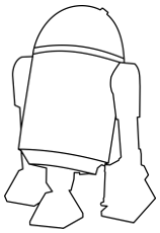
# Translating Paperwork

## Contents

# 1 This is not the document you're looking for

**This document describes how to translate the user interface of Paperwork in other languages. Unless this is what you want to do, this is not the document you're looking for.**



# 2 Requirements

- a GNU/Linux system (Ubuntu, Debian, Fedora, etc)
- Git

```
apt install git git-gui # Ubuntu/Debian
```

- Intltool

```
apt install intltool  # Ubuntu/Debian
```

- Basic knowledge of the shell

# 3 Getting a copy of Paperwork

## 3.1 Forking


Fig. 1: Forking Paperwork on GitHub

**The repository *Paperwork* contains the GTK+ user interface. This is the only part that requires translations. You will need a copy of this repository on your Github account in order to submit a pull request later.**

## 3.2 Downloading

**You must have configured GitHub SSH access for these commands to work.**

```
# download a copy of Paperwork repository
$ git clone git@github.com:[your login]/paperwork.git
$ cd paperwork
```

**If you already have a copy of Paperwork repository, you can simply update it. You have to update first the copy stored on your Github account. You can then use the following commands to update the copy on your computer:**

```
$ cd paperwork
$ git pull
```

## 3.3 Selecting the branch

**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*.

**Translations updates are allowed on any branches. However, it is recommended to make translations on the branch *testing* while it exists. You have to subscribe to the mailing-list to know when it is created or merged. If the branch *testing* doesn't exist at the moment, you're advised to work on the branch *unstable*.**

```
$ git checkout [branch]  # switch the wanted branch
```

# 4 Translating

## 4.1 Extracting the strings to translate

**Strings to translate are in the program itself. They have to be extracted for translation. A script has been included in Paperwork repository for this purpose.**

```
$ ./localize.sh upd-po
```

**Strings to translate and their translations (if they already have one) are stored in the *locale/[language].po*. You can use any file editor to modify this file (GEdit, Vim, etc).**

## 4.2 Updating the translations

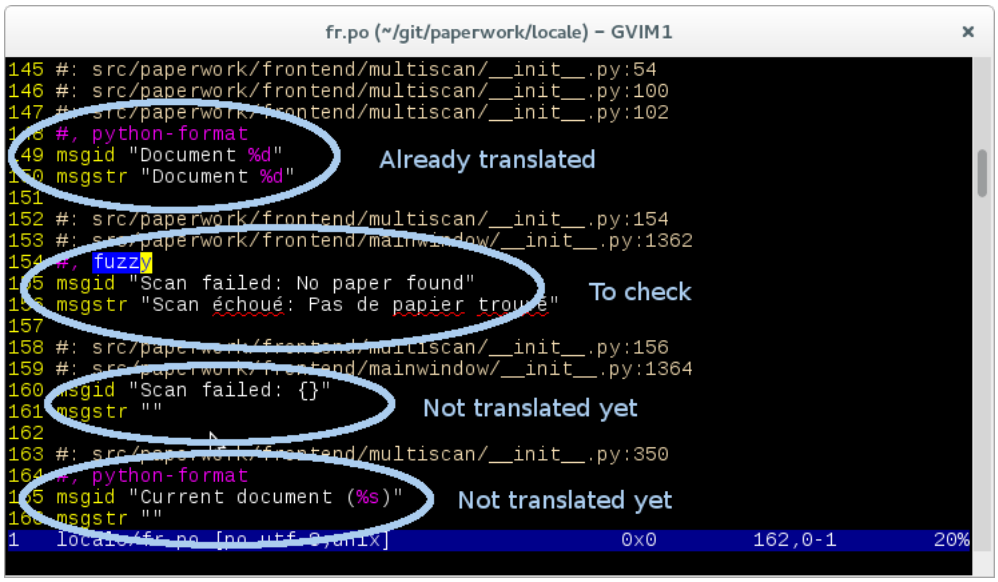



Fig. 2: Example of .po file

**Here are the steps to follow:**

- Open the .po file corresponding to your language (*locale/fr.po* for French for instance)
- Look for the keyword "*fuzzy*" : When a string is tagged with the keyword "*fuzzy*", it means the string didn't have any translations yet, but intltool found one that is really close. It is *often wrong* !
- Look for *msgstr ""*. Strings that have an empty translation basically don't have one at all. This is the only indicator a string has been added to Paperwork and requires translations.

**Strings and translations can be multi-lines.**

**Some strings can contain python format elements ("*%s*", "*%d*", "*{}*", etc). Their order *cannot* be changed ! If it is a problem for you, please open a ticket on GitHub. The string can be changed to use labels, which will allow you to reorder the elements.**

**Mixes of singular and plural are hard to handled. If one them is too tricky for you to handle, please open a ticket on GitHub. Workarounds can be found.**

### 4.3 Compiling the translations

**Translations must be compiled. It allows Paperwork (via gettext) to fetch them quickly at runtime.**

```
$ ./localize.sh gen-mo
```

**It will regenerated the .mo files in *locales/*.**

### 4.4 Testing the translations

**You can run Paperwork from your Git repository using the following command:**

```
python3 src/launcher.py
```

**Paperwork looks at the system settings to know which language must be used. If you want to try other ones, simply set the LANG environment variable.**

```
$ LANG=C python3 src/launcher.py  # Original strings (English)
$ LANG=fr_FR python3 src/launcher.py  # French
$ LANG=de_DE python3 src/launcher.py  # German
```

## 5 Adding a new language

**To add a new language to Paperwork, only the script *localize.sh* must be modified to include this new language. If you're unsure how to do it, simply open a ticket on GitHub and it will be done for you.**

## 6 Submit the new translations

### 6.1 Commit your translations

**You must first create a Git commit containing your translations. Beware that *localize.sh* always modifies all the .po and .mo files for all the languages, but you must commit *only* your own changes. The simplest way for that is to use the command '*git gui*'. It will clearly show you what you're adding to your commit.**

**Both .po and .mo files must be commited at the same time.**

**The commit message must indicate this is about updating the translations. Other than that, there are no specific rules.**

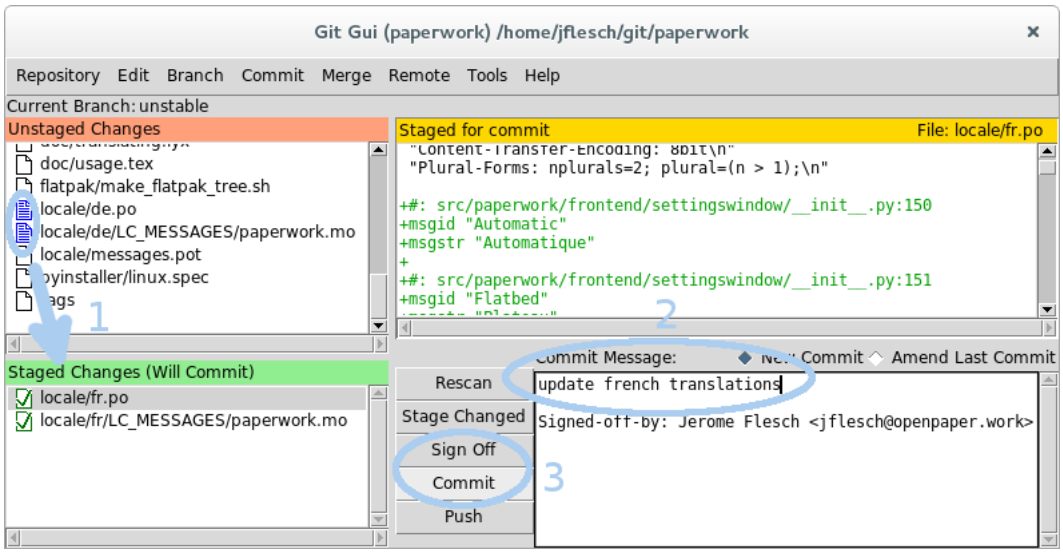



Fig. 3: git gui

## 6.2 Pushing

**You can then push your new commit to your repository on GitHub. Either click "*Push*" in *git gui*, or use the command "*git push*" in the shell.**

## 6.3 Pull request

**Go to GitHub and create a pull request for *jflesch/paperwork*.**