卒業論文

hikiutils を用いた
卒業論文作成

関西学院大学　理工学部　情報科学科

1234 西谷滋人

2017 年 3 月

指導教員　　西谷　滋人　教授

目次

目次

■ユーザ独自の help を作成する gem

■gem とは　正式名称は RubyGems．Ruby 用のライブラリを使う時に必要となるソフトウェアのこと．パッケージ管理ツール gem があることで，Ruby 用ライブラリのインストール，アンインストール，バージョン管理などを簡単に行うことができる．プログラミング言語 Ruby のファイルに付属されていて，無料で利用することができる．参考文献 2(https://blog.codecamp.jp/rails-gem 1/27)

■gem のメリット

- 標準化された構造があるので，初めてみた人でも分かるようになっている．
- gem があることで，簡単に Ruby 用ライブラリをインストールでき，初心者でもアプリ機能を装備できる．
- 誰でも作成，配布が可能である．

■my_help とは　西谷研究室で使用している gem.

```
Usage: my_help [options]
    -v, --version                    show program Version.
    -l, --list                       list specific helps
    -e, --edit NAME                  edit NAME help(eg test_help)
    -i, --init NAME                  initialize NAME help(eg test_help).
    -m, --make                       make executables for all helps.
    -c, --clean                      clean up exe dir.
        --install_local              install local after edit helps
        --delete NAME                delete NAME help
        --hiki                       my_help2hiki
```

上記のようなコマンドが用意されていて，ユーザ独自の help(メモ) を作成することができる．

■my_help を研究室内で利用する利点

- 研究室内でのメモの書き方が統一できる．
- どこにメモをしたか忘れることがない．
- 普段研究の為に使うターミナルから離れること無くメモを残すことができるので，

書きたいときにすぐに書くことができる.

■hiki **とは**　Ruby で書かれた高機能・高速 Wiki クローン．参考文献 3(`http://hikiwiki.org/ja/` 1/27) CGI(Commond Gateway Interface) を利用して，Web サーバと連動して動く．参考文献 4(`https://ja.wikipedia.org/wiki/Hiki` 1/27) 西谷研究室では，hiki の形式を利用してサイトを作り，研究室内での情報共有や gem の使い方などを掲載して閲覧できるようにしている．また，卒業論文の作成にも hiki の形式で作成している．

■TeX とは 「テック」または「テフ」と読む，組版ソフト．スタンフォード大学の Donald E.Knuth 教授によって作られた．他のソフトと組み合わせて，組版結果を画面や紙に出力したり，PDF 形式で出力したりする．

■TeX の特徴

- フリーソフトなので無料で入手でき，自由に中身を調べたり改良したりできる．
- 商品利用が自由．
- Windows，Mac，Linux などの OS に関わらず同じ動作をする．
- TeX の文書はテキストファイルなので，普通のテキストエディタで読み書きでき，再利用やデータベース化が容易．
- 数式の組版に定評があり，数式をテキスト形式で表す標準となっている．

■latex とは 「ラテック」または「ラテフ」と読む．DEC(現在の HP) のコンピュータ科学者 Leslie Lamport によって機能強化された TeX. もとの TeX と同様にフリーソフトとして配布されている．

■gem 中の hikis から hiki への自動変換　ruby のライブラリーパッケージの標準となる gem の directory 構造に hikis という directory を作って文書作成している．hiki –initialize でこのなかの hikidoc ファイルをウェブ上の hiki と同期する機能を提供する．これによって，gem/hikis で作成した文書は，github あるいは rubygems.org を通じて共有可能となる．以下にこの同期をスムーズに行うための幾つかの convention を使用法とともにまとめている．

■使用法，コマンド

- hiki –initialize で必要なファイル (Rakefile, ./.hikirc, hiki_help.yml) が copy される
- hiki_help.yml は適宜 /.my_help に copy して hiki_help として利用 `my_help` **参照** `(MyHelp_install)`
- rake sync によって hiki ディレクトリーと同期が取られる
- rake convert 20 TARGET.png によって，figs/TAERGET.png に 20
- hiki -u TARGET によってブラウザー表示される

■同期に関する制約

- hiki はフラットな directory 構造を取っている
- hiki の文書はスネーク表記 (例えば，latex2hiki_saki) で階層構造に似せている
- hiki の url の接頭語となる名前を basename の directory 名とする．
- directory 名が'hikis' である場合はその親 directory 名となる．
- /.hikirc の target directory を同期先の directory とする．
- /.hikirc がない場合は同期先の directory を聞く．
- それらは./.hikirc に保存される

■テキスト

- テキストの拡張子は'.hiki' としている
- hiki での url はテキスト前とディレクトリーから自動生成される
- 例えば，hiki2latex_saki/introduciton.hiki とすると hiki2latex_saki_introduciton と変換される
- attach_anchor では

```
'{{attach_anchor(test.png, hiki2latex_saki)}}'
```

と，directory 指定しなければならない.

1.4 メモ，wiki，latex の自動変換システム

1.4.1　my_help2hiki とは

個々のユーザが独自のメモを作ることのできる gem，my_help を利用して作成したメモを wiki へ自動変換することで，個人のメモを互いに見れるようにし，研究室内で情報を共有する．

1.4.2　コマンド

TARGET –push　作成したメモ (TARGET) をサーバに送る
my_help –hiki　作成したメモを hiki 形式に変換し，wiki で表示できるようにする．

■コマンドの振る舞い

■TARGET –push

```
date_dir = File.join(ENV['HOME'],'.my_help')
FileUtils.cd(data_dir)
system "rm -rf ~/.my_help/*.yml~"
system "scp -r ~/.my_help saki@nishitani0:~"
system "ssh saki@nishitani0 ls ~/.my_help"
```

- 1，2 行目

my_help では，作成したメモが.my_help のディレクトリに自動的に追加されるので，ディレクトリを.my_help に移動する．

- 3 行目

.my_help にメモが追加されるとき，yaml 形式のファイルで保存される．メモを更新すると，一つ前に保存したファイルは*.yml というファイル名でバックアップとして残される．**rm -rf** で不必要なファイルは削除し，サーバにコピーするときのデータ量を減らしている．

- 4 行目

scp -r　/[directory 名] [server 名] server に ssh 接続を行い，directory を server にコピーする．-r はディレクトリ全体をコピーすることを示している．西谷研究室で利用している nishitani0 というサーバにコピーしている．

- 失敗例

ターミナルでこのコマンドを使わずにコピーする場合

```
/Users/saki/my_help% ssh nishitani0
Last login: Mon Jan  2 22:16:29 2017 from 59x87x69x163.ap59.ftth.ucom.ne.jp
[nishitani0:~] saki% cp -r ~/.my_help saki@nishitani0
[nishitani0:~] saki% exit
logout
Connection to nishitani0 closed.
```

上記にようにすると，ssh で接続し.my_help を directory を cp を使ってコピー，ssh 接続を切るという作業を 1 行ずつ行うことができる．同じようにコマンドの中身を下記のようにした．

```
system "ssh nishitani0"
system "cp -r ~/.my_help saki@nishitani0"
system "exit"
```

実行結果

```
/Users/saki/my_help% bundle exec exe/my_todo --push
Last login: Sat Jan 28 03:49:46 2017 from 59x87x69x163.ap59.ftth.ucom.ne.jp
[nishitani0:~] saki%
```

このように 1 行目のみ実行したところで終了してしまった．ssh 接続を行うと，その後の命令は全て server 側で実行されることになるため，1 行ずつ行うと途中でターミナルでは実行ができなくなってしまいうまくいかない．

- 失敗例 2

ssh [**server 名**] [**command**] とすると server に ssh で接続して command を実行しログアウトすることができるため，以下のように書き換えた．

```
system "ssh nishitani0 cp ~/.my_help saki@nishitani0"
```

実行結果

```
/Users/saki/my_help% bundle exec exe/my_todo --push
```

```
cp: /Users/saki/.my_help is a directory (not copied).
```

ssh でファイルやディレクトリをコピーするコマンドは cp ではなく scp なので，実行することができなかった．

- 5 行目

nishitani0 に ssh 接続し.my_help の中身を書き出して，コピーができているかコマンドを実行した時に確認が行えるようにしている．

■my_help –hiki

```
system "emacs_help --to_hiki > ~/Sites/hiki-1.0/data/text/emacs_help_saki"
system "my_todo --to_hiki > ~/Sites/hiki-1.0/data/text/my_todo_saki"
system "ssh_help --to_hiki > ~/Sites/hiki-1.0/data/text/ssh_help_saki"
system "open -a safari 'http://localhost/~saki/hiki-1.0/?FrontPage'"
```

- 1 3 行目

my_help には，**TARGET –to_hiki** というコマンドがあり，これによって yaml 形式で保存されているメモを hiki 形式で書き出すことができる．この –hiki のコマンドを使って hiki 形式にしたものを，wiki で表示することのできるフォルダである **/Sites/hiki-1.0/data/text/**に入れることで，wiki での表示を可能にしている．emacs_help，my_todo，ssh_help は全て私の my_help に入っているメモ．

- 4 行目

wiki のページ **FrontPage** を表示するコマンド．これによりメモが更新されているのを即時確認することができる．FrontPage は以下のようになっている．

```
!saki's help
*[[ssh_help_saki]]
*[[my_todo_saki]]
*[[emacs_help_saki]]
```

先頭に**!**をつけることで 1 行目の saki's help を見出しにし，2 4 行目は*****によって箇条書き，各括弧でリンクになっている．

■利便性　メモを更新してから，上記の２つのコマンドを共に行うと，自分の my_help にあるメモ，サーバのバックアップ，研究室内のメンバーが見ることのできる wiki の全てが同じ内容になる．研究室内全員が同じ情報を得ることで，情報を共有するという目標が達成できる．また，メモをすべて my_help で作成しておけば，忘れてしまったときに my_help のみを確認すればよく，管理が楽になる．gem の開発中や課題の途中にもすぐメモを残すことができるので，それぞれのユーザが得た暗黙知もすぐに形式知化することが可能になることが期待される．

■hiki 形式で書かれた文章を latex 形式に自動変換する

■使用法，コマンド

- hiki2latex -v : hiki2latex の version 表示
- hiki2latex -p FILE : latex の一般的なフォーマットへ変換
- hiki2latex -b FILE :

```
Usage: hiki2latex [options] FILE
    -v, --version                    show program Version.
    -l, --level VALUE                set Level for output section.
    -p, --plain FILE                 make Plain document.
    -b, --bare FILE                  make Bare document.
        --head FILE                  put headers of maketitle file.
        --pre FILE                   put preamble file.
        --post FILE                  put post file.
        --listings                   use listings.sty for preformat with style
```

■gem 中の hikis から hiki への自動変換　ruby のライブラリーパッケージの標準となる gem の directory 構造に hikis という directory を作って文書作成している．hiki –initialize でこのなかの hikidoc ファイルをウェブ上の hiki と同期する機能を提供する．これによって，gem/hikis で作成した文書は，github あるいは rubygems.org を通じて共有可能となる．以下にこの同期をスムーズに行うための幾つかの convention を使用法とともにまとめている．

■使用法，コマンド

- hiki –initialize で必要なファイル (Rakefile, ./.hikirc, hiki_help.yml) が copy される
- hiki_help.yml は適宜 /.my_help に copy して hiki_help として利用 `my_help` **参照** `(MyHelp_install)`
- rake sync によって hiki ディレクトリーと同期が取られる
- rake convert 20 TARGET.png によって，figs/TAERGET.png に 20
- hiki -u TARGET によってブラウザー表示される

■同期に関する制約

- hiki はフラットな directory 構造を取っている
- hiki の文書はスネーク表記 (例えば，latex2hiki_saki) で階層構造に似せている
- hiki の url の接頭語となる名前を basename の directory 名とする.
- directory 名が'hikis' である場合はその親 directory 名となる.
- /.hikirc の target directory を同期先の directory とする.
- /.hikirc がない場合は同期先の directory を聞く.
- それらは./.hikirc に保存される

■テキスト

- テキストの拡張子は'.hiki' としている
- hiki での url はテキスト前とディレクトリーから自動生成される
- 例えば，hiki2latex_saki/introduciton.hiki とすると hiki2latex_saki_introduciton と変換される
- attach_anchor では

```
'{{attach_anchor(test.png, hiki2latex_saki)}}'
```

と，directory 指定しなければならない.

my_help2hiki_saki_hiki_sync のコピー書き換えていく