Projet Patacrep

Version 5.0.0

L'équipe Patacrep

févr. 19, 2018

Table des matières

Le projet *Patacrep* fournit des outils pour compiler un carnet de chants en LaTeX (en utilisant le paquet songs [1]).

Les chansons sont écrites, au choix, en utilisant le langage chordpro [2] (auquel cas elles seront converties en LaTeX lors de la compilation du carnet), ou en utilisant directement LaTeX.

[1] http://songs.sourceforge.net/
[2] http://www.chordpro.org/

CHAPITRE 1

Documentation

1.1 Installation

1.1.1 Dépendances

Patacrep est un projet reposant sur beaucoup de techniques, il a donc un certain nombre de dépendances à installer pour le faire fonctionner. Ces dépendances sont les mêmes pour tous les systèmes d'exploitation, mais la méthode d'installation diffère. Ces dépendances sont les suivantes :

— Python 3.3 ou plus récent. **Python 2 n'est pas supporté** ;
— LaTeX, et en particulier `lualatex` ;

Patacrep a aussi des dépendances optionnelles, qui peuvent ajouter des fonctionnalités, mais ne sont pas obligatoires :

— Lilypond, pour compiler des partitions.

GNU/Linux

Il n'existe pas (pour le moment ?) de paquet pour les différentes distributions. En attendant, `patacrep` peut être installé en utilisant pip (éventuellement dans un virtualenv [3], pour garder un environnement d'installation propre). Voir la *section suivante* pour les instructions.

Voici quelques informations supplémentaires pour certaines distributions.

Debian

Il n'existe pas (encore ?) de paquet debian pour *patacrep*. Les paquets à installer sous Debian (et ses dérivées comme Ubuntu) sont :

— Python 3.4
  — Installer les paquets nécessaires (pour une installation de *patacrep* sans *virtualenv*) :

```
apt-get install python3.4 python3-pip
```

  — Installer les paquets nécessaires (pour une installation de *patacrep* avec *virtualenv*) :

[3] http://virtualenv.readthedocs.org

```
apt-get install python3-virtualenv
```

— LaTeX :
  — Nécessaires : `texlive texlive-latex-base texlive-latex-recommended texlive-latex-extra`
  — Optionnel pour *patacrep*, nécessaire pour *patadata* : `texlive-lang-english texlive-lang-french texlive-lang-portuguese texlive-lang-spanish texlive-lang-italian texlive-fonts-extra`
— Lilypond (optionnel) : `lilypond`

Il est aussi possible, en utilisant stdeb [4], de créer un paquet `.deb` à la volée :

```
python setup.py --command-packages=stdeb.command bdist_deb
sudo dpkg -i deb_dist/python3-patacrep_5.0.0-1_all.deb
```

Ubuntu 16.04

En plus des paquets précédent, il est nécessaire pour que LaTeX fonctionne bien d'installer les deux paquets suivants :

```
apt-get install texlive-luatex  texlive-xetex
```

CentOS 6.5

Sur CentOS 6.5, un certain nombre de logiciels par défaut sont trop anciens pour faire fonctionner Patacrep correctement. En particulier, vous devriez utiliser une version récente de TeXLive, et installer ghostscript en version 9.

MacOSX

Vous devrez installer les dépendaces suivantes :

— Python 3 [5].
— LaTeX. La distribution MacTeX [6] est la plus simple à installer. Une installation personnalisée de TeXLive fonctionnera aussi si vous savez ce que vous faites.
— Lilypond peut être utile si vous souhaitez compiler les partitions dans les chansons. Ce n'est toutefois pas une dépendance obligatoire. Vous pouvez le télécharger à cette adresse [7]. Décompressez l'archive, puis placez-la dans `/Applications`. Vous devrez ajouter un lien vers lilypond pour que *songbook* puisse le trouver en lançant les commandes suivantes dans un Terminal :

```
sudo ln -s /Applications/LilyPond.app/Contents/Resources/bin/lilypond /usr/
↪local/bin/lilypond
echo "export PATH=$PATH:\usr\local\bin" >> ~/.bashrc
```

La première commande va vous demander votre mot de passe administrateur.

Windows

Voici où vous pourez trouver les dépendances nécessaires :

— Python 3 [8] ;
— MikTeX [9] pour avoir accès a LaTeX ;

http://github.com/astraw/stdeb
https://www.python.org/download/
https://tug.org/mactex/
http://www.lilypond.org/download.fr.html
https://www.python.org/download/
http://miktex.org/download

— Lilypond [10].

1.1.2 Installation depuis PyPi

Une fois les dépendances installées, vous pouvez vérifier que tout s'est bien passé en lançant les commandes

```
python --version
lualatex --version

# Uniquement si vous avez installé lilypond
lilypond --version
```

Si le numéro de version s'affiche, tout va bien, si vous avez une erreur `command not found`, il y a un problème.

Pour la suite, la bibliothèque *Patacrep* et le programme **`songbook`** sont disponibles sur le Python Packages Index [11], vous pouvez donc les installer avec `pip`.

Sans virtualenv

```
pip3 install patacrep
```

Puis pour tester si tout a bien fonctionné

```
songbook --version
```

Avec virtualenv

Pour éviter de mélanger les installations système (par *apt*, *yum* ou autre) et les installations de *pip*, il est recommandé d'installer l'application localement dans un *virtualenv* :

```
$ virtualenv -p python3 virtualenv
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in virtualenv/bin/python3
Also creating executable in virtualenv/bin/python
Installing setuptools, pip...done.
$ . virtualenv/bin/activate
(virtualenv)$ pip install patacrep
...
$ songbook --version
...
```

Si vous optez pour une installation dans un *virtualenv*, il faudra lancer la commande

```
. PATH_TO_VIRTUALENV/bin/activate
```

avant chaque utilisation de **`songbook`** ou de *patacrep*.

Mac OS X

Si vous avez des erreurs lors de l'installation, deux cas sont possibles :

— erreur `permission denied` : vous n'avez pas de droits en écriture dans `/usr/bin`. Ajoutez `sudo` devant la commande fautive, et relancez-la (`sudo pip3 install patacrep`). Un mot de passe administrateur vous sera demandé.

http://www.lilypond.org/windows.fr.html
http://pypi.python.org/pypi/patacrep

— `songbook: command not found`, c'est que `/usr/local/bin` n'est pas dans votre `PATH`. Pour l'ajouter, exécutez la commande

```
echo "export PATH=$PATH:/usr/local/bin" >> ~/.bashrc
```

1.1.3 Installation de la version de développement

Pour obtenir la version de développement, il est possible de télécharger les sources en utilisant git [12] :

```
git clone https://github.com/patacrep/patacrep.git
cd patacrep
pip3 install -r Requirements.txt
python3 setup.py install
```

Pour mettre à jour la version de développement, utilisez simplement

```
git pull
pip3 install -r Requirements.txt
python3 setup.py install
```

depuis le dossier `patacrep`

1.1.4 Mise à jour

Pour effectuer une mise à jour de *Patacrep*, vérifiez que les dépendances sont à jour, puis lancez la commande

```
pip3 install --upgrade patacrep
```

1.2 Création d'un carnet

Un carnet ou recueil de chant est décrit par un fichier `.yaml` qui peut être écrit manuellement. Les sections suivantes expliquent la *syntaxe* de ces fichiers, ainsi que les options de *mises en page* des carnets compilés. La manière de *compiler un recueil* est aussi détaillée.

1.2.1 Introduction

Un carnet est un ensemble de chansons qui peut peut être accompagné de divers éléments : index des chansons ou auteurs, page de titre, liste d'accords, préface, etc.

Un carnet est décrit par un fichier `.yaml` (détaillé *ci-après*). Il est généré par `songbook` en assemblant des fichiers de chansons `.csg` (ou `.tsg`), des templates `.tex`, des fichiers LaTeX `.tex`, des images, etc. Tout ce contenu provient de dossiers de donnée appelés *datadir*.

Un carnet fini, au format PDF, est l'intégration d'un contenu dans une mise en page particulière. Les options qui gouvernent cette mise en page sont principalement décrites dans les *templates*. Les différentes manières d'ajouter du contenu à un carnet sont décrites dans la section *Gestion du contenu*.

Exemple de fichier `.yaml`

Un exemple de fichier `.yaml` est fourni avec le code source [13] :

[12] http://git-scm.com
[13] https://github.com/patacrep/patacrep/blob/4b801c6b90a2c225668ead4b47442f344f94ab80/examples/example.yaml

```
content:
  - section: "Chants Traditionnels"
  - "chevaliers_de_la_table_ronde.csg"
  - "greensleeves.csg"
  - "vent_frais.csg"
  - section: "Exemples"
  - "exemple*.csg"

book:
  lang: en
  encoding: utf-8
  pictures: yes
  template: default.tex
  onesongperpage: no

chords: # Options relatives aux accords
  show: yes
  diagramreminder: important
  diagrampage: yes
  repeatchords: yes
  lilypond: no
  tablatures: no
  instrument: guitar
  notation: alphascale

authors: # Comment sont analysés les auteurs
  separators:
  - and
  ignore:
  - unknown
  after:
  - by

titles: # Comment sont analysés les titres
  prefix:
  - The
  - Le
  - La
  - "L'"
  - A
  - Au
  - Ces
  - De
  - Des
  - El
  - Les
  - Ma
  - Mon
  - Un
```

Syntaxe des fichiers `.yaml`

Un fichier `.yaml` est écrit au format YAML [14]. Le format YAML est un type de fichier texte dans lequel on représente des information de trois manières différentes :

- les listes d'éléments ordonnés : chaque élément commence sur une nouvelle ligne, précédé d'un tiret `-`. Tous les élements d'une même liste doivent avoir la même indentation (décalage depuis le bord gauche).
- les tableaux associatifs composés de multiples *clef : valeur*, les clefs étant la plupart du temps des chaînes, et les valeurs pouvant être n'importe quel type (une liste, un autre tableau associatif, une donnée scalaire).

[14] http://yaml.org/

— les données scalaires : chaînes de charactère (délimitées par des guillemets anglais `"..."`), nombre, booléen (`yes`/`no`).

Le fichier `.yaml` contient un tableau associatif, dont les clefs sont les noms d'options, et les valeurs associées sont les valeurs de ces options. Le type des valeurs dépend de l'option considérée. Les sections sur le *contenu des carnets* et sur la *mise en page* contiennent les détails concernant les différentes options.

Compiler un fichier `.yaml`

Le fichier `carnet.yaml` peut être compilé (*ie* transformé en fichier PDF) en ligne de commande, avec la commande suivante :

```
songbook chemin/vers/carnet.yaml
```

L'intégralité des options de la commande `songbook` sont disponible dans la *section dédiée*.

Ecrire ses propres fichiers `.yaml`

Le contenu d'un carnet est géré par le mot-clef `content` d'un fichier `.yaml`. Les différents types de contenus disponibles sont décrits dans la section *Gestion du contenu*.

Tous les autres mots-clefs des fichiers `.yaml` servent à faire la mise en page des carnets. Les options sont présentées dans la section *Mise en page du carnet*.

1.2.2 Gestion du contenu

Cette partie décrit comment choisir les chants à insérer dans un carnet.

Introduction

Le contenu d'un carnet est défini avec l'option `content` du fichier *.yaml*, sous la forme d'une liste. Le type de cette liste dépend du contenu à inclure. Voici un exemple de contenu.

```
content:
  - tex: "intro.tex"
  - section: "Chansons à boire"
  - "boire/*.csg"
  - section: "Chansons d'amour"
  - sort:
      key: ["by", "title"]
      content:
        - "amour/*.csg"
        - "love/*.csg"
```

Comme nous pouvons le voir, la valeur de `content` est une liste de tableaux associatifs ou de chaînes de caractères. Une chaîne de caractères est automatiquement transformé en tableau avec une clé `song`. Ainsi `"boire/*.csg"` et `song: "boire/*.csg"` sont équivalents. La *clé* d'un tableau associatif (avant le `:`) est une chaîne indiquant le *type de contenu* considéré : par exemple `section: "Chansons à boire"` va créer une section ayant pour titre *Chansons à boire*, tandis que

```
- sort:
    key: ["by", "title"]
    content:
      - "amour/*.csg"
      - "love/*.csg"
```

va inclure toutes les chansons des répertoires `amour/*.csg` et `love/*.csg`, triées par auteur (`by`), puis par titre (`title`).

Lorsqu'un tableau de type `content` n'a pas de contenu, cela va inclure toutes les chansons du répertoire `songs` :

```
content:
```

Ou alors pour inclure toutes ces chansons, triées pas auteur, album puis titre (c'est le tri par défaut) :

```
content:
  sort:
```

Types de contenus disponibles

Les types de contenus gérés par `patacrep` sont fournis par des extensions (ou plugins). Un certain nombre (décrits ci-après) sont proposés par défaut, et il est possible d'en écrire d'autres.

`song` [liste de chansons] Ce plugin, utilisé par défaut en l'absence de mot-clé, permet d'inclure une liste de chansons, triées par ordre alphabétique du nom de fichier. Il est suivi d'une ou plusieurs expressions rationnelles correspondant aux noms de fichiers à inclure. La syntaxe précise de ces expressions est décrite dans la documentation du module glob [15] ; le minimum à savoir est que `/` est utilisé pour parcourir les répertoires, `..` correspond au répertoire parent, et `*` à n'importe quelle chaîne de caractères.

```
content:
  song:
    - "premiere.csg"
    - "boire/*.tsg"
```

Est équivalent à (mot-clé `song` automatique) :

```
content:
  - "premiere.csg"
  - "boire/*.tsg"
```

Les fichiers sont recherchés successivement dans les datadirs associés à un carnet : `song` commence par chercher dans le repertoire `songs` du premier datadir et si au moins un fichier correspond à l'expression rationnelle, stoppe la recherche et passe à l'expression suivante. Sinon, il cherche dans le datadir suivant, et ainsi de suite jusqu'à la fin de la liste.

`sort` [liste triée de chansons] Ce plugin permet l'inclusion de chansons, triées selon un certain ordre. Il prend deux arguments (facultatifs) : `key` pour la liste des champs selon lesquels les chansons de l'argument `content` doivent être triées. Ces champs correspondent aux *keyvals* de l'environnement `song` (documentation [16]), à ceux ajoutés par `patacrep`, ainsi que ceux éventuellement ajoutés par le template courant, et à des champs spéciaux. cela donne finalement :
Paquet songs [17]

Nom	Description
by	auteurs
cr	informations de copyright
li	licence
sr	référence à la bible (le paquet *songs* a été écrit à l'origine pour des chants religieux)
index	une entrée supplémentaire dans l'index pour un vers
ititle	une entrée supplémentaire dans l'index pour un titre

Paquet *patacrep*

Nom	Description
album	Album
original	Titre original
cov	Chemin de l'image de couverture (relative ou non au répertoire du fichier `.csg`)
url	URL de la chanson

https://docs.python.org/3.4/library/glob.html
http://songs.sourceforge.net/songsdoc/songs.html#sec5.1
http://songs.sourceforge.net

Valeurs spéciales

Nom	Description
title	Titres
path	Chemin du fichier

L'ordre de tri par défaut est : auteurs, album, titre.

Il faut remarque la liste de contenu de `sort` n'est pas nécessairement une liste d'expression rationnelle : c'est n'importe quel élément de contenu qui renvoie une liste de chansons. Ainsi (en utilisant le plugin `cwd` décrit ci-après) le `content` suivant est parfaitement valide.

```
content:
  sort:
    content:
      - cwd:
        path: repertoire
        content: "*.csg"
```

Une conséquence de cela est que ne pas donner de `content` à `sort` permet d'inclure toutes les chansons du répertoire `songs`, récursivement.

`cd`[18] [changement de répertoire] Lorsque plusieurs chansons du même répertoire sont incluses, il peut être fastidieux de redonner le chemin complet à chaque fois. Ce plugin permet d'éviter ce travail. Les deux valeurs suivantes de la variable `content` sont équivalentes :

```
content:
  - cd:
    path: des/repertoires/vers
    content:
      - "chanson1.csg"
      - "chanson2.csg"
      - "chanson3.csg"
```

```
content:
  - "des/repertoires/vers/chanson1.csg"
  - "des/repertoires/vers/chanson2.csg"
  - "des/repertoires/vers/chanson3.csg"
```

Cette commande recherche en priorité des chants dans le sous-dossier `path` relatif au dossier du fichier *.yaml*. Si aucun contenu n'est trouvé, alors les chants sont recherché dans le sous-dossier `path` relatif au répertoire `songs` des *datadir* (dans lequel sont cherchées les chansons par défaut).

Enfin, il faut remarquer que, tout comme le plugin `sort`, la liste de contenu de `cd`[19] n'est pas limitée à une liste d'expressions rationnelles correspondant à des chansons : elle peut être n'importe quel contenu correspondant à une liste de chansons. De plus, la commande `cd` utlisée sans préciser de `path` permet d'inclure toutes les chansons (récursivement) comprises dans le répertoire `path`.

```
content:
  - "chants_a_boire/*.csg"
```

Ne va inclure que les chants situés directement dans le dossier `chants_a_boire`, alors que la command `cd` suivante va aussi inclure les chants des sous-dossiers `chants_a_boire/de_l_eau/`, `chants_a_boire/du_vin/`...

```
content:
  cwd:
      path: chants_a_boire
```

`section` [sections LaTeX] Ce plugin permet d'inclure des sections (et chapitres, paragraphes, etc.). Les mots-clefs sont `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`, ainsi que leurs versions étoilées (qui ne seront pas numérotées).

https://docs.python.org/2/library/cd.html#module-cd
https://docs.python.org/2/library/cd.html#module-cd

Ces mots-clefs ont pour contenu soit une chaîne de caractères (le titre), soit une valeur `name` et une valeur `short` pour les versions non étoilées (le titre, et le titre court, optionnel, pour la table des matières).

```
content:
  chapter: "Chansons d'amour"
  chapter:
    name: "Chansons à boire"
    short: "prosit"
```

`songsection` [sections du paquet songs[20]] Ce plugin introduit deux mots-clefs `songchapter` et `songsection`, qui correspondent aux sections et chapitres définis par le paquet *songs*. Le style de ces sections est plus cohérent avec l'apparence des chansons, mais elles ne sont pas numérotées, et il n'y a pas de version étoilée.

Exemple : `songchapter: "Chansons d'amour"`

`tex` [inclusion de fichiers LaTeX quelconques] Il est possible avec ce plugin d'inclure un fichier LaTeX quelconque. L'unique mot-clef `tex` prend en argument le ou les fichiers latex à inclure dans le carnet.

Exemple : `tex: "intro.tex"`

`include` : inclusion d'un autre fichier de contenu. Ce plugin permet d'inclure un autre fichier au format `YAML` dans le contenu du carnet courant. Ainsi, mettre `include: "my_content.sbc"` ira chercher le fichier `my_content.sbc` et placera le contenu de ce fichier dans le carnet principal. `my_content.sbc` doit contenir une liste d'éléments de contenu valide. Par exemple, on peut imaginer qu'il contienne ceci :

```
- section: "Chansons à boire"
- "boire/*.csg"
- section: "Chansons d'amour"
- include: "amour.sbc"
```

En particulier, il peut donc contenir un ou plusieurs autres mot-clef `"include"`, et il est possible de les trier à postériori :

```
content:
  - sort:
      content:
        include: "amour.sbc"
```

Un cas d'utilisation typique est l'inclusion du même contenu dans plusieurs carnets différents. L'extention `.sbc` est arbitraire, et a été choisie commme abréviation de « SongBook Content » (contenu de carnet de chants), cependant il est tout à fait autorisé d'utiliser d'autres extensions : `include: "fichier.yaml"`.

Les fichiers inclus de cette manière sont recherchés dans tous les *datadir* plus le dossier dans lequel se trouve le fichier dans lequel `"include"` a été rencontré.

`addsongdir` [ajout d'un dossier à la liste des dossiers à parcourir pour trouver des chansons.] Le chemin (`path`) est relatif au chemin du fichier `.yaml`.

```
content:
  - addsongdir:
      path: "../sous_dossier/"
      content:
```

`setcounter` [spécifie le numéro de la prochaine chanson.] Utile si l'on souhaite qu'une nouvelle section recommence la numérotation à 1 ou, au contraire, sauter des numéros pour que la section change de centaine par exemple.

Usage : `setcounter: 101`

[20] http://songs.sourceforge.net

Il est possible d'ajouter son propre type de contenu (images, fichiers abc, ...) à un carnet en écrivant son propre plugin. La procédure est décrite dans la section *Ajouter du contenu aux carnet : écrire un plugin*.

Les fichiers utilisés par `patacrep` pour créer des carnets de chant sont rassemblés dans des dossier qui suivent une organisation particulière. Cette organisation est décrite dans la section *Organisation des datadirs*.

1.2.3 Mise en page du carnet

La mise en page des carnets est gérée par un système d'options : il est possible de spécifier la taille et l'orientation du papier, le type de police des accords, *etc.* en mettant le bon mot clef dans un fichier `.yaml`.

La plupart des options peuvent être omises : elle prendront alors une valeur par défaut, documentée ci-dessous.

Exemple de fichier `.yaml`

Un fichier `.yaml` contient un dictionnaire YAML, dont les clefs sont les noms des options et leur valeur associée pour la mise en page. Le mot-clef `content` ne gère pas la mise en page, mais le *contenu du carnet*.

Voici par exemple le fichier `.yaml` qui fournit les options par défaut :

```
book: # Options générales
  lang: en
  encoding: utf-8
  pictures: yes
  template: default.tex
  onesongperpage: no

chords: # Options musicales
  show: yes
  diagramreminder: important
  diagrampage: all
  repeatchords: yes
  lilypond: no
  tablatures: no
  instrument: guitar
  notation: alphascale

authors: # Analyse des auteurs
  separators:
  - and
  ignore:
  - unknown
  after:
  - by

titles: # Analyse des titres
  prefix:
  - The
  - Le
  - La
  - "L'"
  - A
  - Au
  - Ces
  - De
  - Des
  - El
  - Les
  - Ma
  - Mon
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - Un
content:
```

Liste des options

Les valeurs par défaut sont données ici en supposant que **la langue principale du carnet est le français**.

Options générales

Ce sont des sous-clés de la clé `book`.

lang

Langue du carnet (Code ISO 639-1 à 2 lettres).
- Défaut : `en`
- Type : Chaîne de charactères
- Valeurs : `fr` et `en` sont actuellement supportés

onesongperpage

Commencer toutes les chansons sur une nouvelle page.
- Défaut : `no`
- Type : Booléen

pictures

Afficher les couvertures des albums.
- Défaut : `yes`
- Type : Booléen

template

Template de carnet à utiliser.
- Défaut : `patacrep.tex`
- Type : Chaîne de charactères
- Valeurs : Voir le dossier `templates` des datadirs pour les autres fichiers disponibles

encoding

Encodage des fichiers à lire (chansons, templates, etc.). Peut aider à résoudre des problèmes d'accentuation.
- Défaut : `utf-8`
- Type : Chaîne de charactères

Options musicales

Ce sont des sous-clés de la clé `chords`.

show

Afficher les accords au sein des paroles.
- Défaut : `yes`
- Type : Booléen

diagramreminder

Rappeler en début de chansons certains diagrammes d'accords.
- Défaut : `important`
- Type : Chaîne de charactères
- Valeurs :
  - `all` : Rappel de tous les accords présents dans le chant
  - `important` : Rappel des accords peu communs du chant
  - `none` : Aucun rappel d'accords

diagrampage

Insérer une page d'accords en début de carnet.
- Défaut : `all`
- Type : Chaîne de charactères
- Valeurs :
  - `all` : Inclusion de tous les accords (communs et peu communs)
  - `important` : Inclusion des accords peu communs
  - `none` : Aucune page d'accords en début de carnet

repeatchords

Afficher les accords dans tous les couplets (disponible uniquement pour certains chants).
- Défaut : `yes`
- Type : Booléen

lilypond

Inclure les partitions musicales (nécéssite le logiciel libre lilypond).
- Défaut : `no`
- Type : Booléen

tablatures

Inclure les tablatures.
- Défaut : `no`
- Type : Booléen

instrument

Instrument pour lequel il faut rappeler les accords.
- Défaut : `guitar`
- Type : Chaîne de charactères
- Valeurs :
  - `guitar` : Guitare
  - `ukulele` : Ukulélé

notation

Notation des accords.
- Défaut : `solfedge`
- Type : Chaîne de charactères, ou liste de sept chaînes de caractères.
- Valeurs :
  - `alphascale` : Système internanial ABCDEFG
  - `solfedge` : Système français Do Ré Mi
  - liste : Liste des noms des notes, en commençant par La. Ainsi, par exemple, `solfedge` est équivalent à `['La', 'Si', 'Do', 'Ré', 'Mi', 'Fa', 'Sol']`.

Analyse des auteurs

Ce sont des sous-clés de la clé `authors`.

separators

Mots qui séparent les noms d'artistes. Par exemple, si cette option contient `et`, une chanson ayant comme artiste `Georges Brassens et Charles Trenet` apparaitra dans l'index à la fois à `Brassens` et `Trenet`.
- Défaut : `- and`
- Type : Tableau de mots

ignore

Noms d'artistes à ignorer. Permet par exemple de spécifier que l'auteur d'une chanson est inconnu, sans pour autant avoir `Anonyme` apparaitre dans l'index.
- Défaut : `- unknown`
- Type : Tableau de mots

after

Mots introduisant les noms des auteurs. Par exemple, si cette option contient `de`, une chanson ayant comme artiste `Musique de Jean Boyer, chantée par Georges Brassens` apparaitra dans l'index à la fois à `Jean Boyer` et `Georges Brassens`.
- Défaut : `- by`
- Type : Tableau de mots

Analyse des titres

Ce sont des sous-clés de la clé `titles`.

prefix

Préfixe à ignorer lors du tri des titres (notamment dans l'index).
- Défaut :

```
- The
- Le
- La
- "L'"
- A
- Au
```

(suite sur la page suivante)

(suite de la page précédente)

```
- Ces
- De
- Des
- El
- Les
- Ma
- Mon
- Un
```

— Type : Tableau de mots

Options des templates

Certains options sont propres aux templates utilisés. Par exemple le template `patacrep.tex` (qui inclut notamment `default.tex`) permet de personnaliser certaines couleurs et la page de garde.

Les options des templates sont regroupées sous la clé `template` avec comme sous-clé le nom du fichier de template :

```
# options précédentes `book`, `chords`...

template:
  default.tex:
    title: "Recueil de chansons pour guitare"
    author: "L'équipe Patacrep"
  patacrep.tex:
    color:
      songlink: FF0000
      hyperlink: 0000FF
    bgcolor:
      note: D1E4AE
      songnumber: AED1E4
      index: E4AED1
```

Template `default.tex`

title

Titre du carnet de chants.
— Défaut : `"Recueil de chansons pour guitare"`
— Type : Chaîne de charactères

author

Auteur du carnet de chants.
— Défaut : `"L'équipe Patacrep"`
— Type : Chaîne de charactères

classoptions

Options de la classe LaTeX.
— Défaut : *(vide)*
— Type : Tableau de mots

Template `patacrep.tex`

subtitle

Sous-titre du carnet (pour la page de garde).
- Défaut : *(vide)*
- Type : Chaîne de charactères

version

Version du carnet (pour la page de garde).
- Défaut : *(vide)*
- Type : Chaîne de charactères

url

Site web de l'auteur (pour la page de garde).
- Défaut : `"http://www.patacrep.com"`
- Type : Chaîne de charactères

email

Courriel de l'auteur (pour la page de garde).
- Défaut : `"crep@team-on-fire.com"`
- Type : Chaîne de charactères

picture

Image pour la page de garde.
- Défaut : `"img/treble_a"`
- Type : Chaîne de charactères

picturecopyright

Copyright pour l'image de la page de garde.
- Défaut : `"Dbolton \\url{http://commons.wikimedia.org/wiki/User:Dbolton}"`
- Type : Chaîne de charactères

footer

Pied de page de la page de garde.
- Défaut : `"Créé avec le programme Songbook (\\url{http://www.patacrep.com})"`
- Type : Chaîne de charactères

color : songlink

Couleur des liens vers les chants.
- Défaut : `4e9a06`
- Type : Couleur en hexadécimal

color : hyperlink

Couleurs des liens hypertextes.
- Défaut : `204a87`
- Type : Couleur en hexadécimal

bgcolor : songnumber

Couleur de fond des numéros de chants.
- Défaut : `D1E4AE`
- Type : Couleur en hexadécimal

bgcolor : note

Couleur de fond des indications.
- Défaut : `D1E4AE`
- Type : Couleur en hexadécimal

bgcolor : index

Couleur de fond des lettres de l'index.
- Défaut : `D1E4AE`
- Type : Couleur en hexadécimal

1.3 Écriture des chansons

Les chansons peuvent être écrites dans deux formats différents : en Chordpro ou en LaTeX. Le format Chordpro est plus simple et pleinement pris en charge par patacrep, mais ses fonctionnalités sont un peu plus limitée ; LaTeX, en revanche, est plus compliqué, mais bien plus puissant, et seul un sous-ensemble de ce langage est pris en charge. Pour la majeure partie des cas, Chordpro devrait être suffisant.

1.3.1 Chordpro

Le format ChordPro consiste à insérer les accords entre crochets au milieu des paroles, juste avant l'endroit où ils doivent être joués. Lors de la compilation, les accords sont disposés au dessus du texte qui suit immédiatement. D'autres informations peuvent être spécifiées entre accolades (le titre, l'artiste, le début du refrain. . .).

Exemple

```
{lang : en}
{columns :  2}
{title : Greensleeves}
{subtitle: Example of the chordpro format}
{artist: Traditionnel}
{cover : traditionnel.jpg}
{album : Angleterre}

{partition : greensleeves.ly}

A[Am]las, my love, ye [G]do me wrong
To [Am]cast me oft dis[E]curteously
```

(suite sur la page suivante)

(suite de la page précédente)

```
And [Am]I have loved [G]you so long
De[Am]lighting [E]in your [Am]companie

{start_of_chorus}
   [C]Greensleeves was [G]all my joy
   [Am]Greensleeves was [E]my delight
   [C]Greensleeves was my [G]heart of gold
   And [Am]who but [E]Ladie [Am]Greensleeves
{end_of_chorus}

I [Am]have been ready [G]at your hand
To [Am]grant what ever [E]you would crave
I [Am]have both waged [G]life and land
Your [Am]love and [E]good will [Am]for to have

I [Am]bought thee kerchers [G]to thy head
That [Am]were wrought fine and [E]gallantly
I [Am]kept thee both at [G]boord and bed
Which [Am]cost my [E]purse well [Am]favouredly

I [Am]bought thee peticotes [G]of the best
The [Am]cloth so fine as [E]fine might be
I [Am]gave thee jewels [G]for thy chest
And [Am]all this [E]cost I [Am]spent on thee

{comment:test of comment}

{guitar_comment: test of guitar comment}

{image: traditionnel.jpg}

Thy [Am]smock of silke, both [G]faire and white
With [Am]gold embrodered [E]gorgeously
Thy [Am]peticote of [G]sendall right
And [Am]this I [E]bought thee [Am]gladly
```

Directives

TODO

Tutoriels externes : Qu'est ce qu'une tablature au format Chordpro ? (par oukelelatab) [21]

Outils externes

Il existe différents outils en ligne pour convertir des textes bruts au format chordpro :

— Song Formater (par UkeGeeks) [22]

1.3.2 LaTeX

Avertissement : Afin de détecter les méta-informations de la chanson (titre, auteur, etc.), une analyse syntaxique du préambule du fichier (les arguments de `\beginsong`) est effectuée. Mais puisque LaTeX est Turing-complet, une vraie analyse syntaxique est très complexe, et dépasse de loin le cadre de ce projet. Patacrep n'est donc capable de lire correctement qu'un sous-ensemble du LaTeX. Il n'y a pas de spécification

[21] http://oukelelatab.free.fr/index.php?static3/le-format-chordpro
[22] http://ukegeeks.com/tools/songformater

de ce qui est analysé et ce qui ne l'est pas, mais pour faire simple, si tous les arguments sont entourés d'accolades (`by={Boris Vian}`) et ne contient aucune commande LaTeX (mis à part les accents), cela devrait fonctionner.

Une chanson est un fichier texte `chanson.sg` placé dans le répertoire `songs` (ou un de ses sous-répertoires) d'un datadir. C'est un fichier LaTeX interprété avec le paquet songs [23]. Si la connaissance de LaTeX peut-être nécessaire pour faire des choses compliquées, cette documentation devrait permettre d'écrire des chansons sans connaissance préalable de LaTeX. Pour un langage encore plus simple (moins puisant que LaTeX, mais qui devrait suffire dans la plupart des cas), il est aussi possible d'utiliser *Chordpro*.

Ce fichier commence par la ligne `\beginsong{titres}` (éventuellement précédée des commandes définissant la *langue* de la chanson et le *nombre de colonnes*), termine par la ligne `\endsong`, et contient entre ces deux lignes les commandes nécessaires pour renseigner *les titres*, écrire les *couplets et refrains*, placer des *accords*, ou insérer *d'autres éléments*.

Exemple

Voici un exemple simplifié de la chanson Sad robot [24] par *Pornophonique*.

```
\selectlanguage{english}
\songcolumns{2}
\beginsong{Sad robot}
  [by=Pornophonique,cov=8-bit-lagerfeuer,album=8-bit lagerfeuer]

  \cover
  \gtab{Dm}{XX0231}
  \gtab{F}{1:022100}
  \gtab{C}{X32010}
  \utab{Dm}{2210}
  \utab{F}{2010}
  \utab{C}{0003}

  \lilypond{Sad_robot}

  \beginverse
    His \[Dm]steely skin is covered
    By \[F]centuries of dust
    \[C]Once he was a great one
    \[Dm]Now he's dull and rust
  \endverse

  \begin{repeatedchords}
    \beginverse*
      An \[Dm]oily tear he's crying
      \[F]Can you feel the pain
      Of the \[C]sad, sad robot
      And it's \[Dm]driving him insane
    \endverse*

    \beginverse*
      He can't \[Dm]turn back time nor history
      So his \[F]life became a misery
      He \[C]has to face the destiny
      Nobody \[Dm]cares anymore
    \endverse*

    \beginchorus
```

(suite sur la page suivante)

http://songs.sourceforge.net
http://www.jamendo.com/fr/track/81740

(suite de la page précédente)

```
      \[Dm]Sad, sad robot
      \[F]Sad, sad robot
      \[C]Sad, sad robot
      All a\[Dm]lone
    \endchorus
  \end{repeatedchords}
\endsong
```

Langue

Définir la langue de la chanson permet de respecter la typographie. Par exemple, en français, un point d'exclamation est précédé d'une espace, alors qu'en anglais, il est collé au mot qui le précède.

La définition de la langue se fait avant la commande `\beginsong{Titre}`, en utilisant :

```
\selectlanguage{english}
\beginsong{Titre}
```

Les langues disponibles sont celles reconnues par le paquet LaTeX Babel [25].

Titres et méta-informations

Les titres (le titre principal, et des titres alternatifs éventuels), ainsi que le nom de l'auteur, l'album, etc., sont définis avec la commande `beginsong{titres}[informations]`. Par exemple :

```
\beginsong{Sad Robot}
  [by={Pornophonique}, cov={8-bit-lagerfeuer}, album={8-bit lagerfeuer}]
```

Les différents titres sont séparés par des doubles barre obliques : `beginsong{titre1 \\ titre2 \\ titre3}`.

La liste des paramètres disponibles (certains paramètres de *songs*, et d'autres ajoutés par *patacrep*), est donnée ici. Pour une utilisation avancée, il est possible de définir de nouveaux paramètres [26].

— Paquet songs [27]

Nom	Description
by	auteurs
cr	informations de copyright
li	licence
sr	référence à la bible (le paquet *songs* a été écrit à l'origine pour des chants religieux)
index	une entrée supplémentaire dans l'index pour un vers
ititle	une entrée supplémentaire dans l'index pour un titre

— Paquet *patacrep*

Nom	Description
album	Album
original	Titre original
cov	Chemin de l'image de couverture (relative ou non au répertoire du fichier `.sg`)
url	URL de la chanson

Index

Les titres et auteurs des chansons reçoivent un traitement particulier avant d'être intégrés dans l'index.

http://www.ctan.org/pkg/babel
http://songs.sourceforge.net/songsdoc/songs.html#sec11.8
http://songs.sourceforge.net

Titres Les articles courants (*Les*, *Le*, *The*, *A*, etc.) en début de titre peuvent être supprimés. Cela permet de trier les titres selon le premier mot « important ».

L'option `titleprefixwords` permet de définir la liste des articles à ne pas considérer en début des titres.

Par défaut, les titres sont affichés dans la table des matières en rejetant l'article entre parenthèses (par exemple *Raven (The)*). Il est possible de modifier cela en redéfinissant la commande LaTeX `\indextitle{article}{titre}`. Par exemple, pour afficher le titre sans modifications (*The Raven*), on pourra mettre dans le préambule du fichier LaTeX (ou plutôt dans un *template*) :

```
\renewcommand{\indextitle}[2]{#1 #2}
```

Noter que cette commande prend toujours deux arguments, le premier pouvant être vide (par exemple `\indextitle{}{Enivrez-vous}`). Il faut donc être vigilant à ce que le résultat de la commande prenne en compte ce cas-là.

Auteurs Le traitement des auteurs est fait de telle manière à ce que, par exemple, une chanson ayant pour auteur `Composée par Jean Boyer (1945), chantée par Georges Brassens` apparaisse dans l'index des auteurs à *Boyer* et *Brassens*.

Les options `authwords` permettent de paramétrer ce traitement. Cette option est un dictionnaire ayant trois clefs `sep`, `ignore` et `after`. Le traitement est le suivant (en prenant pour exemple *Paroles de William Blake (Milton, 1808), musique de Hubert Parry (1916), chanté par Emerson,~Lake~and~Palmer*, avec `authwords` valant *« sep »: [« and », « et »], « ignore »: [« anonyme »], « after »: [« de », « par »]*).

1. Les parenthèses (et leur contenu) sont supprimées.

   ```
   Paroles de William Blake, musique de Hubert Parry, chanté par␣
   ↪Emerson,~Lake~and~Palmer
   ```

2. La chaîne est découpée suivant les séparateurs de `authwords['sep']` (c'est-à-dire `and` et `et` dans le cas présent), ainsi que la virgule.

   ```
   Paroles de William Blake
   musique de Hubert Parry
   chanté par Emerson,~Lake~and~Palmer
   ```

   On remarque que `Emerson,~Lake~and~Palmer` n'a pas été découpé selon le `and`, car ce séparateur n'est pas entouré d'espaces mais d'espaces insécables `~`.

3. Tout ce qui précède des éléments de `authwords['after']` (`par` et `de` dans notre exemple) est supprimé.

   ```
   William Blake
   Hubert Parry
   Emerson,~Lake~and~Palmer
   ```

4. Les auteurs correspondant à des auteurs de la liste `authwords['ignore']` sont supprimés (aucun dans notre exemple).

   ```
   William Blake
   Hubert Parry
   Emerson,~Lake~and~Palmer
   ```

5. Les auteurs sont découpés entre le prénom (ou l'article pour les groupes) et le nom de famille. Le découpage est fait à la dernière espace.

   ```
   William / Blake
   Hubert / Parry
   / Emerson,~Lake~and~Palmer
   ```

6. Les auteurs sont passés à la commande `\indexauthor{prénom}{nom}`, qui va se charger d'afficher correctement les noms (voir paragraphe suivant).

Par défaut, les auteurs sont affichés dans l'index avec le prénom rejeté après le nom, avec une virgule (par exemple *Poe, Edgar Allan*). Il est possible de modifier cela en redéfinissant la commande LaTeX `\indexauthor{prénom}{nom}`. Par exemple, pour afficher le prénom entre parenthèse en début de nom (*(Edgar Allan) Poe*), on pourra mettre dans le préambule du fichier :

```
\renewcommand{\indexauthor}[2]{(#1) #2}
```

Cette commande prend toujours deux arguments, le premier pouvant être vide (par example `\indexauthor{}{Simon and Garfunkel}`). Il faut être donc vigilant à ce que sa définition prenne en compte ce cas spécial.

Couplets, refrains

La chanson se compose d'une succession de couplets (*verse*) et de refrains (*chorus*). Un couplet figure dans un environnement `verse`, c'est-à-dire qu'il commence par `\beginverse` et se termine par `\endverse`. De la même manière, un refrain est placé dans un environnement `chorus`, c'est-à-dire entre les balises `\beginchorus` et `\endchorus`. Les paroles sont écrites normalement entre les balises d'ouverture et de fermeture de l'environnement.

Note : Contrairement à ce qui est habituel en LaTeX, les retours à la ligne sont respectés. Il n'est donc pas nécessaire de sauter une ligne ou d'utiliser une commande `\\` ou `\par` à chaque fin de vers.

Numérotation

La numérotation se fait automatiquement pour chaque `\beginverse` rencontré. Cependant, il est parfois plus lisible de scinder un couplet en deux parties, la deuxième partie ne devant pas être numérotée. Pour cela, nous utilisons la commande `\beginverse*` ; il faut alors fermer l'environnement par `\endverse*`. Par exemple, un couplet en huit vers se décompose souvent en deux strophes de quatre vers comme dans l'exemple suivant.

```
\beginverse
  His \[Dm]steely skin is covered
  By \[F]centuries of dust
  \[C]Once he was a great one
  \[Dm]Now he's dull and rust
\endverse

\beginverse*
  An oily tear he's crying
  Can you feel the pain
  Of the sad, sad robot
  And it's driving him insane
\endverse*
```

Accords

Pour préciser sur quelle syllabe un accord doit être joué, on utilise une commande spéciale. Par exemple, la commande `\[E]` produira un `Mi` au dessus de la syllabe suivante dans le PDF.

Il est impératif d'utiliser la convention anglo-saxone de notation des accords (A, B, C, D, E, F, G) et non pas la notation latine (La, Si, Do, Ré, Mi, Fa, Sol). En revanche, suivant la langue utilisée pour le recueil, le rendu des accords dans le PDF pourra être différent (l'accord `\[D]` sera affiché `Ré` si la langue du songbook est `french`). Ce rendu est paramétrable avec *l'option* `notenamesout`.

Par défaut, l'accord est majeur (C fait référence à l'accord de Do majeur). Les accords mineurs sont précisés par un `m` minuscule. Le symbole bémol ♭ est représenté en utilisant le caractère `&`. Le dièse ♯ est codé par le caractère

#. Les autres notations sont simplement ajoutées comme des caractères à l'accord principal. Par exemple, l'accord de `La bémol mineur` est noté `[A&m]`.

Note : Pour des raisons techniques, le symbole # (tout comme `&`) ne peut pas être utilisé dans les environnements `nolyrics`. Dans ce cas là, il faut utiliser `shrp` (respectivement `flt`).

Répétition

De façon à avoir un document lisible et relativement compact, les accords des couplets et des refrains ne sont renseignés qu'une seule fois à leur première occurrence. En effet, même si jouer les morceaux du premier couplet en chantant les paroles du second peut demander un peu de gymnastique, cela fera travailler votre mémoire tout en offrant un texte bien moins surchargé et (beaucoup) moins de pages à imprimer.

Si toutefois vous souhaitez que les accords soient répétés dans toute la chanson, vous pouvez utiliser l'option `repeatchords` du template de votre recueil (voir la section *Création d'un carnet*). Il faut évidemment pour cela que les accords soient renseignés dans tous les couplets des chansons.

Chœurs et répétitions

Lorsqu'une phrase ou un couplet est répété plusieurs fois d'affilée, il est conseillé d'utiliser la commande `\rep` plutôt que d'écrire `\bis` ou d'indiquer directement (x4). Par exemple, si le mot `Hallelujah` est répété quatre fois, nous écrirons~ :

```
Hallelujah \rep{4}
```

La commande `echo` fait référence à des chœurs (ou similaire).

```
Hallelujah \echo{Hallelujah}
```

Caractères spéciaux

Quelques caractères doivent être écrits différemment en utilisant des commandes LaTeX pour un obtenir un meilleur rendu typographique dans le PDF. Les deux exemples principaux sont les trois points de suspension (…) et le caractère *œ*. Pour représenter ces caractères, vous devez utiliser respectivement les commandes `\dots{}` et `\oe{}` (ou utiliser les caractère UTF-8 `…` et œ). On utilise des accolades autour des commandes de sorte que les commandes puissent être insérées où vous le désirez sans interférer avec le reste du texte.

Inclure d'autres éléments

Partitions

Si vous souhaitez ajouter une ligne mélodique dans une chanson, vous pouvez utiliser Lilypond [28] pour générer la partition. Créez pour cela un nouveau fichier `partition.ly` dans le même répertoire que la chanson. Il faut inclure le fichier d'en-tête `header` et définir l'option `paper-height` de façon à ce que la partition produite tienne sur une page avec le moins de blanc possible. Une première estimation est de compter 1.6 cm pour une ligne. Puis, écrivez votre partition entre accolades, comme dans l'exemple suivant.

```
\begin{lilypond}
\include "header"
\paper{paper-height = 3.3\cm}
{
```

(suite sur la page suivante)

[28] http://www.lilypond.org/

(suite de la page précédente)

```
  \key c \major
  \time 2/4
  \relative c''
    {
      e4 c g'2 a4 a8. a16 g8 e4 c8
      a'4 a8. a16 g8 f e c d2~ d4
      e8 f g4 g8. g16 f8 e d c a c4 a8 g4
      c8 d e8 g4 g,8 e' e d d c2
    }
}
\end{lilypond}
```

Enfin, pour insérer votre partition `partition.ly` dans une chanson, utilisez la commande `lilypond` dans le fichier `sg` adéquat :

```
\lilypond{partition}
```

Diagrammes des accords

Étant donné qu'un accord de guitare ou de ukulélé peut se jouer de plusieurs façons différentes et qu'il est parfois judicieux de privilégier telle ou telle position, *patacrep* permet de représenter schématiquement ces accords en début de chanson sous forme de diagramme. Pour cela, nous utilisons les commandes `\gtab` (guitare) et `\utab` (ukulélé) juste avant le premier couplet ou refrain. Dans le cas où ces accords ne sont pas standards, ils peuvent être marqués comme importants avec les commandes `\gtab*` et `\utab*`. Voici quelques exemples classiques~ :

```
\gtab{C}{3:002220}
\gtab*{Amaj7}{5:X0221X}
\utab{C}{0003}
\utab{B&m}{1:2000}
```

- Les six chiffres correspondent aux six cordes de la guitare (`Mi`, `La`, `Ré`, `Sol`, `Si`, `Mi`).
- La valeur du chiffre indique la frette sur laquelle on appuie.
- Un 0 désigne une corde jouée à vide.
- Un X indique que la corde ne doit pas être jouée.
- Une valeur avant un « : » désigne un barré (« `3:` » indique un barré à la 3 e frette).

Note :

- X est la lettre majuscule x. Un x minuscule produira une erreur lors de la compilation.
- 0 est le chiffre zéro et non pas la lettre majuscule o.

Intersongs

Le paquet songs [29] permet d'insérer du texte quelconque entre deux chansons (une autre méthode consiste à utiliser le plugin *tex*).

Un *intersong* est un fichier ayant pour extension `.is`, et contenant un environnement `intersong`, c'est-à-dire commençant par la ligne `\begin{intersong}` et finissant par `\end{intersong}`.

Tout a été fait pour qu'ils puissent être manipulés comme des chansons, c'est-à-dire :

- il est possible de spécifier la *langue* et le nombre de *colonnes* comme pour les chansons (en plaçant les commandes correspondantes en tout début de fichier, avant même `\begin{intersong}`) ;
- il est possible de trier les intersong avec les chansons, en utilisant la commande `\sortassong{Titre}[Paramêtres]`. Cette commande ne produit rien dans le document final, mais indique au plugin *sort* comment trier les chansons.

[29] http://songs.sourceforge.net

Par exemple, pour introduire une biographie de Georges Brassens avant l'ensemble de ses chansons (dans un recueil trié par auteur, album puis titre), on pourra avoir le fichier `brassens.is` suivant :

```
\selectlanguage{french}
\begin{intersong}
\sortassong{}[by={Georges Brassens}]

Georges Brassens était un chic type né en 1921.
\end{intersong}
```

Colonnes

La commande `songcolumns` détermine le nombre de colonnes sur lequel sera présentée la chanson. Elle s'utilise juste avant la commande `beginsong`. Généralement une chanson se présente sur 1, 2 ou 3 colonnes. Par convention, utilisez deux colonnes par défaut.

```
\songcolumns{2}
\beginsong{Titre}
```

1.4 Programme `songbook`

Le programme **`songbook`** est une interface minimale à la bibliothèque `patacrep` : il permet de compiler un carnet de chants à partir du fichier `.yaml` (pour des programmes qui permettent de manipuler ce fichier `.yaml`, ou les chansons, voir *patagui* ou *patanet*).

1.4.1 Utilisation

```
songbook [-h] [--version] [--datadir DATADIR [DATADIR ...]] [--verbose] [--steps␣
↪STEPS] [--cache CACHE] [--error {failonsong,failonbook,fix}] book.yaml
```

L'argument obligatoire est le nom d'un fichier `.yaml`, qui est compilé en un carnet de chants. Les options optionnelles sont les suivantes.

`-h, --help`
: Affiche une aide et termine le programme.

`--version`
: Affiche la version du programme.

`--datadir` `<DATADIR> [<DATADIR> ...]`, **`-d`** `<DATADIR> [<DATADIR> ...]`
: Répertoire contenant les données, contenant (mais ce n'est pas obligatoire) les sous-répertoires `songs`, `img`, `latex`, `templates`, `python`.

`--verbose, -v`
: Affiche davantage de messages lors de la compilation.

`--steps` `<STEPS>`, **`-s`** `<STEPS>`
: Spécifie les étapes de la compilation à effectuer. La valeur par défaut est `tex,pdf,sbx,pdf,clean`. Les étapes disponibles sont :

Étape	Description
`tex`	Produit le fichier `.tex`.
`pdf`	Compile le fichier `.tex`.
`sbx`	Compile les index (chansons et auteurs).
`clean`	Supprime les fichiers temporaires.
`#cmd`	Étape spéciale : la suite d'une chaîne commençant par le signe # sera exécutée dans un shell.

Plusieurs étapes (sauf l'étape spéciale) peuvent être combinées en une seule option `--steps`, séparées par des virgules.

`--cache` `<CACHE>`, **`-c`** `<CACHE>`
: Spécifie si oui ou non le cache devrait être utilisé (lu et écrit). Par défaut, vaut `yes` (le cache est utilisé) ; utiliser `--cache=no` pour désactiver son utilisation.

`--error` `{failonsong,failonbook,fix}`, **`-e`** `{failonsong,failonbook,fix}`
: Par défaut, *songbook* essaye de corriger ou d'ignorer les erreurs (syntaxe, fichiers manquants, etc.) dans les chansons et le carnet. Cette option permet de changer ce comportement.
— `failonsong` : arrête la compilation dés qu'une chanson présentant une erreur a été analysée.
— `failonbook` : arrête la compilation après que toutes les chansons aient été analysées, si au moins une erreur a été rencontrée.
— `fix` (valeur par défaut) : essaye de corriger ou d'ignorer les erreurs.
Il faut remarquer que la compilation peut échouer même avec l'option `--error=fix`.

1.4.2 Utilisation avec patadata

Le projet *patadata* contient des carnets de chants prêts à être compilés. Ils sont dans le répertoire books [30].

Par exemple, pour compiler l'ensemble des chansons de ce répertoire en un seul carnet, il est possible de se placer dans la racine de *patadata* et d'exécuter

```
songbook books/songbook.yaml
```

1.5 Aller plus loin

Le projet Patacrep est relativement modulaire et permet une personalisation avancée des carnets produits, et une gestion fine des fichiers de données.

La gestion des fichiers de données est décrite dans la section *Organisation des datadirs* ; la personalisation des types de contenu intégrables à un carnet dans la section *Ajouter du contenu aux carnet : écrire un plugin* ; et la personalisation de la mise en page des carnets est dans la partie *Changer la mise en page des carnets : le système des templates*.

1.5.1 Organisation des `datadirs`

La bibliothèque `patacrep` va chercher des information dans un ensemble de répertoires, relatif à un répertoire de base (nommé `datadir`). Plusieurs `datadir` peuvent être définis : si le fichier requis n'est pas trouvé dans le premier `datadir`, il est cherché dans le second, et ainsi de suite.

Les sous-répertoires de ces `datadir` sont les suivants.

`img` Ce répertoire contient des images qui peuvent être incluses à l'aide de la commande :

```
{image: image.png}
```

ou en LaTeX :

```
\includegraphics{image.png}
```

Ce répertoire est automatiquement inclus dans la liste des répertoires dans lesquels ces commandes vont chercher les images : pour les fichiers de ce répertoire, il n'est donc pas nécessaire de préciser leur chemin d'accès absolu.

`latex` Ce répertoire peut contenir des fichiers LaTeX (`.tex` ou `.sty`). C'est dans ce répertoire que vous pouvez mettre les fichiers de paquets LaTeX qui ne sont pas inclus avec votre distribution.

[30] https://github.com/patacrep/patadata/tree/master/books

songs Les chansons pouvant être incluses dans le carnet de chant sont dans ce répertoire. Aucune organisation spécifique n'est imposée à l'intérieur de ce répertoire.

templates Comme son nom l'indique, les *templates* sont recherchés dans ce répertoire.

python Dans ce répertoire peuvent être placés des modules Python complémentaires. Pour le moment, ceci est uniquement utilisé pour *écrire ses propres plugins*.

1.5.2 Ajouter du contenu aux carnet : écrire un plugin

Dans toute la suite, nous allons créer un plugin `foo`, associé au mot-clef `foo`, qui écrit dans le carnet le contenu de l'argument `bar`, quasiment sans traitement. Ce plugin se présente sous la forme d'un fichier `foo.py` (le nom de fichier est libre), présent dans un sous répertoire `python/content` d'un `datadir`.

Définition

Un plugin se présente sous la forme d'un fichier Python, présent dans un répertoire `python/content` (relativement à un des `datadir`). Ce fichier doit contenir une variable `CONTENT_PLUGIN`, qui est un dictionnaire dont les clefs sont des mots-clefs, et les valeurs des fonctions *parse*.

Lors de la compilation du carnet, au moment de l'analyse de la variable `content` du fichier `.sb`, lorsqu'un de ces mots-clefs est rencontré, la fonction `parse()` correspondante est appelée.

Notre plugin d'exemple contient donc le code suivant (où `parse()` est une fonction, définie plus tôt dans le fichier, dont nous allons parler dans la partie suivante).

```
CONTENT_PLUGIN = {'foo': parse}
```

Classe `content.Content`

L'objet qui produit quelque chose dans le recueil est une instance de la classe `content.Content`. La méthode d'initialisation est libre, et la méthode principale est la méthode `content.Content.render()`, qui prend en argument le contexte [31] courant, et renvoie une chaîne de caractères à inclure dans le fichier `.tex`.

Plus de détails sur cette classe (ainsi que sur les autres méthodes utilisées) sont disponibles dans le docstring de cette classe [32].

Pour notre exemple, nous allons définir une nouvelle classe `Foo`, héritant de cette classe `content.Content`.

```
from patacrep.content import Content

class Foo(Content):

  def __init__(self, arguments):
    """Fonction d'initialisation

    Le moteur de plugin ne va pas appeler cette fonction directement : chaque
    plugin est donc libre de définir cette initialisation comme il l'entend.
    """
    self.arguments = arguments

  def render(self, __context):
    return self.arguments['bar']
```

http://jinja.pocoo.org/docs/api/#the-context
https://github.com/patacrep/patacrep/blob/v4.0.0/patacrep/content/__init__.py#L82-121

Fonction `parse()`

La fonction `parse()` est appelée lorsque le mot clef est rencontré, avec comme arguments :

keyword le mot clef ayant déclenché l'appel à cette fonction ;

argument l'argument passé au mot-clef ;

config le dictionnaire contenant la configuration du recueil en cours de construction. Le modifier est autorisé.

Ainsi, si le contenu du recueil comprend

```
- foo:
    bar: "something"
    content:
      - "one"
      - "two"
```

notre fonction `parse()` sera appelée avec comme arguments `parse('foo', "bar": « something », "content": ["one", "two"], config)`.

Cette fonction doit retourner une liste (éventuellement vide) d'objets de classe `content.Content` (ou une de ces sous-classes). Ces objets seront intégrés au carnet (en utilisant principalement leur méthode `content.Content.render()`) dans l'ordre dans lequel ils apparaissent dans cette liste.

Notre fonction va donc être la suivante :

```
def parse(keyword, argument, config):
  return [Foo(argument)]
```

Bilan

Notre plugin est maintenant fonctionnel et réagit au mot clé `foo`.

1.5.3 Changer la mise en page des carnets : le système des templates

Le processus de génération d'un carnet en PDF est le suivant :

1. Création du fichier .tex
2. Compilation avec **`lualatex`**
3. Compilation des index
4. Compilation avec **`lualatex`** prenant en compte les index

Après l'étape 1, l'intégralité du carnet (contenu et mise en page) est défini. Pour modifier un carnet, il faut donc se pencher sur cette étape. La gestion du contenu est déléguée aux *plugins*, ici nous allons voir comment modifier la mise en page. LaTeX étant utilisé pour créer ces carnets, une bonne connaissance de ce langage est nécessaire.

Ces fichiers `.tex` sont créés avec le moteur de template Jinja2 [33], en utilisant une syntaxe légèrement différente de la version par défaut et plus adaptée à LaTeX.

Fonctionnement d'un système de templates

Idée de base

Lors de la création de carnets de chants avec LaTeX, on écrit souvent des fichiers ayant la même structure, mais un contenu un peu différent. Un moteur de template permet de décrire la structure du fichier final, en indiquant quelle valeur devra être utilisée à quel endroit pour effectuer le rendu du document.

[33] http://jinja.pocoo.org/

Syntaxe

Il y a deux types d'éléments de syntaxe avec Jinja2 : des variables, qui seront simplement remplacées par leurs valeurs et des instructions de contrôle, qui permettent d'ajouter de la logique dans la construction des templates.

Variables **:** Les variables sont délimitées par deux parenthèses : `((song))` sera remplacé par le contenu de la variable `song` lors du rendu d'un template. Il est possible d'appeler des méthodes python sur les variables : `(( songlist.length() ))` est tout à fait valide (si toutefois l'objet `songlist` a une méthode `length()`).

Les variables accessibles dans un template sont l'ensemble des options définies dans le *bloc de déclaration des variables*, et prennent soit les valeurs par défaut, soit les valeurs définies par les utilisateurs dans les fichiers `.sb`.

Lorsque ces variables sont des chemins de fichiers, il est important de toujours utiliser la fonction `path2posix()` afin de normaliser l'écriture de ces chemins. Par exemple : `(( path2posix(path) ))`.

Instructions **:** Les instructions sont signalées par `(* *)`. Il en existe plusieurs, qui suivent le même format : `(* <instruction> <arguments> *)`, et qui se terminent parfois par `(* end<instruction> *)`.

Les instructions les plus utiles sont détaillées ici.

block **:** L'instruction `(* block *)` permet de placer du contenu à un endroit du document. Si le bloc en question existe dans un des *templates parents*, le contenu du bloc sera placé à cet endroit.

La syntaxe exacte est :

```
(* block mon_bloc *)
   Contenu
(* endblock *)
```

Plusieurs blocs sont définis dans les *templates par défaut*

extends **:** Il est possible d'étendre un template près-existant. Dès lors, tout le contenu écrit dans un bloc sera placé dans le bloc correspondant du template parent. Si le bloc n'existe pas, le contenu sera placé à la fin du fichier, et donc ignoré à la compilation LaTeX car placé après le `\end{document}`. Pour étendre un template particulier, on utilise

```
(* extends "template.tex" *)
```

if **:** Il est possible d'effectuer des placements conditionels avec les instructions `if`. La syntaxe est simplement :

```
(* if <condition1> *)
   Placé si la condition 1 est vraie
(* elif <condition2> *)
   Placé si la condition 2 est vraie
(* else *)
   Placé si les conditions 1 et 2 sont fausses
(* endif *)
```

Les insructions *elif* et *else* sont facultatives, et les conditions peuvent être n'importe quelle expression valide en Python. Par exemple :

```
(* if booktype == "chorded" *)
   \addschords
(* endif *)

(* if textwidth > 42 *)
   \collumns{3}
(* endif *)
```

for **:** Il est possible de répeter un contenu avec une boucle `for`. La syntaxe est la suivante :

```
(* for lang in languages_list *)
   Contenu à être répété, en utilisant la variable ((lang))
(* endfor *)
```

Un cas d'utilisation pourrai être :

```
(* for lang in languages_list *)
   (* if not lang == "french" *)
      \setlang{ ((lang)) }
   (* else *)
      \setmainlang{ ((lang)) }
   (* endif *)
(* endfor *)
```

Pour le reste des fonctionalités de Jinja, vous pouvez aller voir la documentation[34], en retenant que `{{ variable }}` et `{% instruction %}` ont été remplacé par `(( variable ))` et `(* instruction *)`.

Templates par défaut

Les templates suivants sont fournis par défaut par `Patacrep` et remplissent des fonctions différentes.

`layout.tex` Défini l'ensemble des `block` qui seront accessibles aux autres templates. Les blocs suivants sont définis, dans cet ordre :

documentclass Bloc contenant la commande `\documentclass{article}`. À surcharger pour changer la classe LaTeX utilisée, ou ajouter des options ;

preambule Bloc placé avant le `\begin{document}`. Utile pour importer des packages ou redéfinir des macro LaTeX ;

title Bloc utilisé pour placer les commandes de la page de titre, *i.e.* `\maketitle` ;

preface Pour ajouter une préface au recueil ;

index Pour placer les index ;

chords Pour placer une liste d'accords au début du carnet ;

songs Le contenu principal est placé dans ce bloc ;

postface Pour ajouter une postface au recueil.

Ce template ne produit aucun fichier PDF.

`songs.tex` Le template `songs.tex` étends `layout.tex`, et se charge de placer le contenu dans le document. Il contient le minimum nécessaire pour que les chansons (mais pas les index) soient rendues.

`default.tex` Le template `default.tex` étends `songs.tex`, et applique une mise en forme minimale, ainsi que les index. Si vous voulez créer votre propre mise en page, c'est sans doute celui-ci qu'il vous faudra étendre.

`patacrep.tex` Le template `patacrep.tex` étends `default.tex`, et applique la mise en forme spéciale du projet Patacrep. Si vous souhaitez modifier légèrement la mise en page du carnet, ce template sera utile.

Créer son propre template

Pour créer votre propre template et l'utiliser, il vous faudra creer un fichier `mon_template.tex` dans un sous-dossier `templates` d'un *datadir*, et ajouter `template: mon_template.tex` dans la section `book` de votre fichier `.yaml`. Le plus simple pour vous est encore de faire hériter votre template de l'un des templates par défaut de Patacrep, comme `default.tex` ou `patacrep.tex`. Vous pourrez alors (re)définir les commandes LaTeX de votre choix.

La documentation[35] (en anglais) du package `songs` explique comment modifier la mise en page des carnets créés, et quelles commandes redéfinir.

http://jinja.pocoo.org/
http://songs.sourceforge.net/songsdoc/songs.html#sec11

Les variables

Note : Ce paragraphe n'a pas encore été mis-à-jour avec la dernière version de patacrep. Il est conseillé de s'inspirer des templates existants (`default.tex`, `layout.tex`...) et de ne pas hésiter à demander de l'aide sur github [36].

Si vous voulez accéder à des variables dans vos templates avec la syntaxe `(( ma_variable ))`, vous devez définir ces variables au début de votre fichier de template. Ces définitions doivent être placées entre les instructions `(* variables *)` et `(* endvariables *)`, et sont décrites au format JSON selon le schéma suivant :

```
{
"ma_variable": {"description": {"english": "english description", "french":
↪"description française"},
                "default": {"default":[]}
}
```

Les variables sont déclarées dans un dictionnaire, dont les clefs sont les noms des variables, et les valeurs des dictionnaires. Dans ces valeurs peuvent entrer plusieurs clefs, dont les plus utiles sont `"description"`, qui est un dictionnaire de description de cette variable ; et `"default"` qui renseigne la valeur par défaut de cette variable.

La valeur par défaut peut être de tous les types acceptés par JSON (chaînes, listes et dictionnaires) et peut dépendre ou non de la langue, avec la syntaxe suivante :

```
{
"ma_variable": {"default": {"default":"Valeur indépendante de la langue."},
"mon_autre_variable": {"default": {"french":"Valeur par défaut pour un carnet en␣
↪français.",
                                   "english":"Valeur par défaut pour un carnet en␣
↪anglais.",
                                   "default":"Valeur par défaut si la langue n'est␣
↪ni le français ni l'anglais."
                                    }
}
```

[36] https://github.com/patacrep/patacrep/issues/new

CHAPITRE 2

Liens externes

— Page officielle : http://patacrep.com
— Forum : http://www.patacrep.com/forum
— Page développement : http://github.com/patacrep/patacrep

CHAPITRE 3

Projets liés

3.1 patacrep

Le nom *patacrep* désigne à la fois l'ensemble du projet (un ensemble d'outils), et la bibliothèque au cœur du projet (l'outil principal, utilisé par les autres outils).

- *patacrep* regroupe les projets décrits ci-après (*patacrep*, *pataextra*, *patadata*, *patanet*, *patagui*) et est un ensemble d'outils de manipulation de chants et de carnets de chants ;
- *patacrep/patacrep* est la bibliothèque principale, qui fournit les outils principaux nécessaires à cette manipulation, ainsi qu'un outil en ligne de commande (*songbook*).

Pages de développement : organisation [37], patacrep [38].

3.2 patadata

Ce projet contient un grand nombre de chansons pouvant être manipulées par *patacrep*, ainsi que divers carnets de chants les rassemblant.

Page de développement [39]

3.3 patanet

Une interface web est en cours de développement à l'heure où nous écrivons cette documentation. Elle permettra de créer des carnets de chants en utilisant les outils *patacrep*, dans un navigateur web.

Page de développement [40]

http://github.com/patacrep
http://github.com/patacrep/patacrep
http://github.com/patacrep/patadata
http://github.com/patacrep/patanet

3.4 Outils non maintenus

3.4.1 patagui

Une interface graphique a été développée pour une ancienne version de *patacrep*. Nous manquons malheureusement de temps pour la maintenir à jour. Elle devrait fonctionner avec la version 3.7.2 de *patacrep*, mais pas avec les suivantes.

Nous espérons que son développement reprendra dans le futur (par nous ou de nouveaux contributeurs), mais elle est pour le moment obsolète.

Page de développement [41]

3.4.2 pataextra

Quelques outils pour manipuler les chansons sont disponibles dans ce projet. Malheureusement, ce projet est peu maintenu, les outils sont peu documentés, et peut-être obsolètes.

Page de développement [42]

http://github.com/patacrep/patagui
http://github.com/patacrep/pataextra

CHAPITRE 4

Outils externes

Voici quelques outils que l'utilisateur pourrait avoir à manipuler.

4.1 LaTeX

L'ensemble d'outils *patacrep* produit un fichier LaTeX, qui est compilé pour devenir un carnet de chants au format PDF. Les chansons peuvent êtres écrites en *chordpro*, mais il est possible *d'utiliser directement LaTeX*, auquel cas il pourra être utile de se familiariser avec cet outil, avec par exemple Une courte (?) introduction à LaTeX2e [43] (original en anglais [44]).

4.2 Songs

En particulier, le paquet songs [45] (documentation [46] en anglais) est utilisé pour rendre les chansons, et en particulier pour marquer les accords.

Les cas d'utilisation de base sont expliqués dans la section *Écriture des chansons*.

4.3 Lilypond

Il est possible d'intégrer des partitions aux chansons. Si ces partitions peuvent être intégrées comme des images (JPG, PNG, PDF) en utilisant la *directive* `image`, *Lilypond* permet de manipuler les partitions écrites au format Lilypond.

Plus d'informations dans la section *Partitions*, et sur le site du projet [47].

http://mirrors.ctan.org/info/lshort/french/lshort-fr.pdf
http://mirrors.ctan.org/info/lshort/english/lshort.pdf
http://songs.sourceforge.net
http://songs.sourceforge.net/songsdoc/songs.html
http://www.lilypond.org/

4.4 Python

Python [48] est le langage de programmation utilisé pour écrire la majeure partie des outils de *patacrep*. S'il n'est pas nécessaire d'avoir entendu parler de ce langage pour créer un carnet de chant, c'est en Python que peuvent être écrites des extensions permettant d'inclure de nouveaux formats de contenu aux carnets de chant. Voir la section *Ajouter du contenu aux carnet : écrire un plugin* pour plus d'informations.

4.5 jinja2

Jinja2 [49] est le moteur utilisé pour écrire les templates. Bien qu'écrit en Python, la connaissance de ce langage n'est nécessaire que pour des cas d'utilisation avancés.

L'utilisation pour les cas de base est décrite dans la section *Changer la mise en page des carnets : le système des templates*.

http://python.org
http://jinja.pocoo.org/

Index

Symbols

S