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

- `{{attach_anchor(hikiutils_bob.pdf,hikiutils_bob)}}`

1　イントロダクション

卒業論文は，大学で課される普通の授業のレポートに比べて深い内容が要求されます．また，研究という側面から体裁や版権など出版における多くの掟を遵守しながら，高い品質を保つ必要です．そのため，指導教官との編集作業が重要となります．編集作業の効率を高めるためには，共同作業を促進するプラットフォームが必要です．

体裁などを考えると Latex の使用が標準ですが，執筆段階での煩わしさを低減するため，Mark Up 記法による文章作成が一般的です (「ドキュメント作成システム構築ガイド，GitHub, RedPen, Asciidoctor, CI によるモダンライティング」, 伊藤敬彦，吉村孝広，技術評論社)．西谷研では Mark Up 記法の一種である hiki 記法を用いて執筆します．hiki 記法から html へ変換するソフト hikidoc による容易な変換が可能なため，日記 web appli の hiki diary でも利用されている有名なソフトです．hiki 記法をベースにして，wiki wiki web と同等の機能を提供する hiki システムが github に公開されており，個人の手持ちのパソコンにインストールして wiki を利用することができます．西谷研ではさらに hiki 記法を拡張して，コードのカラー表示，数式の latex 記述が可能となるシステムを利用しています．

本資料では，卒論の編集作業を効率化する図に示すようなシステムの使い方を紹介します．

例えば，卒業生を yamane としましょう．yamane の個人の Mac の自分の directory(hikiutils_yamane) に幾つかのファイルを作成して卒論を書いているとします．これを指導教官 (bob) が編集するとします．この同期には，Github により提供される共同作業環境を使います．これだけでは編集中の文書の体裁がわかりにくいでしょう．そこで，hiki システムにより容易に web ブラウザ上に完成形を表示しつつ執筆することが求められます．このような環境を提供するのが，本システムです．

2　hiki との同期

2.1　hikiutils の install

hikiuitls を gem から install しておく必要がある．コマンドは以下の通り．

```
gem install hikiutils
```

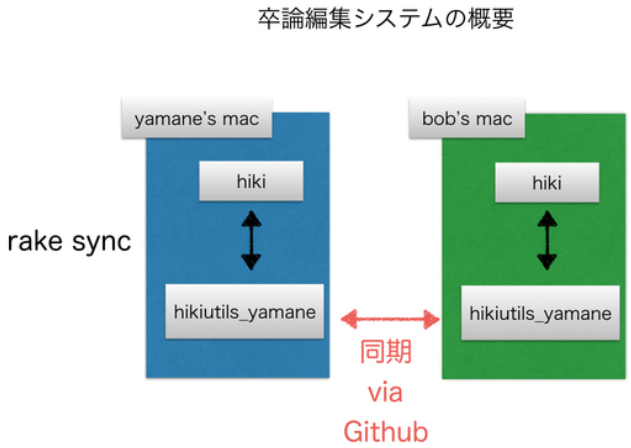

図 1　卒論編集システムの基本概念.

さらに

```
hiki -v
```

で 0.2.3.2 以上であることを確認.

2.2　個別ディレクトリーの構成

図に hikiutils_bob のディレクトリー構成を示す.
コマンド

```
hiki -i
```

によって以下のようなファイルが作成される.

```
bob% ls -lat
total 1072
-rw-r--r--    1 bob   501       99   1 20 12:44 .hikirc
drwxr-xr-x    5 bob   501      170   1 20 12:44 figs/
drwxr-xr-x   12 bob   501      408   1 20 12:34 ./
-rw-r--r--    1 bob   501        8   1 20 11:01 .gitignore
-rw-r--r--    1 bob   501     3507   1 20 11:01 Rakefile
```

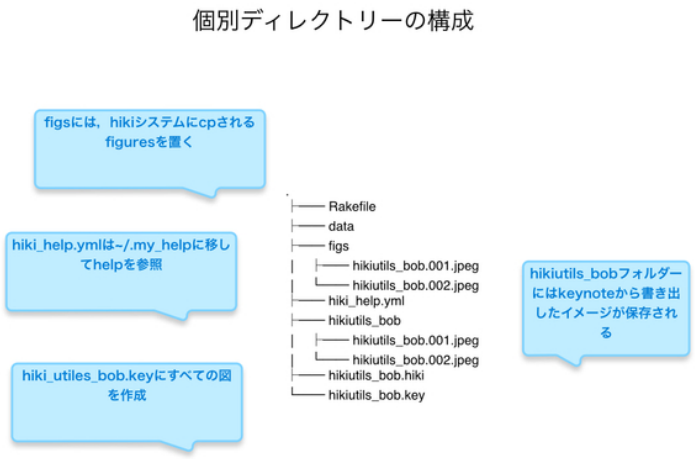

図 2　hikiutils_bob のディレクトリー構成.

```
drwxr-xr-x   2 bob  501       68  1 20 11:01 data/
-rw-r--r--   1 bob  501     2595  1 20 11:01 hiki_help.yml
drwxr-xr-x  26 bob  501      884  1 20 11:00 ../
```

この.hikirc にデータが設定データが自動的に入る．さらに hikiutils_bob.hiki および hikiutils_bob.key を作成する．これで執筆ファイルの基本構成が出来上がる．

keynote で図を作成して，hikiutils_bob.hiki に文章を記述していく．

2.3　システムの概要

図に論文作成システムの概要を示している．gem のなかに作られた例えば，hikiutils_bob などの directory 内に Rakefile などを次節のコマンドを使って入れる．この directory で hikiutils_bob.hiki などの hiki フォーマットの document を作っていく．rake sync によって web を提供する hiki の text, attach, parser などと同期する．一方，最終の卒業論文形態である pdf にするために latex_dir 内に*.tex を生成する．これは次章で解説する．

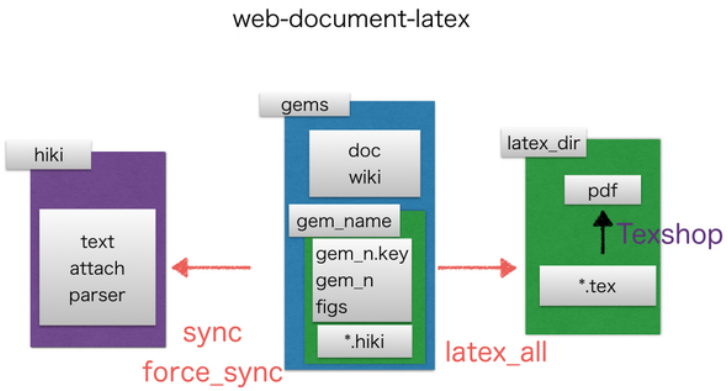

図 3　システムの概要.

2.4　一般的な執筆手順

1. 書類の作成
   (a) open -a mi hikiutils_bob.hiki
2. keynote を開ける
   (a) open hikiutils_bob.key
3. keynote のイメージを figs に
   (a) keynote でイメージへ書き出し (hikiutils_bob を仮定)
   (b) rake convert 60 hikiutils_bob
4. hiki システムとの同期
   (a) rake sync
5. hiki システムで表示
   (a) hiki -u hikiutils_bob (個別の hiki file を表示)
   (b) rake touch (全てを表示)

2.5 rake が用意しているタスク

rake の用意しているコマンドは次のとおり．

```
rake sync              # hikiの同期
rake force_sync        # hikiの強制同期
rake chenv             # For hiki Errno::ENOENT, Errno::EACCES

rake convert           # convert fig size SCALE TARGET_DIR
rake increment         # increment fig NUBERS in FILE
rake number            # numbering figs from the NUBER in FILE

rake touch             # FILE.hikiあるいはhikiファイルすべてを最新状態に更新(
    new)
rake open              # FILE.hikiあるいはhikiファイルをedtiorで開く(new)
rake github            # Githubのdirをsafariでopen(new)
rake reset_hiki        # hikiシステムにあるゴミファイルを掃除する(new)
```

2.5.1 rake sync

hikiutils_bob にある必要な書類を hiki システムにコピーする．その際，名前の書き換えを行う．

表 1

hikiutils_bob での名前	hiki システムでの名前
hikiutils_bob.hiki	hikiutils_bob
introduction.hiki	hikiutils_bob_introduction

figs ディレクトリー内のファイルは hiki/cache/attache/hikiutils_bob に cp される．従って，hiki 文書中で参照するには，`{{attach_view(hogehoge.png)}}` という記述が必要となる．

2.5.2 rake force_sync

hiki システム側で直接変更を加えると，hikiutils が sync した時と差ができる．これを検知して，ユーザに注意を喚起する仕組みがある（rake check_previous)．これは sync した時に自動的に呼び出される．違いの出た files を修正した後に強制的に同期をとるためのコマンドとして，force_sync が用意されている．

2.5.3 rake chenv

hiki システム上で error が出た場合 (Errno::ENOENT, Errno::EACCES) に試してほしい．error の状況は個人の設定によってちがうため，対処法の実装は網羅されていない．うまくいかない場合は西谷に Issues として投げるように．

2.5.4 rake convert VAL DIR

keynote が吐き出したイメージを変換するためのコマンド．ImageMagick がインストールされている必要がある．ない場合は，自分で brew から install するか，`https://www.imagemagick.org/script/binary-releases.php` からダウンロード．うまくいかない場合は donkey に聞いてみてください．

```
rake convert 80 hikiutils_bob
```

によって，DIR=hikiutils_bob にある png ファイルを VAL=80

2.5.5 rake increment

keynote でページを追加すると hiki での参照 (attach_view) にずれが生じる．いまのところこれを解消する方法はなく手で修正を加える必要がある．ずれが単純な場合には，

```
1 cp hikiutils_bob.hiki tmp.hiki
2 rake increment 2 tmp.hiki > tmp2.hiki
```

として attach_view のページ番号を単純に増加させることができる．

2.5.6 rake number

rake increment と同じく figs 内の通し番号が変わった時に attach_view の通し番号を調整するコマンド．

```
rake number 3 hikiutils_bob.hiki > tmp.hiki
cp tmp.hiki hikiutils_bob.hiki
```

とすると

```
8c8
{{attach_view(hikiutils_bob.002.jpeg)}}
---
{{attach_view(hikiutils_bob.003.jpeg)}}
21c21
{{attach_view(hikiutils_bob.003.jpeg)}}
---
```

```
{{attach_view(hikiutils_bob.004.jpeg)}}
```

などと番号を 3 から順に振り替えてくれる.

2.5.7 rake open [FILE] (new)

FILE.hiki あるいは hiki ファイルを edtior で開く.

2.5.8 rake touch [FILE](new)

FILE.hiki あるいは hiki ファイルすべてを最新状態に更新.

2.5.9 rake github(new)

git remote -v にある origin を safari で open します.

2.5.10 rake reset_hiki(new)

hiki システムにあるゴミファイルを掃除します.

```
bob% rake reset_hiki
hikiutils: provide utilities for helping hiki editing.
"open␣-a␣mi"
target_dir : /Users/bob/Sites/nishitani0/Internal/data/text
-rw-rw-rw-  1 bob  staff    182  2  5 12:03 hikiutils_bob
-rw-rw-rw-  1 bob  staff   7220  2  5 12:03 hikiutils_bob_latex_all
-rw-rw-rw-  1 bob  staff  10104  2  5 12:03 hikiutils_bob_sync
-rw-r--r--  1 bob  staff    944  2  5 12:03 hikiutils_bob_toc
remove hikiutils_bob_toc[ynqlA]?
```

となります. yes, no, quit, list, All のなかから選んでください.

2.6 github による同期

これらの作業の流れを示すシーケンス図は次の通りである.
git init, fork が済んでいると仮定して同期の手順を以下に示す.

1. git push 作業
   (a) git add -A
   (b) git commit -m 'first commit'
   (c) git push origin master
2. github で bob へ pull request をかける
3. bob の編集後，通知がくる. まつことなく作業しても ok.

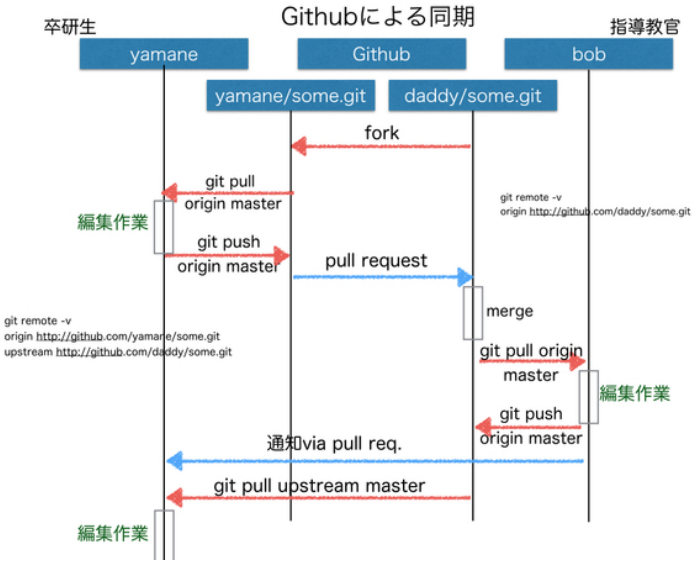

図 4　卒論編集時の github による同期のシーケンス図.

4. git pull 作業
   (a) git pull upstream master

以下には関連するコマンドと動作を記した.

git remote -v　github の remote アドレスを表示
git remote add upstream git@...　github のアドレスを upstream として登録.

2.7　hikiutil 関連のヘルプ

2.7.1　hiki で卒論を書くときの初期化と掟

- 目的：西谷が後で迷わないように決まったファイル構造を堅持すべし
- 文書：hiki で書く. のちには，latex に変換するプログラムを提供します
- 図表：すべての図表を keynote にまとめる，タイトルを分かりやすく書く
- データ：data ディレクトリにまとめる．ファイル名を keynote の対応する図表中に記す
- hiki –initialize で初期ファイル (Rakefile, ./.hikirc, hiki_help.yml) が copy される

- hiki_help.yml を適宜 /.my_help に copy して hiki_help として利用，(my_help 参照)
- Errno::EACCES や permission error がでたときは rake chenv を試してみる（報告して）
- rake sync によって hiki ディレクトリーと同期が取られる
- hiki -u TARGET によってブラウザー表示される
- テキストの拡張子は'.hiki'
- 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

2.7.2 図表に関する制約

- すべての図表を keynote にまとめる
- keynote に書いたスライドはイメージに書き出して，rake convert 80 TARGET_DIR で figs に変換
- rake sync で figs にある files は hiki/target_dir に cp される
- `convert #{source} -resize 20% #{target}` **によって,** `target=figs/TAERGET.png` **に** 20% **に縮小して保存される**
- convert -density 300 view.svg view.png で 300dpi で変換
- attach_anchor では'`{{attach_anchor(test.png, hiki2latex_saki)}}`'と，directory 指定しなければならない.
- keynote であとで図を挿入して番号が変わった時の原稿の一括変換
- rake increment 2 boundary_bob.hiki boundary_bob ¿ tmp.hiki
- rake convert 60 boundary_bob
- rake sync
- hiki -u boundary_bob_tmp

2.7.3 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

2.7.4 テキストに関する制約

- 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

3　実際の最終形態 (卒論=pdf) への変換

hiki+keynote で卒論の内容ができたら，それを卒論の最終形態つまり pdf へ変換する必要が出てきます．

ここで問題が発生します．hiki よりも latex の方が機能が豊富なこと．つまり，簡易に書くには hiki などの mark down で書いていくのがいいんですが，文書として完成度を高めるには，latex で細かい設定を調整する必要がどうしても出てきます．

具体的には，

1. 図の配置を調整する wrap の数値調整
2. 参照文献の記述
3. リスト，引用の体裁
4. 章立て階層構造

などが問題になるところです．

これは hiki ではどうしようもありません．一つの手はもう少し高機能な mark up 言語例えば asciidoc などに変更することですが，これはどこまでいっても終わりのない方向のようで，結局は latex で書いているのと同じになる可能性があります．

我々は違う戦略をとります．それは，

`latex` をベースにして，`hiki` を生成する

と言う手です．

DRY(Don't Repeat Yourself) 原則さえ守れば，文書管理はいいのですから，ある段階まですすめば hiki ではなく latex を原本にするのです．そのための変換器 latex2hiki とその派生 rake 環境が用意されています．そちらはこちらで卒業後に変換します．参照文献は latex に移してから修正してください．

ここでは，hiki でできることをとことん突き詰めておきます．

3.1　install

```
hiki -i
```

で install されています．新たに使うコマンド群は次の通りです．

```
rake latex             # FILE1 を latex に変換
rake latex_all         # すべての hiki ファイルを latex 変換
rake latex_wrap        # FILE1 を wrap format で latex に変換
rake reset_latex_dir   # latex_dir のゴミファイルを削除
```

3.2 rake latex（個別ファイルの変換）

```
rake latex sync.hiki
```

とすると，

```
latex_dir/sync.tex
```

に latex 変換後のファイルが生成します．これを，TeXShop で command_t 変換します．完成例はこちらです．

- {{attach_anchor(sync.pdf,hikiutils_bob)}}

うまくいかないときは，terminal で

```
platex sync.tex
dvipdfmx sync.dvi
```

を試してみてください．

3.2.1 注意

hiki の初めの部分は，

```
bob% head -3 hikiutils_bob.hiki
! title:hikiutils - i による卒論作成システム
! autor:Shigeto R. Nishitani
! date: Kwansei Gakuen Univ., 2017/1
```

とすると，

```
\begin{document}
\title{hikiutils - i による卒論作成システム}
\author{Shigeto R. Nishitani}
\date{ Kwansei Gakuen Univ., 2017/1}
\maketitle
```

と変換してくれます．次節の latex_all では，basename.hiki に書かれたそれらの情報は，head.tex で title などを用意するべきなので，自動で消されます．

必要な図は，figs から自動で取るようになっていますが，サイズや解像度が問題のときは手動で調整してください．

3.3 rake latex_all 手順（ディレクトリー内の一括変換）

pwd の directory と同名の basename に.hiki の拡張子がついたファイルが用意されていて，

```
rake latex_all
```

をおこなうと自動で全部を一体にまとめた文書への latex 変換が出来上がります．たとえば，hikiutils_bob.hiki の記述を

```
bob% cat hikiutils_bob.hiki
!hikiutilsを用いた卒業論文作成

! [[hikiutils_bob_sync]]
! [[hikiutils_bob_latex_all]]
```

とすると，sync.tex, latex_all.tex が latex_dir 内に変換されます．さらに，開いている hikiutils_bob.tex を TeXShop で変換してみてください．

うまくできないときは**ここ** (http://qiita.com/hideaki_polisci/items/3afd204449c6cdd995c9 を参照して自力で入れてみてください．だめなら donkey.

- {{attach_anchor(hikiutils_bob.pdf,hikiutils_bob)}}

というようになります．

3.3.1 下準備

latex_dir 内に幾つかの tex 雛形を入れておく必要があります．

```
hiki -i
```

と hikiutils 環境を再度初期化すると，自動でインストールする設定です．なかったら手動で作ってください．Rakefile 以外は上書きしません．また，latex_dir/head.tex は下に記す修正が必要です．

■head.tex　題目，学生番号，氏名を変更する．年月をチェック．
setcountertocdepth は toc をどこまで表示するかのレベルに対応します．原稿作成時は階層がわかりやすいように深めにしていますが，本番では 2 程度で十分です．

```
bob% cat head.tex
\title{卒業論文\\
\vspace{4cm} hikiutilsを用いた\\卒業論文作成}
\author{ 関西学院大学　理工学部　情報科学科\\\\1234 西谷滋人}
\date{\vspace{3cm} 2017年　3月\\
\vspace{3cm} 指導教員　　西谷　滋人　教授}
\maketitle
\setcounter{tocdepth}{6} %
```

■pre.tex　latex_all のときは使っていません．formated の時に読むようにしていますが... 読めてないようです．いずれ調べます．

■main の preamble 部　あまり変更しないほうがいいですが，いずれいじることになります．例えば，フォントポイント数を 12 から 10pt に変えるなどです．

```
\documentclass[12pt,a4paper]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\usepackage[dvipdfmx]{color}
\usepackage{listings,jlisting}% to use japanese correctly, install jlistings.
\lstset{
  basicstyle={\small\ttfamily},
  identifierstyle={\small},
  commentstyle={\small\itshape\color{red}},
  keywordstyle={\small\bfseries\color{cyan}},
  ndkeywordstyle={\small},
  stringstyle={\small\color{blue}},
  frame={tb},
  breaklines=true,
  numbers=left,
  numberstyle={\scriptsize},
  stepnumber=1,
  numbersep=1zw,
  xrightmargin=0zw,
  xleftmargin=3zw,
  lineskip=-0.5ex
}
\lstdefinestyle{customCsh}{
  language={csh},
  numbers=none,
}
\lstdefinestyle{customRuby}{
  language={ruby},
  numbers=left,
}
\lstdefinestyle{customTex}{
  language={tex},
```

```
  numbers=none,
}
\lstdefinestyle{customJava}{
  language={java},
  numbers=left,
}
```

3.4 補助コマンドの解説

3.4.1 rake reset_latex_dir(latex_dir のゴミ掃除)

わかりやすいようにまとめたり，ファイルの名称を変更した時には，過去の tex ファイルが残る．それらを整理する時に使用．head.tex だけを escape して rm -rf で消すので，注意が必要．特に図形の files や bb files は消えるので注意．

3.4.2 wrap 関係

figure 環境を wrapfigure 環境で作るための幾つかのコマンド群です．卒論では figure 環境で作る方がいいんですが，journal 論文などのページ数が制限された場合は，wrapfigure で text の回りこみや位置調整を行う必要があります．それらのための環境を埋め込む仕組みです．

- rake change_wrap(wrap で変換)
- rake latex_base(latex に変換するだけの下請け)
- rake latex_wrap(figure 環境だけを wrapfig 環境に変える)

```
1 desc "latex␣conversion␣FILE1"
2 task :latex => [:latex_base] do
3   exit
4 end
5
6 desc "latex␣conversion␣FILE1␣with␣wrap␣format"
7 task :latex_wrap => [:latex_base, :change_wrap] do
8   exit
9 end
```

完成例はこちらです．

- {{attach_anchor(calphad_bob.pdf,hikiutils_bob)}}

3.5　参照のシステム

latex へ文献参照を渡すために，下記のようなフォーマットでの記述を行ってください．

```
図{{ref(fig:SystemOverview)}}に卒論編集システムの概観を示しています．

!!!caption: (fig:SystemOverview)卒論編集システムの概観．
{{attach_view(hikiutils_bob.006.jpeg)}}

*基本的な使い方{{cite(listings1)}}
*独自のカラー化{{cite(listings2)}}
!reference:
:listings1:[[http://d.hatena.ne.jp/mallowlabs/20061226/1167137637]]
:listings2:[[http://www.ipc.akita-nct.ac.jp/~yamamoto/comp/latex/make_doc/
    source/source.html]]
```

ここで，fig:以下に snake name は使えません．Camel で書くようにしてください．

これを sync, all_latex をかけると図 5 のようになります．latex の文献参照 (cite, bibliography)，図表番号の参照 (ref, label) が正常に処理されていることが確認できます．また，web 表示した場合でもそれほご違和感なく読めるでしょう．

図 5　参照機能を利用した場合の変換後の表示．

4 Rakefile 開発に関するメモ

4.1 日本語の code listings

日本語の jlisting の使用法が判明しました．listings では日本語表示がちゃんとなされません．そこで，

```
\usepackage{listings,jlisting}
```

とします．これで，日本語が含まれた code も綺麗に表示してくれます．

4.2 参照のシステム

latex へ文献参照を渡すために，下記のようなフォーマットでの記述を行ってください．

```
図{{ref(fig:SystemOverview)}}に卒論編集システムの概観を示しています．

!!!caption: (fig:SystemOverview)卒論編集システムの概観．
{{attach_view(hikiutils_bob.006.jpeg)}}

*基本的な使い方{{cite(listings1)}}
*独自のカラー化{{cite(listings2)}}
!reference:
:listings1:[[http://d.hatena.ne.jp/mallowlabs/20061226/1167137637]]
:listings2:[[http://www.ipc.akita-nct.ac.jp/~yamamoto/comp/latex/make_doc/
    source/source.html]]
```

ここで，fig:以下に snake name は使えません．Camel で書くようにしてください．

これを latex にかけると次の通りになります．（体裁は乱れている）

図 6 に卒論編集システムの概観を示しています．

- 基本的な使い方 [1]
- 独自のカラー化 [2]

参考文献

[1] http://d.hatena.ne.jp/mallowlabs/20061226/1167137637

[2] http://www.ipc.akita-nct.ac.jp/~yamamoto/comp/latex/make_doc/source/source.html

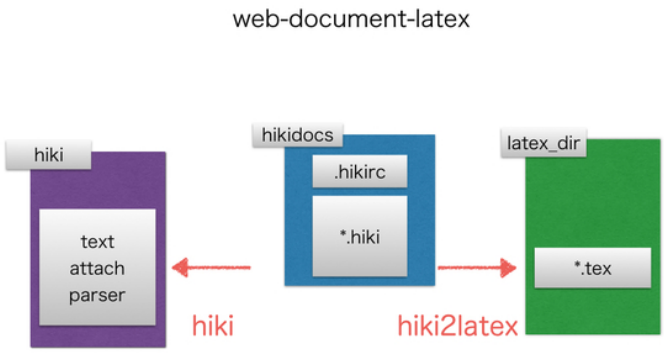

図 6　卒論編集システムの概観.

4.3　システムの概要

卒論編集システム開発時のメモです．図 6 に卒論編集システムの概観を示しています．

hiki システムとの同期は，hikiutils の hiki が下請けしている．一方，latex_dir への出力は latex2hiki が引き受けている．フォーマットをいじるときには，基本的に

hiki へ　hiki がやるので，そのまえに tmp.txt へ写して置換
tex へ　latex2hiki からの出力 (tmp.txt) を処理して tex へ

で行っている，あるいはおこなう．

4.4　rake mk_toc

latex が作成する tableofcontents の実態である toc ファイルから hiki 用の toc.hiki を作成する．