The Asian Journal of TEX, Volume 4, No. 2, December 2010
Article revision 2010/12/27

BibTEX-based Manuscript Writing Support System for Researchers*

Shin-ichi Todoroki
National Institute for Materials Science,
Namiki 1-1, Tsukuba, Ibaraki
305-0044, Japan
Todoroki.Shin-ichi@nims.go.jp

Tomoya Konishi
Anan National College of Technology,
265 Aoki Minobayashi, Anan, Tokushima
774-0017, Japan
konishi@anan-nct.ac.jp

Keywords BibTEX, Ruby, Publication list

Abstract A list of publications can help researchers with their writing if each item on the list includes links to their manuscript files stored in their personal computers. This is because they can quickly find their previous work, figures and photographs from the list and reduce their writing time by reusing them. We have developed a system providing such lists on a web browser by using Ruby scripts and BibTEX bib files. This system is designed to generate an author's list of publications in various formats and to manage current manuscripts to provide an adequate return for keeping the database up to date.

1 Introduction

When writing a research paper, we often want to see material that we prepared and saved in the past including text, figures, and photographs. Thus, it is important to be able to retrieve any files that we might have saved in our personal computers (PC). If we can find the materials we need immediately, we can reuse part of them in the current manuscript and thus save time. This becomes much more important over time, because we will have a large number of stored documents and less time to devote to writing.

On the other hand, researchers must keep a list recording of their research publications because such lists are important when applying for jobs, promotion and funding. Thus, it makes sense to use the list as a portal to the files stored in a PC, namely, as a tool for retrieving past manuscripts. Since most of the files in researchers' PCs are closely related to items in their list of publications, they can easily recall any target file via their memory of related publication activities [1]. Thus, it is very convenient for researchers to have a personal publication list where each item has links to the related local files.

*A Japanese translation of this article is available at the homepage of the Asian Journal of TEX, http://ajt.ktug.kr.

The use of reference management software [7] appears to be a convenient way of compiling such a publication list. However, this kind of software is not necessarily designed for managing manuscripts in a PC. We therefore have to add certain functions to the software. In addition, we have to focus on the fact that we produce new files every day. Thus, the list should be updated as frequently as possible by its owner with as little effort as possible.

In this report, we describe such a manuscript writing support system based on BIBTEX [5] and Ruby [10], namely Ruby scripts manipulating BIBTEX bib files with a web user interface. Its design and implementation are reported in Sec. 2 and 3 respectively and Sec. 4 provides our conclusion.

2 Design of sustainably manageable publication list

2.1 Policy

The functions we must add to the reference management software are as follows:

- A function for registering additional information, such as path to manuscript files, into the database of the software
- A function for modifying the layout of the list to include links and/or buttons[1] for opening the manuscript files

In addition, we have to design an update procedure (1) that is simple enough not to disturb our daily work and (2) that gives us an adequate return for keeping it up to date.

To meet the second requirement, it is a good idea to update automatically and simultaneously both the list itself and other versions such as the lists published on researchers' homepages [1]. In addition, it is very effective to generate a deadline list for manuscripts in preparation. One of the present authors has been motivated by this list to keep his main list updated for six years. The following describes the procedure for constructing this deadline list.

1. When writing a new manuscript, register related information in the reference management software database, including its deadline, the path to the manuscript in the author's PC, and, if needed, web sites providing related information (style guide, conference venue, etc.).
2. Run the software to select the entries including the deadline and show them sorted by date. Each item includes the following links (see FIG. 1 right).
   (a) A link to launch a shell terminal (or a folder window) at the folder in which the manuscript is saved. This is the starting point for writing work.
   (b) A link to launch an application for modifying the database entry of the manuscript. This should be clicked before updating the list.
   (c) Links to access the web sites registered in the database.

In brief, each item in this list contains two starting points, one for manuscript writing and one for database management. After the manuscript is published, related

1. For simplicity, we use 'links' in the following text.

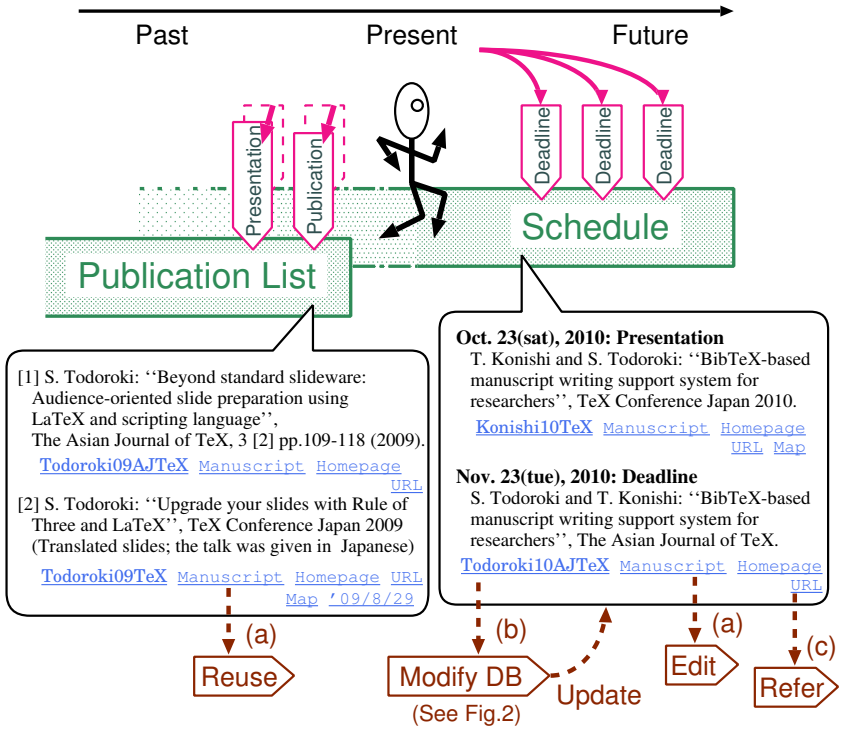

FIGURE 1. Relationship between the lists of deadlines and publications in this system.

bibliographic information is registered in the database, and the corresponding item is moved to the publication list (see FIG. 1 upper left) with these two starting points, which will be used later to access the manuscript and correct its database record.

These functions should be conveniently implemented to meet the requirement (1) mentioned above.

2.2 Choice of reference management software

We decided to use BIBTEX bib files for bibliographic data storage because of the followings reasons in addition to that we have been using BIBTEX for a long time.

Simple format: The text-based file format is easy to read and edit for human. Moreover, various parsers are available other than BIBTEX and some are written in light weight scripting languages including Ruby, Phython and Perl.

Extensibility: Since the number of reserved words is limited and user-defined fields are ignored by BIBTEX, the users can extend the fields without any compatibility violations.

Data portability: Practically, it is one of the standard formats for bibliographic information. Thus, the users can import data from other database through an appropriate conversion method.[2]

2. This means that it is possible to manage the data in one of other standard formats like XML. The choice depends on the user's specific requirements including skills, experience and preference.

```
@Unpublished{Todoroki10AJTeX,
  keywords = {paper, informatics, TeX},
  author =   {Shin-ichi Todoroki and Tomoya Konishi},
  title =    {\BibTeX -based manuscript writing support system for
              researchers},
  journal =  {The Asian Journal of \TeX},
  year =     2010,
  volume =   4,
  number =   2,
  pages =    {xx-xx},
  deadline = {Oct 23 2010},
  path =     {work/2010/10/AJTeX},
  homepage = {http://ajt.ktug.kr},
  url =      {http://oku.edu.mie-u.ac.jp/texconf10/proceedings.pdf},
}
```

FIGURE 2. An example of a bib file containing the fields added for this system (slanted letters).

3 Example implementation

We implemented the system using BIBTEX and Ruby based on the following three policies, which are described in detail in the following subsections.

- All the required information is recorded in BIBTEX bib files.
- All the entries in the bib files are stored in a Ruby object from which required information is extracted to compile a list.
- All the operations are performed through a web browser including the execution of applications for accessing local files.

3.1 Addition of new fields

We added some fields to the BIBTEX bib file to develop the functions of manuscript management and deadline list generation (see FIG. 2).

keywords A comma-separated list of keywords

deadline The deadline date for submissions

path A local path to a folder for manuscript writing

homepage The URL of a website providing information about the publication

url The URL of an online publication

These are named to avoid overriding the definitions used in the biblatex package [4].

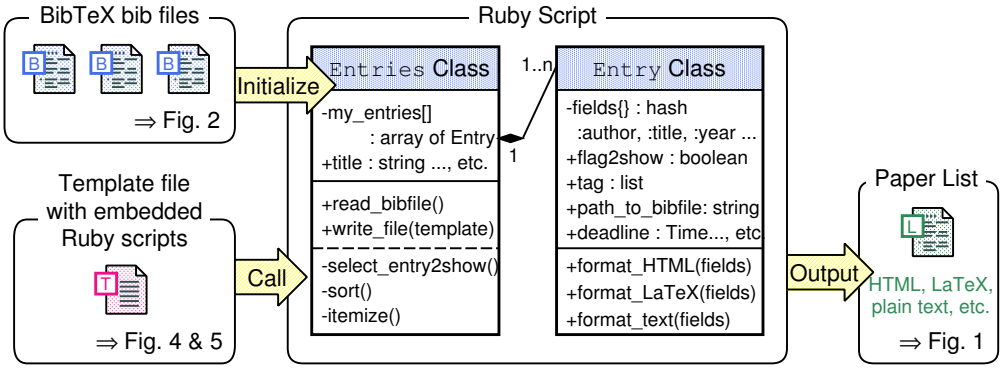

FIGURE 3. Procedure for generating a list through BIBTEX and Ruby.

3.2 Data processing through Ruby

We defined two classes in order to store bibliographic data in bib files (see FIG. 3 center). The `Entry` class is for storing a bibliographic entry in which the field values are stored as `Hash`. Several methods are defined in this class to output a single reference in various formats including HTML, LATEX and plain text. The `Entries` class stores an `Array` containing `Entry` instances. This class is initialized with specified bib files[3] and contains several methods for selecting, modifying and formatting the instances to generate a list.

Here we explain how to obtain a deadline list in HTML format by using these two classes (see FIG. 3 bottom). Prepare a template file as shown in FIG. 4, in which some Ruby scripts are embedded (see slanted sentences which describe the function of the code) to process an pregenerated `Entries` instance and obtain a list from it. In other words, we can control the structure of the list here including which entries are selected, how the entries are sorted, which fields are included in each item, and how each field is expressed.

In addition, certain special links are generated by certain `Entry` class methods; for example, a link to a manuscript file in the folder specified by the *`path`* field (see next subsection in detail), a link to open GoogleMaps at the place specified by the `address` field, and a link to open the list owner's blog-based research notebook[4] [2] on the date specified by the `year` and `month` fields (see the center bottom of FIG. 1).

Finally, this template is converted by eRuby, which is a converter for embedded Ruby code [8], into the final HTML form that appears as shown on the right in FIG. 1 through a web browser.

Using this procedure, we can generate various publication lists in several formats organized by subject, author, publication method, etc. through appropriate template files. FIGURE 5 shows another template for generating a publication list organized by subject in LATEX format.

3. bibtex_parser [6] is used for this initialization process.
4. An electronic notebook served by a user-installed blog server with an authentication function.

```
<html>
<head>List of Deadlines</head>
<body>
  <%=
    (Delete the entries if their deadlines are in the past)
    (Sort the entries by deadline)
    (Specify the fields to be shown (deadline, title, path, ...))
    (Output the list in HTML)
  %>
</body></html>
```

FIGURE 4. Example of a template file for generating a deadline list in HTML format. In practice, the sentences in slanted are replaced with embedded Ruby scripts.

```
\documentclass{article}
\begin{document}
...
\section{Journals}
  <%=
    (Delete the entries in the future)
    (Select the entries including a keyword, 'paper')
    (Sort the entries by date)
    (Specify the fields to be shown (author, title, journal, ...))
    (Output the list in LaTeX)
  %>
\section{Conferences}
  <%=
...
    (Select the entries including a keyword, 'conference')
...
```

FIGURE 5. Example of a template file for generating a publication list organized by publication method in LaTeX format. In practice, the sentences in slanted are replaced with embedded Ruby scripts.

3.3 Web interface

To launch certain applications from the web browser as described in FIG. 1, we must launch an HTTP server that responds to local requests to execute a Common Gateway Interface (CGI) script that launches a prescribed local application. FIGURE 6 shows a sample program of an HTTP server using WEBrick, a Ruby library providing simple HTTP web server services [9]. All the links for CGI execution in the publication lists are generated by one of the methods defined in the `Entry` class. In other words, we have to define these methods so that the HTTP server can accept the CGI requests embedded in the lists.

For example, the server launches a shell terminal (or a folder window) after receiving the argument of the *path* value (see lines 30 and 35 in FIG. 6), and opens a window

to edit the database entry after receiving two arguments, the name of the bib file and the line number at which the entry is registered in the file (see line 41 in FIG. 6).

In addition, we should be aware that the HTTP server shown in FIG. 6 can accept a third person's request. Authentication and a strict syntax check of the arguments should be implemented if the server is launched on a computer with multiple users.

4 Conclusion

We proposed a manuscript writing support system for researchers relating to a web-based personal publication list that helps them find their past manuscripts. It is useful both for manuscript writing and for researchers' self-promotion because their publication list is always updated and generated in various formats. Since the basic functions of this system, which are described in Sec. 2, can be implemented with other reference management software, we hope that our idea will prove useful beyond the TEX community. A subset of our software [3] will soon be made publicly available.

Acknowledgments

We are grateful to one of the reviewers for valuable discussion on Sec. 2.2.

References

1. Shin-ichi Todoroki, *Manuscript writing support system for researchers based on hypertext list of their achievements*, (2008), Translated from Ceramics Japan, **42** (2007), no. 7, 520–524. `http://pubman.mpdl.mpg.de/pubman/item/escidoc:28454`
2. Shin-ichi Todoroki, Tomoya Konishi, and Satoru Inoue, *Blog-based research notebook: personal informatics workbench for high-throughput experimentation*, Appl. Surface Sci. **252** (2006), no. 7, 2640–2645. `http://pubman.mpdl.mpg.de/pubman/item/escidoc:28315`
3. Tomoya Konishi, *MyBibList*. `http://www.anan-nct.ac.jp/material/mybiblist/`
4. Philipp Lehman, *biblatex*. `CTAN:macros/latex/exptl/biblatex/`
5. Oren Patashnik and Leslie Lamport, *BibTeX*. `CTAN:biblio/bibtex/`
6. Jeff Shantz, *bibtex_parser*. `http://rubygems.org/gems/bibtex_parser`
7. Wikipedia, *Comparison of reference management software*. `http://en.wikipedia.org/wiki/Comparison_of_reference_management_software`
8. _____, *eRuby*. `http://en.wikipedia.org/wiki/ERuby`
9. _____, *WEBrick*. `http://en.wikipedia.org/wiki/WEBrick`
10. *Ruby Programming Language*. `http://www.ruby-lang.org`

```
#!/usr/bin/env ruby
require 'webrick'
include WEBrick

s = HTTPServer.new(:BindAddress => '127.0.0.1', :Port => 8000,
                   :DocumentRoot => File.join(Dir::pwd, "public_html"))

# Response to the access via /cgi-bin/*
s.mount_proc('/cgi-bin') { |req, res|
  res.body =<<"EOS"
<html>
<body>
<script language="JavaScript" type="text/javascript"><!--
history.back()
// --></script>
<noscript>
Press the Return button of your browser.
<a href="#{req["referer"]}">Return</a>
</noscript>
</body></html>
EOS
  res.header["Content-Type"] =  "text/html"

  # Applications to be launched
  key,val = req.unparsed_uri.gsub(/^[^?]*\?cmd=/,"").split(/:/,2)
  case key
  when "term"
    val = File.expand_path("~") + "/" + val
    if File.exist?(val) then
      system("gnome-terminal --working-directory=#{val}")
    end
  when "gui"
    val = File.expand_path("~") + "/" + val
    if File.exist?(val) then
      system("nautilus #{val}")
    end
  when "edit"
    file_name , line_num = val.split(/:/)
    if File.exist?(file_name) then
      system("env XMODIFIERS=\"@im=NONE\" " +
             "emacs +#{line_num} #{file_name} 2> /dev/null&")
    end
  end
}
Signal.trap(:INT){ s.shutdown }
s.start
```

FIGURE 6. Ruby-based HTTP server that launches predetermined applications via CGI, `http://localhost:8000/cgi-bin/runcmd.cgi?cmd=`*key*`:`*val* (see line 25).