LaTeX Export In Emacs Org-Mode

Boyun Tang

May 15, 2011

Contents

1 引子

1.1 论学习

人总是在反思中前进的，我反思我的人生，事实上有许多事是做了也白做的无用功而已，有意的，无意的，更多时候是被逼迫的。我个人觉得我做的最有价值的抉择之一就是选择了学习Emacs。很多人可能会觉得花时间去整这么个破文本编辑器不太值得———时间应该留给那些更"重要"的事。对此我倒是抱着不同的态度，因为学习并不是像哥伦布去探索新大陆那样———不停地去探索完全未知的世界，更为自然的过程是我们在有意无意地比较新体验和自己已知的经历，如果两者有某种联系，能产生共鸣的话，那么这个学习曲线将变得相当平滑，反之则如异体移植一般，这个新的知识点将会被慢慢遗忘———因为你无法巩固它。从这个角度来看的话，学习本身是个悖论：

懂得越多则学得越快，反之亦然。

学习就像拼一大块拼图，你手中的碎片越多，拼起来也就越容易，Emacs对于我而言就是这么个碎片之间的胶合剂，Emacs可是能随着自身升级而提升的宝具哦，你有多强Emacs在你手中就可以变得多强。

废话貌似扯太多了，我得趁热把我鼓捣的心得记下来，今天的主角是Emacs和Org-Mode。

2 Org-Mode

Org-Mode是 Emacs 中一个非常强大的主模式，嗯，可以很随意的写博客、记笔记、记实验记录、计划行程、书写Literate Programming代码，当然不是随便涂鸦哟，可以自定义各种格式化输出，呃，LaTeX够牛逼直接在Org-Mode里写专业论文导出成 PDF 也是完全没问题的！

2.1 教程

关于 Org-Mode 体验许多教程，官方的[1]，中文的[2]，嗯，我就不写料，偶只记自己的心得与笔记。‿

2.2 快捷输入之小抄

输入 <e 和 TAB, 得到

```
#+begin_example
#+end_example
```

输入 <s 和 TAB, 得到

```
#+begin_src
#+end_src
```

咳，相应的还有：

```
("s" "#+begin_src ?\n\n#+end_src" "<src lang=\"?\">\n\n</src>")
("e" "#+begin_example\n?\n#+end_example" "<example>\n?\n</example>")
("q" "#+begin_quote\n?\n#+end_quote" "<quote>\n?\n</quote>")
("v" "#+begin_verse\n?\n#+end_verse" "<verse>\n?\n/verse>")
("c" "#+begin_center\n?\n#+end_center" "<center>\n?\n/center>")
("l" "#+begin_latex\n?\n#+end_latex" "<literal style=\"latex\">\n?\n</literal>")
("L" "#+latex: " "<literal style=\"latex\">?</literal>")
("h" "#+begin_html\n?\n#+end_html" "<literal style=\"html\">\n?\n</literal>")
("H" "#+html: " "<literal style=\"html\">?</literal>")
("a" "#+begin_ascii\n?\n#+end_ascii")
("A" "#+ascii: ")
("i" "#+include %file ?" "<include file=%file markup=\"?\">"))
```

这些都是定义在 org-structure-template-alist 中的变量。记在这里，先前也算翻了不少文档了，居然没发现这个。囧，爹多打了多少个字母啊 ~ ~

3 LaTeX

这东东是将被.doc 格式标准奴役地穷苦大众解放出来地神器，嗯。咳，要与吾圣战三百回合地，可以先去这里[3]准备弹药 ~ ~ !
下面请观赏：圣徒与撒旦 ~ ~ !! 囧

http://libai.math.ncu.edu.tw/bcc16/pool/image/knuth-paint.jpg
http://www.bbspot.com/Images/homepage/gates.jpg

3.1 LaTeX 中的代码格式化

格式化论文什么的，就跟写八股文一样，没太大意思，况且很多杂志都有自己的模板，这就更无趣了，你不照着做都不行，既没创意又没难度，没有美感的事情不能做啊 ~ ~ 囧。Beamer 和 Tikz 的 Manual 都太厚，看起来比较花时间（Beamer 的配置挺有意思的，有机会再写）不过我倒是有把代码格式化的包包都有看过。其实格式化代码有两种途径：

定义关键词列表 代表Listings宏包

使用外部词法剖析器 代表为 mint 宏包

3.1.1 Listings宏包

优点 目前来看，Listings宏包的优势在于交叉引用比较强，因为可以通过 escapeinside 设置代码内部的转义符来设定定义行号标签。

缺点 缺点在于，由于并非使用语法剖析器，而只是预定义关键词，所以很可能出现错误的高亮。

[1] 最好的学习方法就是自己查文档哦！‿
[2] 从懂中文的前辈那里偷师比较容易！‿
[3] The Beauty of LaTeX，多好的文章啊，等等，你不会翻墙?? 你，你不会翻墙，你上啥网啊?

3.1.2 mint 宏包

优点 外部调用 python 脚本，类型的剖析相对准确。(说实话这货比起 {{haskell}} 中lhs2TeX + lhs2TeX-hl [4]，就是个战斗力只有 5 的渣渣 ~ ~ 囧)

缺点 文档实在太短，貌似自定义 theme 的接口比较弱。

3.1.3 Org-Mode中的LaTeX导出

Org-Mode中的 html 导出相对简单，不需要重度的配置即可，不过LaTeX导出就不同了，既要定义导出类，而且你必须忍受作为一个中国人的种种不便：

1. 字体问题一直是老大难问题，由字体问题直接引出你该使用哪种TeX 的问题，接着就是流程问题.....我个人选了XeTeX
2. 选了那种TeX 决定了之后对于宏包的取舍，各种载入顺序、冲突等等，做为中国人.....还是得忍，然后就是调试再调试.....

最终在格式化文章以及代码的问题上，我选择了Listings，相较而言文档比较详细、对 Org-Mode 的支持更好些[5]，目前只有Listings支持同时对标题和行数的引用，并且可以一直在Org-Mode下就完成所有工作（不需要额外打开.tex 文件做编辑工作，一切自动生成，且高度可定制）。不过需要注意的是对于 PDF 的导出时，要在代码框外对代码框加标题和标注需要做额外的工作 [5]，即在 $\text{begin}_{\text{src}}$ 环境外额外再使用一个 $\text{begin}_{\text{listing}}$ 环境。咳，对这个的原始文档注释，我没有找到 (测试有效)。。。也许是个原作者才清楚的小 hack。

```
... ....
... \ref{lst:src_blk}
...
\begin{listing}
\caption{The caption.}\label{lst:src_blk}
 <source block>
\end{listing}
... ...
... \autoref{lst:src_blk}
... ...
```

4 配置

4.1 一些思路

参考了不少资源，listings 设置参考了这里[6]，article class 设置、color theme 参考了这里[7]。

本身的配置没太大技术含量，过程就是尝试到不出错能完整输出为止。囧，beamer 类还没动过，没看过文档的东西不太好下手啊，beamer 属于必须精通的那类，咳，必须的，有时间一定要搞起来。

cn-article 类主要特点是集数学公式、符号排版以及代码格式化，算法伪代码书写于一体，同时兼顾了普通中文文档的输出（没预设太多中文字体，咳，因为看得惯的字体真少啊 ~ ~ ◡̈），再润色下地话，咳，直接Org-Mode直接导出篇学术论文都么问题啊（有时间等偶得把Ebib设置好，那就学术一条龙了好吗！），妥妥儿的。

目前还在订制 org 的模板，我觉得 org 的命令和技巧太多，学不过来啊，在 check 了Ess mode文档之后，偶觉得配合Org Babel，虾米RStudio、Matlab-GUI，统统可以消失了好吗，一个Emacs搞定天下不是梦好么。咳，跑题了，关键在于东西太多，知识点太杂，我估摸着把这些玩意儿文档整理下，统统丢进 org 的模板里，然后借助Folding Mode，在打开 org 文档时把这些不常用的选项和文档进行折叠不显示就可以了，需要的时候 check 下。

[4] n 久前，在 Haskell source code layout 上做的一些尝试
[5] 关于如何在 Org source block 外加入LaTeX CAPTION 的讨论，老规矩了，翻 ~ ◡̈
[6] jevopi's little blog，嗯，无意中发现的，这年头评价一个技术博客的标准之一就是：是不是需要翻墙才能看？ ◡̈
[7] Emacs-Fu，从这里学了不少啊，老规矩，不需要翻墙的 blog 不是好 blog ~ ~ ◡̈

My Org-Mode Settings

```
;;; my-org-settings.el --- My Org-Mode Settings

;; Copyright (C) 2011 Boyun Tang

;; Author: Boyun Tang <tangboyun@hotmail.com>
;; Keywords:

;; This program is free software; you can redistribute it and/or modify
;; it under the terms of the GNU General Public License as published by
;; the Free Software Foundation, either version 3 of the License, or
;; (at your option) any later version.

;; This program is distributed in the hope that it will be useful,
;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
;; GNU General Public License for more details.

;; You should have received a copy of the GNU General Public License
;; along with this program. If not, see <http://www.gnu.org/licenses/>.

;;; Commentary:

;;

;;; Code:
(require 'org-install)
(require 'ob-ditaa)
(require 'google-weather)
(require 'org-google-weather)
(require 'org-latex)

;; Folding Mode 相关
(load "folding" 'nomessage 'noerror)
(folding-mode-add-find-file-hook)
(folding-add-to-marks-list 'org-mode "# {{{" "# }}}" nil t)
(add-hook 'org-mode-hook 'folding-mode)



;; 使用宏包格式化源代码Listings只是把代码框用(环境框起来,还需要额外的设置listing)
(setq org-export-latex-listings t)



;; Options for \lset (commandreference to listing Manual)
(setq org-export-latex-listings-options
      '(
        ("basicstyle" "\\color{foreground}\\small\\mono") ; 源代码字体样式
        ("keywordstyle" "\\color{function}\\bfseries\\small\\mono") ; 关键词字体样式
        ("identifierstyle" "\\color{doc}\\small\\mono")
        ("commentstyle" "\\color{comment}\\small\\itshape") ; 批注样式
        ("stringstyle" "\\color{string}\\small")     ; 字符串样式
        ("showstringspaces" "false")                 ; 字符串空格显示
        ("numbers" "left")                           ; 行号显示
        ("numberstyle" "\\color{preprocess}")        ; 行号样式
        ("stepnumber" "1")                           ; 行号递增
        ("backgroundcolor" "\\color{background}")    ; 代码框背景色
        ("tabsize" "4")                              ; 等效空格数TAB
        ("captionpos" "t")                           ; 标题位置top or buttom(t|b)
        ("breaklines" "true")                        ; 自动断行
        ("breakatwhitespace" "true")                 ; 只在空格分行
        ("showspaces" "false")                       ; 显示空格
        ("columns" "flexible")                       ; 列样式
```

```
64          ("frame" "single")                                  ; 代码框:阴影盒
65          ("frameround" "tttt")                               ; 代码框:圆角
66          ("framesep" "0pt")
67          ("framerule" "8pt")
68          ("rulecolor" "\\color{background}")
69          ("fillcolor" "\\color{white}")
70          ("rulesepcolor" "\\color{comdil}")
71          ("framexleftmargin" "10mm")
72          ))
73
74
75
76  ;; 使用一步生成xelatexPDF
77  (setq org-latex-to-pdf-process
78    '("xelatex -interaction nonstopmode %f"
79      "xelatex -interaction nonstopmode %f"))
80
81
82
83  ;; 默认主模式为org-mode
84  (setq default-major-mode 'org-mode)
85
86
87
88  ;; Make Org use ido-completing-read for most of its completing prompts.
89  (setq org-completion-use-ido t)
90
91
92
93  ;; 执行免应答(codeEval code without )confirm
94  (setq org-confirm-babel-evaluate nil)
95
96
97
98  ;; Auctex
99  (setq TeX-auto-save t)
100 (setq TeX-parse-self t)
101 (setq-default TeX-master nil)
102
103
104
105 ;; diaa path 考虑换成DitaaEps
106 (setq org-ditaa-jar-path "~/.emacs.d/ditaa0_9.jar")
107
108
109
110 ;; 各种语言支持Babel
111 (org-babel-do-load-languages
112  'org-babel-load-languages
113  '((R . t)
114    (emacs-lisp . t)
115    (matlab . t)
116    (C . t)
117    (perl . t)
118    (sh . t)
119    (ditaa . t)
120    (python . t)
121    (haskell . t)
122    (dot . t)
123    (latex . t)
124    (js . t)
125    ))
126
127
128
```

```
129
130 ;; REFTEX
131 (defun org-mode-article-modes ()
132   (reftex-mode t)
133   (and (buffer-file-name)
134        (file-exists-p (buffer-file-name))
135        (reftex-parse-all)))
136 (add-hook 'org-mode-hook
137         (lambda ()
138           (if (member "REFTEX" org-todo-keywords-1)
139              (org-mode-article-modes))))
140
141
142
143
144 ;; Latex Export
145 (unless (boundp 'org-export-latex-classes)
146   (setq org-export-latex-classes nil))
147
148 ;; 'my-org-article' for export org documents to the LaTex 'article', using
149 ;; XeTeX and some fancy fonts; requires XeTeX (see org-latex-to-pdf-process)
150 (add-to-list 'org-export-latex-classes
151            '("cn-article"
152 "\\documentclass[10pt,a4paper]{article}
153 \\usepackage{graphicx}
154 \\usepackage{xcolor}
155 \\usepackage{xeCJK}
156 \\usepackage{lmodern}
157 \\usepackage{verbatim}
158 \\usepackage{fixltx2e}
159 \\usepackage{longtable}
160 \\usepackage{float}
161 \\usepackage{tikz}
162 \\usepackage{wrapfig}
163 \\usepackage{soul}
164 \\usepackage{textcomp}
165 \\usepackage{listings}
166 \\usepackage{geometry}
167 \\usepackage{algorithm}
168 \\usepackage{algorithmic}
169 \\usepackage{marvosym}
170 \\usepackage{wasysym}
171 \\usepackage{latexsym}
172 \\usepackage{natbib}
173 \\usepackage{fancyhdr}
174 \\usepackage[xetex,colorlinks=true,CJKbookmarks=true,
175           linkcolor=blue,menucolor=blue]{hyperref}
176 \\usepackage{fontspec,xunicode,xltxtra}
177 \\setmainfont{Times New Roman}
178 \\newcommand\\fontnamemono{DejaVu Sans YuanTi Mono}
179 \\newfontinstance\\MONO{\\fontnamemono}
180 \\newcommand{\\mono}[1]{{\\MONO #1}}
181 \\setCJKmainfont[Scale=0.9]{DejaVu Sans YuanTi Condensed}
182 \\setCJKmonofont[Scale=0.9]{DejaVu Sans YuanTi Mono}
183 \\setCJKfamilyfont{mono}{YaHei Consolas Hybrid}
184 \\hypersetup{unicode=true}
185 \\geometry{a4paper, textwidth=6.5in, textheight=10in,
186         marginparsep=7pt, marginparwidth=.6in}
187 \\definecolor{foreground}{RGB}{220,220,204}
188 \\definecolor{background}{RGB}{62,62,62}
189 \\definecolor{preprocess}{RGB}{250,187,249}
190 \\definecolor{var}{RGB}{239,224,174}
191 \\definecolor{string}{RGB}{154,150,230}
192 \\definecolor{type}{RGB}{225,225,116}
193 \\definecolor{function}{RGB}{140,206,211}
```

```
\\definecolor{keyword}{RGB}{239,224,174}
\\definecolor{comment}{RGB}{180,98,4}
\\definecolor{doc}{RGB}{175,215,175}
\\definecolor{comdil}{RGB}{111,128,111}
\\definecolor{constant}{RGB}{220,162,170}
\\definecolor{buildin}{RGB}{127,159,127}
\\punctstyle{kaiming}
\\title{}
\\fancyfoot[C]{\\bfseries\\thepage}
\\chead{\\MakeUppercase\\sectionmark}
\\pagestyle{fancy}
\\tolerance=1000
[NO-DEFAULT-PACKAGES]
[NO-PACKAGES]"
           ("\\section{%s}" . "\\section*{%s}")
           ("\\subsection{%s}" . "\\subsection*{%s}")
           ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
           ("\\paragraph{%s}" . "\\paragraph*{%s}")
           ("\\subparagraph{%s}" . "\\subparagraph*{%s}")))


;; allow for export=>beamer by placing
;; #+LaTeX_CLASS: beamer in org files

(add-to-list 'org-export-latex-classes
  ;; beamer class, for presentations
  '("beamer"
"\\documentclass[11pt,professionalfonts]{beamer}

\\mode<{{{beamermode}}}>
\\usetheme{{{{beamertheme}}}}
\\usecolortheme{{{{beamercolortheme}}}}

\\beamertemplateballitem
\\setbeameroption{show notes}
\\usepackage{graphicx}
\\usepackage{tikz}
\\usepackage{xcolor}
\\usepackage{xeCJK}
\\usepackage{amsmath}
\\usepackage{lmodern}
\\usepackage{fontspec,xunicode,xltxtra}
\\usepackage{polyglossia}
\\setmainfont{Times New Roman}
\\setCJKmainfont{DejaVu Sans YuanTi}
\\setCJKmonofont{DejaVu Sans YuanTi Mono}
\\usepackage{verbatim}
\\usepackage{listings}
\\institute{{{{beamerinstitute}}}}
\\subject{{{{beamersubject}}}}"
    ("\\section{%s}" . "\\section*{%s}")
    ("\\begin{frame}[fragile]\\frametitle{%s}"
     "\\end{frame}"
     "\\begin{frame}[fragile]\\frametitle{%s}"
     "\\end{frame}")))

  ;; letter class, for formal letters
;; (add-to-list 'org-export-latex-classes
;;          '("letter"
;;            "\\documentclass[11pt]{letter}\\usepackage{xcolor}"

;;            ("\\section{%s}" . "\\section*{%s}")
;;            ("\\subsection{%s}" . "\\subsection*{%s}")
;;            ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
;;            ("\\paragraph{%s}" . "\\paragraph*{%s}")
```

```
;;            ("\\subparagraph{%s}" . "\\subparagraph*{%s}")))

;; (add-to-list 'org-export-latex-classes
;;           '("book"
;;             "\\documentclass{book}"
;;             ("\\part{%s}" . "\\part*{%s}")
;;             ("\\chapter{%s}" . "\\chapter*{%s}")
;;             ("\\section{%s}" . "\\section*{%s}")
;;             ("\\subsection{%s}" . "\\subsection*{%s}")
;;             ("\\subsubsection{%s}" . "\\subsubsection*{%s}")))
;; (add-to-list 'org-export-latex-classes
;;           '("koma-article"
;;             "\\documentclass{scrartcl}"
;;             ("\\section{%s}" . "\\section*{%s}")
;;             ("\\subsection{%s}" . "\\subsection*{%s}")
;;             ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
;;             ("\\paragraph{%s}" . "\\paragraph*{%s}")
;;             ("\\subparagraph{%s}" . "\\subparagraph*{%s}")))


(setq ps-paper-type 'a4
      ps-font-size 16.0
      ps-print-header nil
      ps-landscape-mode nil)



(provide 'my-org-settings)
;;; my-org-settings.el ends here
```

5 效果

一键输出哦，任何搞完 org 文件再打开 tex 文件的行为统统都是不可原谅的好吗！！

5.1 代码输出赏析

这是前几日Top Language上Liu Xinyu童鞋撰写的 Haskell AVL 树实现代码，请有意的童鞋往死里点击该 Email..... 这么好的代码，不好好格式化下对不起苍天大地的.....

AVLTree in Haskell

```haskell

-- AVLTree.hs
-- Copyright (C) 2010 Liu Xinyu (liuxinyu95@gmail.com)
--
-- This program is free software: you can redistribute it and/or modify
-- it under the terms of the GNU General Public License as published by
-- the Free Software Foundation, either version 3 of the License, or
-- (at your option) any later version.
--
-- This program is distributed in the hope that it will be useful,
-- but WITHOUT ANY WARRANTY; without even the implied warranty of
-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
-- GNU General Public License for more details.
--
-- You should have received a copy of the GNU General Public License
-- along with this program. If not, see <http://www.gnu.org/licenses/>.

module AVLTree where

import Test.QuickCheck
import qualified Data.List as L -- for verification purpose only

data AVLTree a = Empty
               | Br (AVLTree a) a (AVLTree a) Int

insert::(Ord a)=>AVLTree a -> a -> AVLTree a
insert t x = fst $ ins t where
   -- result of ins is a pair (t, d), t: tree, d: increment of height
   ins Empty = (Br Empty x Empty 0, 1)
   ins (Br l k r d)
      | x < k  = node (ins l) k (r, 0) d
      | x == k = (Br l k r d, 0)
      | otherwise = node (l, 0) k (ins r) d

-- params: (left, increment on left) key (right, increment on right)
node::(AVLTree a, Int) -> a -> (AVLTree a, Int) -> Int -> (AVLTree a, Int)
node (l, dl) k (r, dr) d = balance (Br l k r d', delta)
   where
     d' = d + dr - dl
     delta = deltaH d d' dl dr

-- delta(Height) = max(|R'|, |L'|) - max (|R|, |L|)
-- where we denote height(R) as |R|
deltaH :: Int -> Int -> Int -> Int -> Int
deltaH d d' dl dr
      | d >=0 && d' >=0 = dr
      | d <=0 && d' >=0 = d+dr
      | d >=0 && d' <=0 = dl - d
      | otherwise = dl

balance :: (AVLTree a, Int) -> (AVLTree a, Int)
balance (Br (Br (Br a x b dx) y c (-1)) z d (-2), _) = (Br (Br a x b dx) y (Br c z d 0) 0, 0)
balance (Br a x (Br b y (Br c z d dz) 1) 2, _) = (Br (Br a x b 0) y (Br c z d dz) 0, 0)
balance (Br (Br a x (Br b y c dy) 1) z d (-2), _) = (Br (Br a x b dx') y (Br c z d dz') 0, 0)
   where
     dx' = if dy == 1 then -1 else 0
     dz' = if dy == -1 then 1 else 0
balance (Br a x (Br (Br b y c dy) z d (-1)) 2, _) = (Br (Br a x b dx') y (Br c z d dz') 0, 0)
   where
```

```haskell
    dx' = if dy == 1 then -1 else 0
    dz' = if dy == -1 then 1 else 0
balance (t, d) = (t, d)

-- check if a AVLTree is valid
isAVL :: (AVLTree a) -> Bool
isAVL Empty = True
isAVL (Br l _ r d) = and [isAVL l, isAVL r, d == (height r - height l), abs d <= 1]

height :: (AVLTree a) -> Int
height Empty = 0
height (Br l _ r _) = 1 + max (height l) (height r)

checkDelta :: (AVLTree a) -> Bool
checkDelta Empty = True
checkDelta (Br l _ r d) = and [checkDelta l, checkDelta r, d == (height r - height l)]

-- Auxiliary functions to build tree from a list, as same as BST

fromList::(Ord a)=>[a] -> AVLTree a
fromList = foldl insert Empty

toList :: (AVLTree a) -> [a]
toList Empty = []
toList (Br l k r _) = toList l ++ [k] ++ toList r

-- test
prop_bst :: (Ord a, Num a) => [a] -> Bool
prop_bst xs = (L.sort $ L.nub xs) == (toList $ fromList xs)

prop_delta :: (Ord a, Num a) => [a] -> Bool
prop_delta = checkDelta . fromList. L.nub

prop_avl :: (Ord a, Num a) => [a] -> Bool
prop_avl = isAVL . fromList . L.nub

-- Helper function for pretty printing
instance Show a => Show (AVLTree a) where
   show Empty = "."
   show (Br l k r d) = "(" ++ show l ++ " " ++
                       show k ++ ":[" ++ show d ++ "] " ++
                       show r ++ ")"
```

5.2 本文以及上述代码的 PDF 文档下载