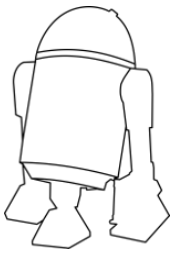
Paperwork Hacking Guide

Contents

1 This is not the document you're looking for

This is an entirely technical document. If you're not looking to contribute to Paperwork's code, this is not the document you're looking for.



This document assumes you're already familiar with Git, Python 3, and GitHub.

2 Please ask before doing

3 Git branches

In the repository, there are usually the following branches:

- *stable* : Matches the latest release version. May also include some bug fixes that will be included in the next update (x.y.z ; ex: 1.2.2). No new features allowed.
- *unstable* : Will become the next version (x.y.0 ; ex: 1.3.0) of Paperwork. New features go here. This branch hasn't been tested yet and things may be broken.
- *testing* : When the unstable branch contains all the wanted features, the branch *testing* is created from the branch *unstable*. Only bug fixes, translations updates and documentation updates are allowed in this branch. This branch is temporary and usually lives for about 1 month. After 1 month, the version is released, and this branch is merged in the branch *stable*.
- *wip-<XXX>* : Branches for work in progress. They are temporary. There are no specific rules for those branches.

4 Coding rules

4.1 PEP8

Stick to the PEP8 as much as possible. Pull requests that does not respect the PEP8 will be rejected.

- Lines are at most 80 characters long (not kidding.)
- Indentation is done using 4 spaces

If you see code that does not respect the PEP8, feel free to fix it. However, please make a dedicated commit.

If the content of a file does not respect the PEP8 and you want to modify it, please *stick to the PEP8*. Do not use the style of the file.

4.2 Commits

One commit = one change.

Try to make your commit message clear but concise.

When writing your commit message, please keep in mind the Git history is about the whole repository. Therefore, please give some context and indicate which part of Paperwork has been touched by your commit.

4.3 Comments

There are no rules regarding comments. Please just use your common sense.

If your code is complex and not commented enough, it will be unmaintainable and will be *removed*. It already happened !

5 Paperwork itself

The code is split in two pieces:

- *backend* : Everything related to document and work directory management. May depend on various things but *not* GTK+
- *frontend* (aka *paperwork-gui* or .. *Paperwork*): The GUI. Entirely dependent on GTK+

5.1 Paperwork-backend

See the user manual (section "advanced use") for a description of the work directory organization.

5.2 Paperwork-gui

6 Development tips

6.1 Configuration file for development only

Paperwork looks for a '*paperwork.conf*' in the current work directory before looking for a '*~/.config/.paperwork.conf*' in your home directory. So if you want to use a different configuration file and/or a different set of documents for development than for your real-life work, just copy your '*~/.config/paperwork.conf*' to '*./paperwork.conf*'. Note that the settings dialog will also take care of updating '*./paperwork.conf*' instead of '*~/.config/paperwork.conf*'.

6.1.1 Verbose

You can use the environment variable '*PAPERWORK_VERBOSE*' to increase or decrease the logging level. The accepted values are: *DEBUG*, *INFO*, *WARNING*, *ERROR*.

6.1.2 PEP8 and Pylint

There is a tool called '*pep8*' (*apt install pep8*). Use it.

Another tool is 'pylint'. A pylintrc is provided for Paperwork:

```
$ cd src ; pylint --rcfile=../pylintrc *
```

7 Dependencies and constraints

7.1 GLib / GObject / GTK

GTK+ is not thread-safe.

7.2 Pillow

Thread-safe.

7.3 Whoosh

Thread-safe. Lock the GIL.

7.4 Poppler

Not thread-safe.

7.5 PyOCR

Thread-safe.

7.6 Pyinsane2, libsane, and scanner drivers

Not thread-safe.

7.7 Libpillowfight / pypillowfight

Thread-safe.

8 Threads safety

Thread safety is a major issue in Paperwork. We need threads to keep the GUI smooth, but unfortunately, a lot of Paperwork dependencies are not thread-safe.

A job scheduling mechanism has been implemented (see *src/paperwork/frontend/util/jobs.py*). The idea here is to run most of the jobs in the same thread (whenever possible).

Each Job object represents a task to do. Some jobs can be stopped and resumed later (for instance JobDocThumbnailer). Each Job instance is used once and only once.

JobFactories instanciate Jobs. They are also used to keep the job recognizable.

Jobs are passed to JobSchedulers. Each JobScheduler represents a thread. They accept jobs using the method *schedule()*. The job with the higher priority is run first. If the job added to the scheduler has an higher priority than the active one, the scheduler will try to stop the active one and run it back later.

Jobs can be cancelled (assuming they are stoppable or not active yet). A single job can be canceled, or all the jobs from a given factory.

There is one main scheduler (called *main*). The main scheduler is the one used to access all the documents contents and the index.

Note that there are other threads running. For instance, there are the thread of PyInsane and the GTK+/GLib main loop.

9 Most complex pieces

9.1 Scan process

9.2 OCR process

9.3 Index updates

9.4 Searching

10 Tests

11 Windows build

12 Versioning

Version have the following syntax: <M>[.<m>[.<U>[<i>][-<extra>]]]

- M = Major version : Major changes made / product completed.
- m = minor version : Minor changes made.
- U = update version : Only bug fixes
- i = installer revision : Only the installer has been changed (Windows only)
- Extra : (Git tags only) May match a Git tag done before a big change (for instance: before switching from Gtk 2 to Gtk 3). If extra == "git", indicates a version directly taken from the git repository.

13 Documentation