations of the `debmans` software, serving as an internal, ad-hoc issue tracker.

Blocking

Those are the things that need to be done to complete the restoration of the manpages service.

- setup virtual host configuration, include:
  - redirections for previous links: `/man/intro`, `/man/1/intro`, `/1/intro`, `/cgi-bin/man.cgi?query=intro&apropos=1&manpath=Debian+Sid&format=html&locale=en`
  - 404 handler to point to 404.html
  - edit cii badge when done, future section
  - CSP and various other SSL flags?
  - CGI config: currently implemented with Flask, see the self-hosted options for a list of deployment possibilities. simplest is probably to provide WSGI and CGI shims. WSGI works well in apache (and also works in Nginx) according digital ocean CGI works everywhere. Also: need a way to pass configs (everything in `debmans.search.main()` actually) between apache and CGI
- test run on manziarly
  - requires access to manpages group? see RT#6485
  - missing dependencies to run properly (even as plain user): setuptools, click, apt, debian, patch to mirror/debian.org.git
- provide DSA team with Puppet ruleset (see dsa-puppet manifests) or config documentation
- ask DSA to deploy the new code, test
- if it works, fix the `manpages.debian.org` DNS to point to the static.d.o DNS. at this point, the MVP is in place

Important

Those are not part of the Minimum Viable Product, but would be important to implement to make this software complete.

- search functionality, in that order
  1. whatis(1): find manpages by name (done by Flask app)
  2. apropos(1): find manpages by description
  3. full text search
- i18n: we parse all languages, but should auto-detect the web browser's language with fallbacks and everything. Apache auto-negociation? Could be like debian.org language menus... The trick is to guess the language list again, similar to the suites issue. The new search box can lookup different locales in the backend, but doesn't have the list of locales in the frontend.

Nice to have

Those are not really necessary but could improve the service.

- unify site and render? a .mdwn file is like a .1.gz file, basically, except it's not extracted from a .deb
- add `__str__()` to classes
- 100% test coverage (about 80% now), edit cii badge when done (quality section)
- move `apt_cache` optimizations upstream
- rotated –logfile
- add sections browser to the index page?
- debian packaging, edit cii badge when done (future and other sections)
- use tox to test against different py envs
- CII suggestions, edit cii badge when done:
  - continuous integration through Gitlab CI? (quality section)
  - hook pyflakes in test suite (quality section)
  - static code analysis with pylint (analysis section)
- embed test suite in main program
- consider a plugin system for extending to more than manpages, would provide the default for `--plugin`
  - pluggy: used by py.test, tox and devpi
  - yapsy
  - PluginBase
  - plugnplay
  - click-plugins: relevant only to add new commands
  - SO also suggests using the standard library `imp.load_module()` or just the builtin `__import__()`, see also the PyPA documentation on how to discover plugins

Possible optimizations

Optimization ideas:

- look at the archive's `Contents-*` files to choose which packages to extract
- extract only targeted files from the archive instead of iterating over it? not sure it's an improvement...
- use os.scandir() where relevant, instead of os.walk() and stat
- use multiprocessor.Pool for background job rendering? maybe by firing up rendering as soon as pages are created
- use md5sums to check if files were modified, edit cii badge when implemented (security section)
- pre-compile all regexes

Those will be implemented as needed, remember:

> Premature optimization is the root of all evil. – Donald Knuth

Internal code tasks

Those are `todo` items extracted from the code. Priority of those is indeterminate unless otherwise noted.

Todo

we should probably have derived Template directly here.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/checkouts/latest/debmans/renderer.py:docs
of debmans.renderer.JinjaRenderer, line 7.)

Todo

support %(target)s instead of standard output, if necessary?

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/checkouts/latest/debmans/renderer.py:docs
of debmans.renderer.CommandRenderer.render, line 6.)

Todo

this assumes cross-references are done with the `.Xr` macro, which is unfortunately not often the case in my tests. so some manual cross-ref will be required here.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/checkouts/latest/debmans/renderer.py:docs
of debmans.renderer.MandocRenderer, line 3.)

Todo

croaks on the `kodi(1)` manpage, a weird redirect, which we should handle manually here. the fix, according to `mandoc(1)` is to chdir to the correct relative directory. looking at `zshall(1)`, `.so` looks like an "include" directive.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/checkouts/latest/debmans/renderer.py:docst
of debmans.renderer.MandocRenderer, line 7.)

Todo

this may be slow in large directories and may be reimplemented with `os.scandir()` if we ever depend on Python 3.5 or later.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/checkouts/latest/debmans/renderer.py:docst
of debmans.renderer.find_files, line 4.)

Todo

document that compressed manpages are mandatory

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/debmans/envs/latest/local/lib/python2.7/site-packages/click-6.7-py2.7.egg/click/core.py:docstring of debmans.renderer.render, line 11.)

CHAPTER 7

License

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of

the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- 1. The work must carry prominent notices stating that you modified it, and giving a relevant date.
- 2. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- 3. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- 4. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- 1. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- 2. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- 3. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- 4. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- 5. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- 1. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

- 2. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- 3. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- 4. Limiting the use for publicity purposes of names of licensors or authors of the material; or
- 5. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- 6. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies

made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR

IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could

display a “Source” link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

CHAPTER 8

Acknowledgements

`debmans` was written by Antoine Beaupré and is licensed under the Affero GPLv3, see *License* for the complete license.

Parts of this software were inspired by a review of existing tools, most notably the Ubuntu manpages converter and dgilman's converter. The sources.debian.net service source code and design was also directly used in some parts, which is partly why `debmans` is licensed under the AGPL.

Also thanks to Paul Wise for nudging me along and all the help navigating the various tools and protocols to make all this work.

This project mostly follows the Core Infrastructure best practices, see the full report for details.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

debmans.extractor, 17
debmans.logger, 23
debmans.renderer, 19
debmans.utils, 23

Index

A

apt_cache (debmans.extractor.PackageExtractor attribute), 17

C

cache (debmans.renderer.JinjaRenderer attribute), 19
command (debmans.renderer.W3mRenderer attribute), 21
CommandRenderer (class in debmans.renderer), 20
CommandRendererError, 21
create() (debmans.extractor.PackageCacheFile method), 18

D

debmans.extractor (module), 17
debmans.logger (module), 23
debmans.renderer (module), 19
debmans.utils (module), 23
DebmirrorError, 19
DefaultManpageRenderer (in module debmans.renderer), 22
destdir (debmans.extractor.PackageExtractor attribute), 17
dryrun (debmans.extractor.PackageExtractor attribute), 17
dryrun (debmans.renderer.JinjaRenderer attribute), 19

E

exists() (debmans.extractor.PackageCacheFile method), 18
extract() (debmans.extractor.PackageExtractor method), 18
extract() (in module debmans.extractor), 19

F

filename (debmans.extractor.PackageCacheFile attribute), 18
files (debmans.extractor.PackageExtractor attribute), 17
find_files() (in module debmans.renderer), 22
find_parent_module() (in module debmans.utils), 23
find_static_file() (in module debmans.utils), 23

G

generated_time() (debmans.renderer.JinjaRenderer method), 20

J

JinjaRenderer (class in debmans.renderer), 19

L

ls() (debmans.extractor.PackageMirror method), 19

M

Man2htmlRenderer (class in debmans.renderer), 21
MandocRenderer (class in debmans.renderer), 21
ManpageRenderer (class in debmans.renderer), 21
MarkdownRenderer (class in debmans.renderer), 20
match_jobs() (in module debmans.renderer), 22
mkdirp() (in module debmans.utils), 23

P

PackageCacheFile (class in debmans.extractor), 18
PackageCorruptedError, 19
PackageExtractor (class in debmans.extractor), 17
PackageMirror (class in debmans.extractor), 18
packages (debmans.extractor.PackageMirror attribute), 18
pattern (debmans.renderer.ManpageRenderer attribute), 21
patterns (debmans.extractor.PackageExtractor attribute), 17
postprocess() (debmans.renderer.CommandRenderer method), 20
postprocess() (debmans.renderer.Man2htmlRenderer method), 21
postprocess() (debmans.renderer.W3mRenderer method), 21

R

regexes (debmans.extractor.PackageExtractor attribute), 17
releases (debmans.extractor.PackageMirror attribute), 18
render() (debmans.renderer.CommandRenderer method), 20
render() (debmans.renderer.JinjaRenderer method), 20
render() (debmans.renderer.MarkdownRenderer method), 20
render() (in module debmans.renderer), 22
root (debmans.extractor.PackageExtractor attribute), 17

S

setup_logging() (in module debmans.logger), 23
site() (in module debmans.renderer), 22
source (debmans.renderer.JinjaRenderer attribute), 20
SOURCES_COMP_FMTS (in module debmans.extractor), 17

T

template (debmans.renderer.JinjaRenderer attribute), 19

U

uptodate() (debmans.renderer.JinjaRenderer method), 20

W

W3mRenderer (class in debmans.renderer), 21
write_file() (debmans.extractor.PackageExtractor method), 18