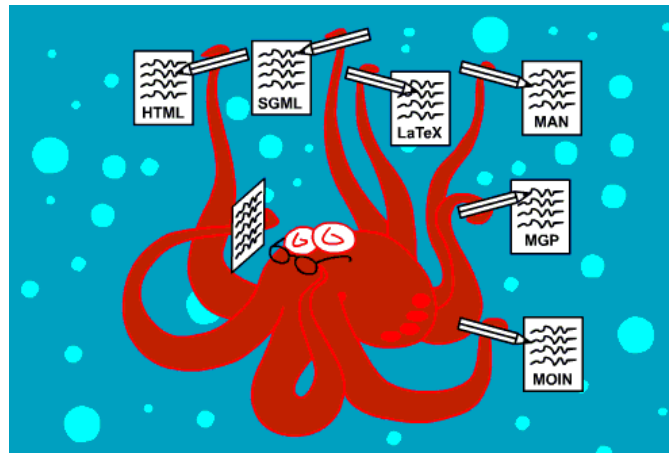


txt2tags 用户指南

原著: Aurélio Marinho Jargas <verde (at) aurelio.net>
翻译: Chris Leng <rookie999.ly (at) gmail.com>



目录

关于本文	4
第一章 初识txt2tags	5
1.1 第一个可能的问题	5
1.1.1 txt2tags是什么?	5
1.1.2 为什么使用txt2tags?	6
1.1.3 相较于其他工具, txt2tags有哪些优点?	6
1.1.4 我需要为此付费吗?	7
1.2 支持的文档结构	7
1.3 支持的目标文件格式	8
1.4 目标文件结构支持一览	11
1.5 三种用户接口: GUI、网页和命令行	11
1.5.1 基于Tk的图形用户接口	12
1.5.2 在线方式	13
1.5.3 命令行	13
第二章 如何开始?	16
2.1 下载、安装Python	16
2.2 下载txt2tags	16
2.3 安装txt2tags	16
2.3.1 为Windows用户专门打包的安装程序	17
2.4 安装编辑器的语法高亮文件	17
第三章 第一个文档	19
3.1 先利其器	19

目录	2
3.2 编写文档头部	19
3.3 第一次转换——图形界面操作	20
3.4 第一次转换——命令行方式	21
3.5 查看结果	21
3.6 编写文档体	22
第四章 掌握txt2tags的结构	24
4.1 文档的各个部分	24
4.1.1 完整的示例	25
4.2 头部域	25
4.3 设置域	26
4.4 主体域	27
4.5 设定	27
4.6 命令行选项	28
4.7 用户配置文件（RC文件）	28
4.8 配置的优先级和加载顺序	29
4.9 %!include 指令	30
4.10 %!includeconf 指令	31
第五章 掌握标记	32
5.1 头部	33
5.2 标题与编号的标题	34
5.3 段落	34
5.4 注释	35
5.5 粗体，斜体，下划线和删除线	35
5.6 等宽字体	36
5.7 完全引用行和完全引用域	36
5.8 隔离行和粗隔离行	37
5.9 链接和命名链接	38
5.10 引用	38
5.11 列表，编号列表和定义列表	39
5.12 图片	39

目录	3
5.13 表格	40
5.14 原始文，原文行和原文域	41
第六章 宏	43
6.1 %%date	44
6.2 %%mtime	45
6.3 %%infile	45
6.4 %%outfile	46
6.5 %%toc	46
第七章 设定	48
7.1 %!Target	49
7.2 %!Options	49
7.3 %!Encoding	50
7.4 %!PreProc	50
7.5 %!PostProc	51
7.6 %!Style	51
7.7 针对特定的目标格式进行设定	52
7.8 关于PreProc和PostProc过滤器的一些细节	52
第八章 黑魔法	54
8.1 使用%!PostProc插入多行文本（比如CSS规则）	54
8.2 使用%!PreProc创建“特定目标类型”的内容	55
8.3 使用%!PreProc改变txt2tags标记	55
第九章 软件历史	57
9.1 1999年1月：史前历史	57
9.2 1999年6月：依然是史前历史	57
9.3 2000年八月：不再是史前历史了	58
9.4 2001年5月：Python化和多目标格式的想法	59
9.5 2001年7月：0.x系列：txt2tags的处女秀（公开发布）	59
9.6 2002年9月：1.x系列：成长	60
9.7 2004月7月：2.x系列：成熟	60

关于本文

本文根据txt2tags（当前版本为2.5）的用户指南英文版翻译而来，读者可通过以下途径获得用户指南的英文原版：

- 用户指南官方网址：<http://txt2tags.sf.net/userguide>
- 用户指南的PDF版本：<http://txt2tags.sf.net/userguide/userguide.pdf>

本译文依照[创作共用约定（署名-非商业性使用-相同方式共享）3.0](#)发布。

第一章 初识txt2tags

本章为概述，介绍txt2tags程序的用途和特色。

1.1 第一个可能的问题

1.1.1 txt2tags是什么？

txt2tags是一个文本格式化和转换工具，能够将带有特定标记的纯文本文件转换为支持的任意一种目标格式，包括：

- HTML文档
- XHTML文档
- SGML文档
- LaTeX源文件
- Unix手册页（man page）
- MagicPoint演示文稿
- Wikipedia维基百科页面
- Google Wiki 页面
- DokuWiki页面
- MoinMoin页面
- PageMaker 6.0 文档
- 纯文本文件

1.1.2 为什么使用txt2tags?

你会发现txt2tags十分有用，如果你：

- 需要将文档以不同格式发布
- 需要以不同格式维护文档的更新
- 撰写技术文档或者手册
- 需要用你不熟悉的某种格式撰写文档
- 对于某种文档格式缺少一个专门的编辑器
- 想用一个简单的文本编辑器维护和更新你的文档

而最大的诱人之处在于：

- 节省时间，专注于内容，抛开格式

1.1.3 相较于其他工具，txt2tags有哪些优点？

源文件可读性好	txt2tags的标记非常简单，使得源文件几乎是自然语言。
目标文件可读性好	与源文件一样，生成的目标文件带有缩进和较短的行长度，可读性好。
一致的标记	txt2tags的标记是独特的，足以适应各种文档而不至于与文档内容产生混淆。
一致的规则	每种标记与其对应的规则是一一对应的，不存在“例外”或“特殊情形”
结构简单	所有支持的格式都是简单的，不带有附加选项或者复杂的修饰。标记只是标记而已，没有任何选项。
易于学习	由于简单的标记和可读的源文件，txt2tags的学习曲线非常的“用户友好”。
漂亮的示例	无论用txt2tags编写简单还是复杂的文档，包含在软件包中的 示例文件 都给出了生动实用的例子。
有用的工具	包含于软件包中的 语法规则文件 帮助你写出没有语法错误的文档（适用于vim, emacs, nano和kate）。
三种用户接口	用Tk编写的 图形接口 ，可供远程和内部网使用的 网页接口 ，以及供高级用户和脚本使用的 命令行接口 。

接下页...

脚本	有经验的用户能够通过完全的命令行模式进行自动任务，以及对转换后的文件进行后编辑。
下载、运行和跨平台	txt2tags只是一个简单的Python脚本，无需编译或下载额外的模块，因此在各种*NIX、Linux、Windows和Macintosh主机上均能完美运行。
时常更新	活跃的邮件列表有用户提供修正和改进。作者本人在工作和业余也在使用txt2tags，因此开发不会无故终止。

1.1.4 我需要为此付费吗？

当然不需要！

txt2tags在GPL许可证下发布，并且完全免费。

1.2 支持的文档结构

以下列出了txt2tags支持的文档结构。

- 头部（文档标题、作者、日期）
- 章节标题（带有/不带编号）
- 段落
- 字体美化
 - 粗体
 - 斜体
 - 下划线
 - 删除线
- 等宽字体（完全引用）
 - 段落内的等宽字体
 - 等宽的行

- 等宽的区域（多行）
- 引用
- 链接
 - URL/Internet链接
 - e-mail链接
 - 本地链接
 - 命名的链接
- 列表
 - 以符号引导的列表
 - 编号的列表
 - 定义列表
- 水平分隔线
- 图片（智能对齐）
- 表格（带/不带边框，智能对齐，列贯穿）
- 针对原文域的特殊标记（不解析）
- 对当前日期的特殊宏定义（能灵活的改变格式）
- 注释（作者备注；TODO；FIXME）

1.3 支持的目标文件格式

HTML HTML是众所周知的格式（提示：Internet）。

Txt2tags能生成干净漂亮，源文件可读的HTML文档，不使用JavaScript、框架或是其他无效的、简洁的技术文档所不需要的格式化技术，但是可以使用独立的CSS文件。txt2tags生成的代码符合“*HTML 4.0 Transitional*”标准。

自2.0版起，txt2tags生成的HTML代码100%通过[w3c validator](#)的校验。

XHTML 新一代的HTML，具有更严格的语法——所有的标记都必须闭合，这使得代码更易理解和解析。大体上来说，可以将其看做HTML。Txt2tags生成的代码符合“*XHTML 1.0 Transitional*”标准。

自2.0版起，txt2tags生成的XHTML代码100%通过w3c validator的校验。

SGML 格式转换软件SGML工具的通用格式。单一的SGML文件能转换为HTML、PDF、PS、info、LaTeX、Lyx、rtf以及XML格式的文档。SGML2* 工具还能自动生成目录表，以及按照章节将文件分割为子页（sgml2html）。

Txt2tags生成的SGML文件为Linux系统文档格式，通过SGML2*工具能够直接转换，而毋需任何额外的目录文件或是SGML那些恼人的要求。

LATEX 超出你想象的强大格式，学术论文的通用格式。用它能生成整本书、复杂的公式以及任何复杂的文字格式，然而，如果你打算手写那些标记，就准备掉头发吧……

Txt2tags将为你处理所有的细节和例外情形，完成能够直接使用的LaTeX文档，你只需考虑文字内容。

LOUT 与LaTeX非常相似，但是以“@”代替“\”作为命令引导符，同时尽量避免使用括号。它“一切皆对象”的处理方式使得标记更加健壮。

Txt2tags能生成可以直接使用的文件，通过“lout”命令即可转换为PDF或PS文件。

MAN UNIX手册页。

文档格式产生消亡，来来去去，UNIX手册页却存活多年，始终坚挺。

有各种工具可以生成手册页文档，但是txt2tags自有其优势：一个源文件，多种产出物。因此，手册页的内容还能转换为HTML、MagicPoint演示文稿，等等。

MGP MagicPoint是一个非常方便的幻灯片工具（就如同Microsoft PowerPoint）。它使用标记语言来定义每一屏显示，因此你可以使用vi/emacs/记事本这样的文本编辑器来完成复杂的幻灯片。

Txt2tags为你完成所有被ISO-8859标准所支持的字体和外观的定义，生成的.mgp文件能够直接使用。

HOTSPOT 1: txt2tags在生成的.mgp文件中使用XFree86 Type1字体！因此你的幻灯片毋需额外配合TrueType字体。

HOTSPOT 2: 字体颜色的定义简洁，即使在只有简陋调色板的系统中（比如`startx -- -bpp 8`），幻灯片依然美观。

你需要做的只是：转换，然后使用。毋需任何修复或是其他额外的工作。

WIKI 想必你一定听说过[维基百科\(Wikipedia\)](#)吧？现在你不必另外学习一种新的标记语法，`txt2tags`将为你把标记文本转换为维基百科的格式。

GWIKI 使你能轻松的将项目文档粘贴到[Google Code Wiki](#)。

DOKU [DokuWiki](#)是一种遵从标准并且易于使用的Wiki，被设计来创建各种类型的文档，其目标群体包括开发团队、工作组以及小型公司。其语法简洁而强大，确保数据文件脱离Wiki仍然保有可读性，同时简化结构化文本的创建，所有的数据都以纯文本文件格式存储——毋需数据库。

MOIN [MoinMoin](#)是什么？它是一种[WikiWiki!](#)

Moin的语法是有些乏味儿的，你必须不断的{{{''''''adding braces and quotes''''''}}}}。Txt2tags为你带来简单的标记和统一的解决方案：一个源文件，多种产出物。

PM6 我估猜你不会知道这是Adobe PageMaker 6.0 独有的标记语言！它包含了样式、颜色表、美化、以及绝大部分PageMaker中的鼠标点击特性。你可以通过“Import Tagged text”菜单项来使用它。从记录上来看，它具有“类HTML”的标记格式。

Txt2tags能生成所有的标记，并且定义好丰富而有效的导言以设置段落样式和格式——这个部分通常是让人头疼的。**特别提醒：**不能有断行！每个段落必须是单独一行。

（作者按：笔者的《[regular expression's book](#)》这本书的葡萄牙语版就是用VI编辑，再用`txt2tags`转换为PageMaker格式然后付印的。）

TXT 纯文本，唯一的真实格式。

尽管`txt2tags`的标记非常直观而且不显眼，你仍然可以选择移除它们——只需将文件转换为纯文本即可。

在源文件中，标题添加下划线，而文本则是左对齐的。

1.4 目标文件格式支持一览

结构	html	xhtml	sgml	tex	lout	man	mgp	wiki	gwiki	doku	moin	pm6	txt
头部	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	N	Y
章节标题	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
段落	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
粗体字	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
斜体字	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
下划线字体	Y	Y	-	Y	Y	-	Y	Y	-	Y	Y	Y	-
删除线字体	Y	Y	N	Y	-	-	-	Y	Y	Y	Y	N	-
等宽字体	Y	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	Y	-
未格式化行	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
未格式化区域	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
引用区域	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
超链接	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	-	-
e-mail链接	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	-	-
本机链接	Y	Y	Y	N	N	-	-	N	N	Y	Y	-	-
命名链接	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	-	-
列表	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
计数列表	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
定义列表	Y	Y	Y	Y	Y	Y	N	Y	-	-	Y	N	Y
水平线	Y	Y	-	Y	Y	-	Y	Y	-	Y	Y	N	Y
图片	Y	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	N	-
表格	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y	N	N
附加结构	html	xhtml	sgml	tex	lout	man	mgp	wiki	gwiki	doku	moin	pm6	txt
图片对齐	Y	Y	N	N	Y	-	Y	Y	-	Y	N	N	-
单元格对齐	Y	Y	Y	Y	N	Y	N	N	-	-	Y	N	N
跨列单元格	Y	Y	N	N	N	N	N	N	-	-	N	N	N

图例

Y 支持

N 不支持（后续版本可能会支持）

- 不支持（在该目标格式中不存在）

1.5 三种用户接口：GUI、网页和命令行

不同用户的需求和使用环境不同，txt2tags相应的也具有灵活的执行方式。

程序有三种用户接口，每一种各有其目标和特性：

- **图形用户接口（GUI）**：使用Tk写成，支持窗口环境和鼠标点击。
- **网页**：使用PHP开发，支持通过浏览器运行txt2tags，而无需安装任何客户端程序。

- **命令行：**程序的核心，使用Python开发，所有的特性都包含于命令行选项之中。

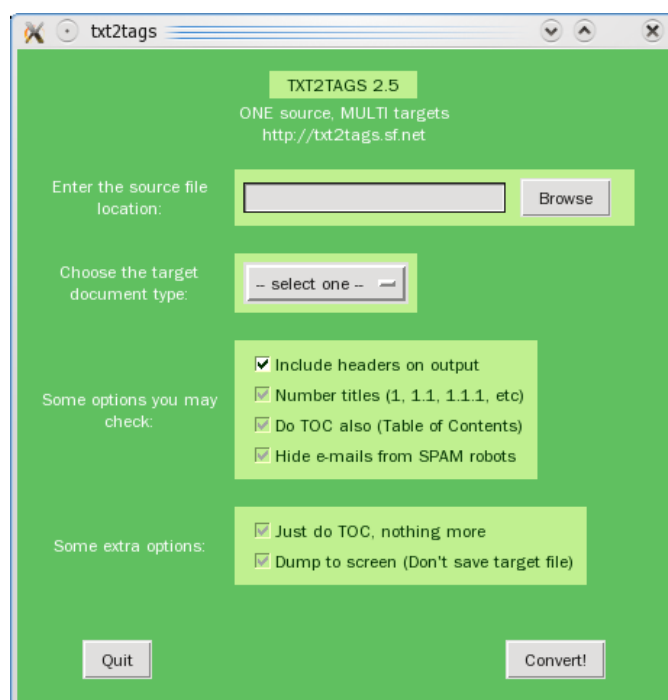
1.5.1 基于Tk的图形用户接口

自1.0版本开始，txt2tags就有一个能工作于Linux、Windows、Mac以及其他操作系统之上的图形用户接口。

程序能自动检测你的系统能否显示图形，如果可以，则不带任何参数的命令调用时运行图形接口。你也可以通过命令行参数`--gui`强制运行图形接口。当某些资源缺失时，程序给出错误信息。

提示：Tkinter是必须的。因为Python的标准发布总是包含它，实际上你的系统中应该已经有了。

图形界面的使用相当的简洁和直观：



1. 加载本机的.t2t文件，其内部定义的选项将自动识别。
2. 如果目标格式为空，必须在此处指定。
3. 此外还有一些选项可以附加选择，但并非必须。
4. 最后，点击“Convert!”按钮。

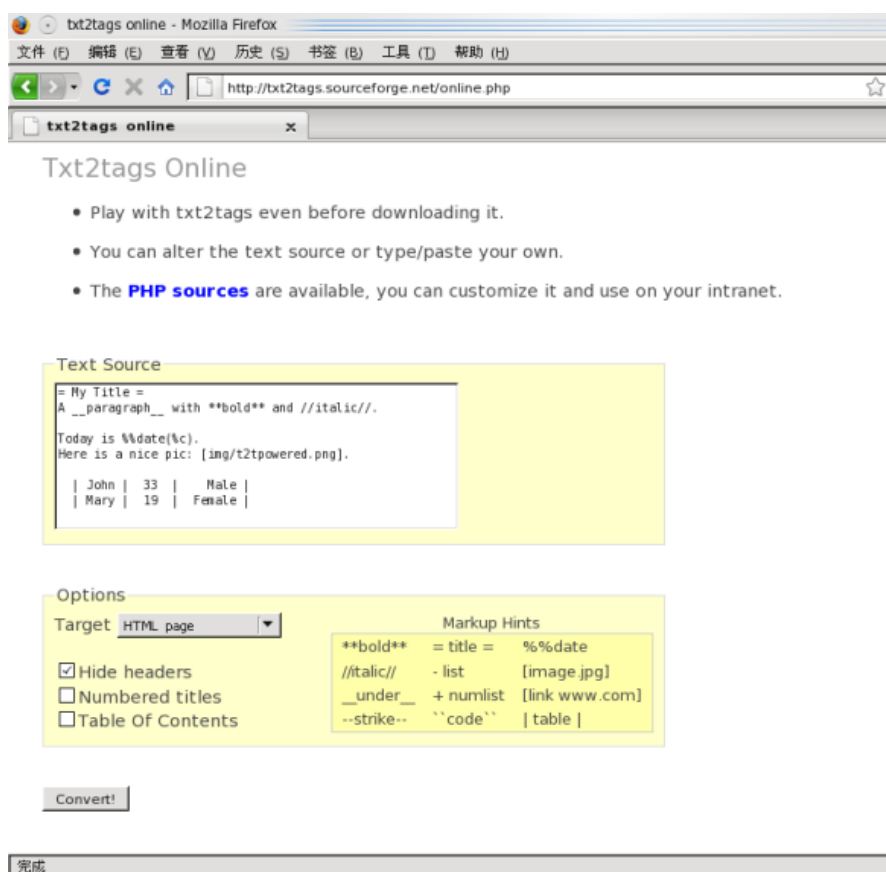
一个有用的选项是“*Dump to screen*”，输出结果不会写到文件，而是显示于屏幕上，以便检查，确认无误之后再输出到文件存储。

界面的默认颜色可以通过修改`~/txt2tagsrc`文件的`!guicolors`设定来更改。比如：

```
% set my own colors for the graphical interface (bg1, fg1, bg2, fg2)
%!guicolors: blue white brown yellow
```

1.5.2 在线方式

基于Web的界面可以通过互联网地址<http://txt2tags.sf.net/online.php>来访问，你可以在下载之前先试用txt2tags。



Web接口也可以部署于本地局域网（intranet）内，免去在每台机器上安装txt2tags的麻烦。

1.5.3 命令行

对于有经验的命令行用户来说，`--help` 选项提供的帮助信息应该足够了：

Usage: txt2tags [OPTIONS] [infile.t2t ...]

```
-t, --target=TYPE    set target document type. currently supported:
                    html, xhtml, sgml, tex, lout, man, mgn, wiki,
                    gwiki, doku, moin, pm6, txt
-i, --infile=FILE    set FILE as the input file name ('-' for STDIN)
-o, --outfile=FILE    set FILE as the output file name ('-' for STDOUT)
-n, --enum-title      enumerate all title lines as 1, 1.1, 1.1.1, etc
-H, --no-headers      suppress header, title and footer contents
    --headers          show header, title and footer contents (default ON)
    --encoding          set target file encoding (utf-8, iso-8859-1, etc)
    --style=FILE       use FILE as the document style (like HTML CSS)
    --css-sugar         insert CSS-friendly tags for HTML and XHTML targets
    --css-inside        insert CSS file contents inside HTML/XHTML headers
    --mask-email       hide email from spam robots. x@y.z turns <x (a) y z>
    --toc              add TOC (Table of Contents) to target document
    --toc-only         print document TOC and exit
    --toc-level=N      set maximum TOC level (depth) to N
    --rc              read user config file ~/.txt2tagsrc (default ON)
    --gui              invoke Graphical Tk Interface
-q, --quiet           quiet mode, suppress all output (except errors)
-v, --verbose         print informative messages during conversion
-h, --help           print this help information and exit
-V, --version        print program version and exit
    --dump-config      print all the config found and exit
```

Turn OFF options:

```
--no-outfile, --no-infile, --no-style, --no-encoding, --no-headers
--no-toc, --no-toc-only, --no-mask-email, --no-enum-title, --no-rc
--no-css-sugar, --no-css-inside, --no-quiet
```

Example:

```
txt2tags -t html --toc myfile.t2t
```

By default, converted output is saved to 'infile.<target>'.
Use --outfile to force an output file name.

If input file is '-', reads from STDIN.

If output file is '-', dumps output to STDOUT.

示例 假设你已经编写好一个正确标记的文件file.t2t，现在我们来尝试一些转换。

转换为HTML	\$ txt2tags -t html file.t2t
转换为HTML，使用输出重定向	\$ txt2tags -t html -o - file.t2t > file.html
包含目录表	\$ txt2tags -t html --toc file.t2t
带数字编号的目录表	\$ txt2tags -t html --toc --enum-title file.t2t
内容概览	\$ txt2tags --toc-only file.t2t
编号的概览	\$ txt2tags --toc-only --enum-title file.t2t
从标准输入读取一行	\$ echo -e "\n**bold**" txt2tags -t html --no-headers -
Email地址识别测试	\$ echo -e "\njohn.wayne@farwest.com" txt2tags -t txt --mask-email --no-headers -
转换后编辑	\$ txt2tags -t html -o- file.t2t sed "s/<BODY .*/<BODY BGCOLOR=green>/" > file.html

提示：从1.6版本起，前处理和后处理可以通过%!preproc 和%!postproc 宏来完成。

第二章 如何开始？

下载然后运行——一切就是这么简单。

2.1 下载、安装Python

首先你必须下载安装Python解释器，如果你的系统已经安装过Python，请跳过本节。

Python是一种极好的编程语言，能够跨平台的在Windows、Linux、UNIX、Macintosh以及其他的操作系统上运行。你可以从[Python主页](#)找到安装文件以及安装指南。Txt2tags运行要求Python 2.0以后的版本。

如果你不确定是否已经安装过Python，请打开一个控制台（tty、xterm、MSDOS），输入命令python，如果Python并未安装，系统会给出错误信息。

2.2 下载txt2tags

Txt2tags的项目主页是<http://txt2tags.sf.net>。全部程序文件打包成一个tarball压缩文件（.tgz 文件），你可以使用大多数的压缩工具来提取（包括WinZip）。请下载最新的版本，早期版本仅仅作为历史存档之用。

2.3 安装txt2tags

事实上，txt2tags只是一个单文件的Python脚本，毋需安装。

运行程序需要的唯一一个文件是脚本txt2tags，压缩包内的其他文件是软件的文档、示例以及一些工具。

最低级的使用txt2tags的方法是直接通过Python来调用：

```
prompt$ python txt2tags
```

如果想将txt2tags 作为一个能够直接运行的程序“安装”到系统，请将txt2tags 脚本复制（或者创建一个链接）到系统PATH 变量定义的目录，并且确认系统能够运行它。

UNIX/Linux/Mac 将脚本文件赋予执行权限（`chmod +x txt2tags`），然后复制到\$PATH 变量定义的目录下（比如：`cp txt2tags /usr/local/bin`）。

Windows 重命名脚本文件，添加.py 作为扩展名（命令行下执行：`ren txt2tags txt2tags.py`），然后复制到系统PATH 环境变量定义的路径下（比如：`copy txt2tags.py C:\WINNT`）

如果想要使用图形用户接口，你可以创建一个桌面启动器（快捷方式）。

2.3.1 为Windows用户专门打包的安装程序

对于Windows用户，txt2tags还有两个打包好的安装程序，你只需点击鼠标即可完成安装：

- 单独的txt2tags脚本，针对已经安装过Python解释器的系统。
- 独立运行的版本，毋需另外安装Python解释器（包含一个嵌入的版本）。

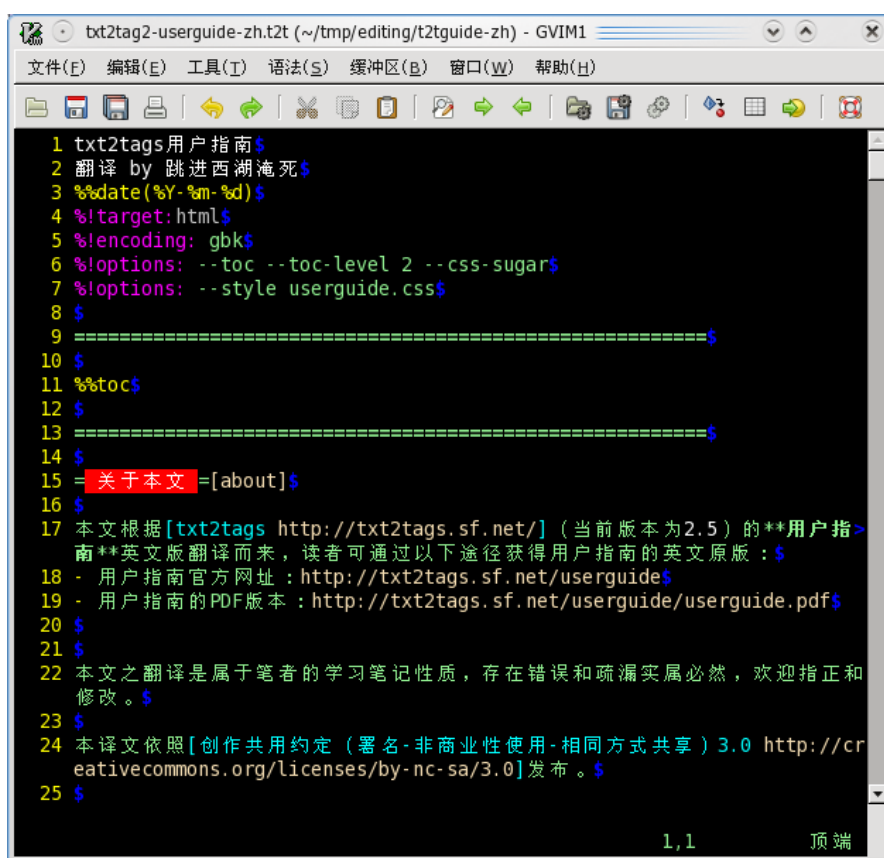
你可以访问 *Txt2tags-Win* 主页下载安装包：<http://txt2tags-win.sf.net/>

2.4 安装编辑器的语法高亮文件

Txt2tags提供适合以下编辑器使用的语法高亮文件：

- Vim (www.vim.org)
- Emacs (www.emacs.org)
- Nano (www.nano-editor.org)
- Kate (<http://kate.kde.org>)
- gedit (<http://www.gnome.org/projects/gedit/>)
- TextMate (<http://macromates.com/>)

语法高亮文件包含全部txt2tags标记和规则，以帮助用户编写出无误的文档——正确书写的标记将以彩色显示。



```
txt2tag2-userguide-zh.t2t (~/tmp/editing/t2tguide-zh) - GVIM1
文件(F) 编辑(E) 工具(T) 语法(S) 缓冲区(B) 窗口(W) 帮助(H)
1 txt2tags用户指南$
2 翻译 by 跳进西湖淹死$
3 %%date(%Y-%m-%d)$
4 %!target:html$
5 %!encoding:gbk$
6 %!options:--toc--toc-level2--css-sugar$
7 %!options:--styleuserguide.css$
8 $
9 =====$
10 $
11 %%toc$
12 $
13 =====$
14 $
15 = 关于本文 =[about]$
16 $
17 本文根据[txt2tags http://txt2tags.sf.net/] (当前版本为2.5) 的**用户指>
南**英文版翻译而来，读者可通过以下途径获得用户指南的英文原版：$
18 - 用户指南官方网址：http://txt2tags.sf.net/userguide$
19 - 用户指南的PDF版本：http://txt2tags.sf.net/userguide/userguide.pdf$
20 $
21 $
22 本文之翻译是属于笔者学习笔记性质，存在错误和疏漏实属必然，欢迎指正和
修改。$
23 $
24 本译文依照[创作共用约定 (署名-非商业性使用-相同方式共享) 3.0 http://cr
eativecommons.org/licenses/by-nc-sa/3.0]发布。$
25 $
1,1 顶端
```

用Vim打开示例文件

语法文件对于不同的编辑器有不同的安装过程，请参考语法文件的头部说明和编辑器的文档。

第三章 第一个文档

3.1 先利其器

你需要三件工具来完成你的第一次转换尝试：txt2tags，一个文本编辑器和一个网络浏览器。

1. 确认一下txt2tags是否已经安装并且能够正确运行。
 - **命令行方式：**执行“txt2tags”命令，程序会返回一个“缺少输入文件（Missing input file）”的信息。如果不能正确运行，尝试使用“python /path/to/txt2tags”，甚至“/path/to/python /path/to/txt2tags”以防止Python不在环境变量定义的路径中。
 - **图形化操作：**点击程序图标启动图形界面。
2. 开启你喜欢的文本编辑器——任何一个都行，无论是古老的VI，还是MS Word，乃至OpenOffice.org。用编辑器创建一个空文档作为你的第一个txt2tags源文件。
3. 启动你偏爱的浏览器，用来查看转换得到的HTML网页。

3.2 编写文档头部

1. 在编辑器中，在第一行写上文档的主标题：我的第一个文档
2. 在第二行写上子标题：*txt2tags*测试
3. 然后在第三行写上时间，比如：星期日，2009

如果一切正常，现在你可以看到一个包含三行内容的文档：

我的第一个文档

txt2tags测试

星期日，2009

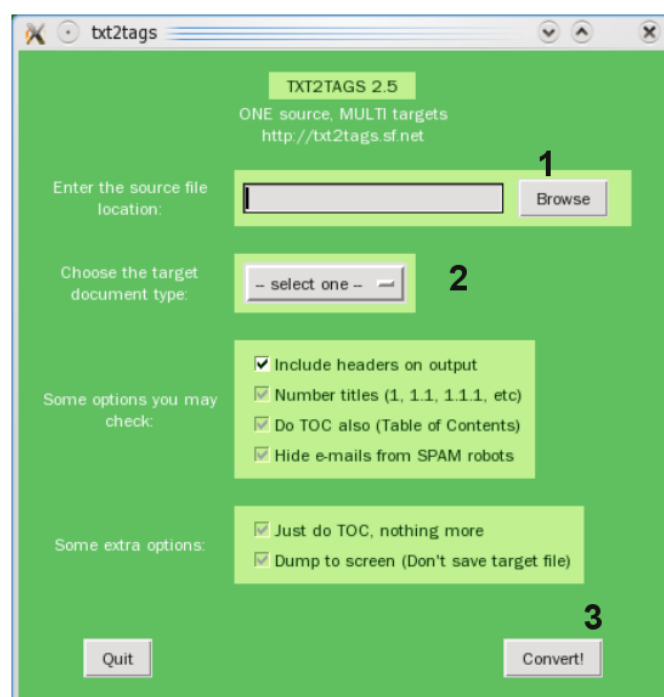
这只是一般文档的一部分，但是我们已经可以用它来尝试一下转换工作了。

将文档以文件名`test.txt`保存，记得要记住保存的路径，因为一会儿你要用到它。

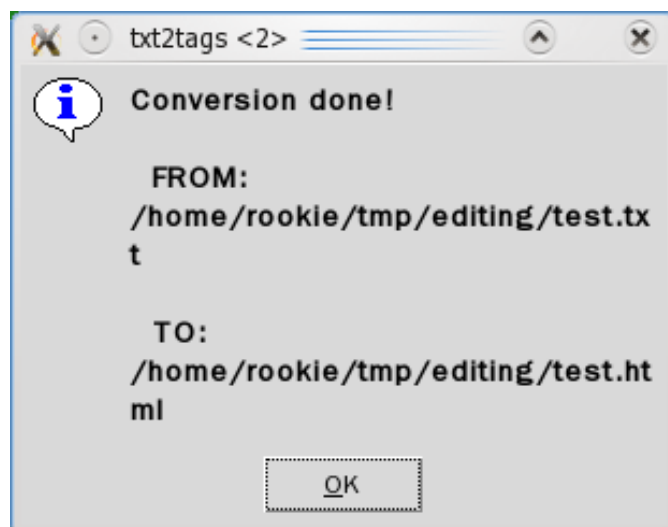
3.3 第一次转换——图形界面操作

如果你使用命令行，请跳过本节，直接阅读下一节。

如果你使用图形界面，请跟着我做：



1. 点击“Browse”按钮，选择你刚才保存的`test.txt`文件。
2. 回到第一个窗口，在“Target document type”下拉菜单中选择“HTML page”。
3. 点击“Convert!”按钮。



这时会弹出一个对话框，告诉你文件已经成功的转换了。生成的HTML文件与源文件在同一个文件夹下，以“html”为扩展名。

3.4 第一次转换——命令行方式

如果你使用图形界面，请跳过本节直接阅读下一节。

如果你使用命令行方式，请切换到源文件所在的目录，执行：

```
txt2tags --target html test.txt
```

请注意：在命令的各部分中间以空格分隔，但是在选项字符串“--target”内部不能有空格，这个选项后面紧跟着字符串“html”，告诉程序你想要的目标文件格式。最后一部分是源文件的文件名。

如果转换正常完成，结果会存储到文件test.html中，同时给出“*txt2tags wrote test.html*”的信息。如果不是这样，程序会报错，提示你可能出错了，请仔细检查一下。

一个示例：

```
prompt$ txt2tags --target html test.txt
txt2tags wrote test.html
prompt$
```

3.5 查看结果

用浏览器打开文件test.html以检查是否正常。



看见结果了吗！你只是简单的输入了三行文字，txt2tags则为你完成了所有设置HTML文件头部信息的工作：对齐方式、大小、间隔以及外观。文档的主标题同时也出现在了浏览器的标题栏中。

你只需输入文本，txt2tags帮你搞定一切；)

提示：txt2tags生成的HTML页还能使用外部CSS，以实现页面外观100%的定制。

3.6 编写文档体

现在返回文本编辑器进行下一步：撰写文档的内容。你可以像平日里撰写电子邮件一样的输入文字，毋需特别的“标记”，txt2tags会自动识别出段落和列表。

然后，再一次：保存、转换再查看结果。这就是txt2tags文档的编写流程。你可以把精力放在内容上，更快的完成文件的编写——毋需点鼠标，毋需菜单，没有窗口，没有什么来分散你的注意力。

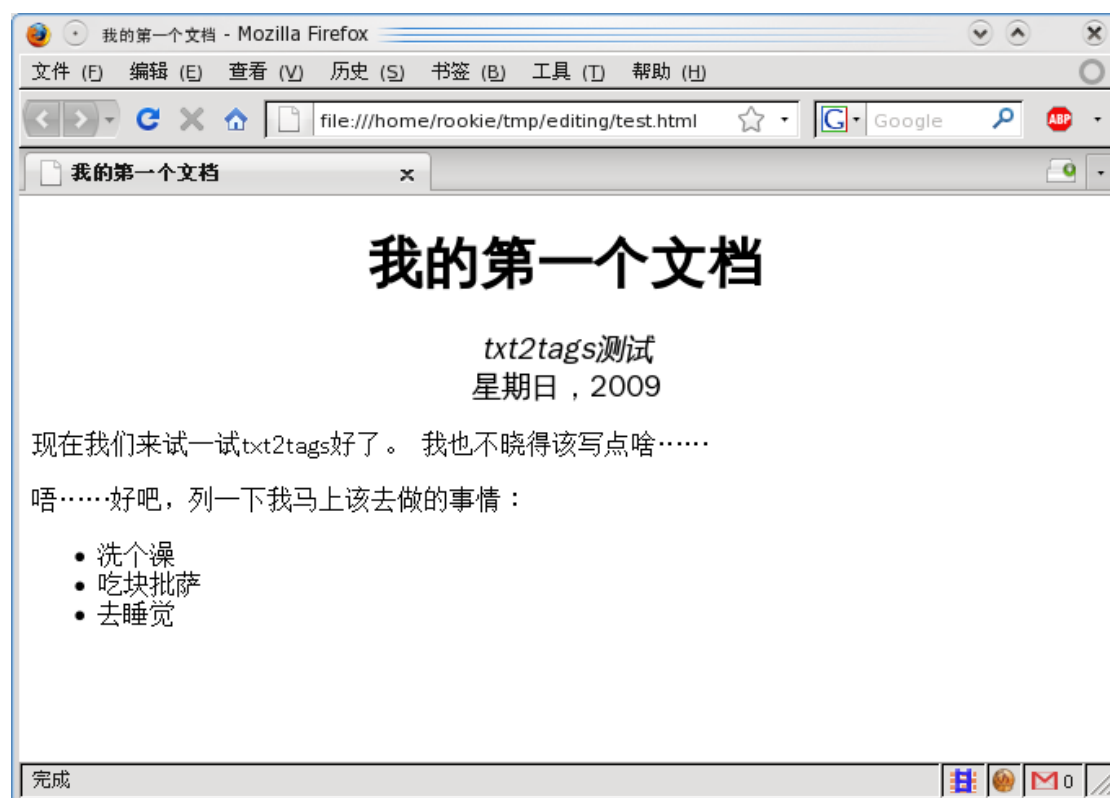
假设文件test.txt中写上了如下的内容，完全的纯文本，将它们跟生成的HTML文件做一个对比：

```
我的第一个文档
txt2tags测试
星期日, 2009
```

现在我们来试一试txt2tags好了。我也不晓得该写点啥……

唔……好吧，列一下我马上该去做的事情：

- 洗个澡
- 吃块披萨
- 去睡觉



你毋需任何HTML语言的准备知识就能够编写出完整的主页，不必插入任何标记。进一步，同一个源文件还可以转换为其他任何txt2tags所支持的格式。

除了纯文本之外，txt2tags有一些非常简单的标记来实现其他的格式和文档结构，比如**粗体**、*斜体*、标题、图片、表格，等等。举个简单的例子，**产生粗体**，== 得到标题==。详情请参阅[标记示例 \(Txt2tags Markup Demo\)](#)。

第四章 掌握txt2tags的结构

4.1 文档的各个部分

Txt2tags的标记文本可以分为三个部分，每个部分各自的规则，实现不同的功能，包括：

头部域 (Header Area) 放置文档标题、作者、版本号以及日期信息。可选区域。

设置域 (Config Area) 定义全局设定，修改解析时行为 (Place for general Document Settings and Parser behavior modifiers.)。可选区域。

主体域 (Body Area) 文件内容。必须的区域。

前两个区域是可选的，只有主体域 是必须的。

各个区域之间的分隔方式比较特别，下一章会详细描述，现在我们给出一个直观的图形化的描述：

```
-----  
|           |  
| HEADERS  | 1. First, the Headers  
|           |  
| CONFIG   | 2. Then the Settings  
|           |  
| BODY     | 3. And finally the Document Body,  
|           |  
| ...     | which goes until the end  
| ...     |  
|-----|
```

简单的说，各个部分是这样定义的：

头部域 文件内容的前三行；或者第一行是空白行，表示“没有头部”。

设置域 紧接着头部（第四行或者第二行）开始，结束于主体域的开始。

主体域 从头部区域之后的第一个有效的行（非注释或设置）开始。

4.1.1 完整的示例

```
My nice doc Title
Mr. John Doe
Last Updated: %%mtime(%c)

%! Target   : html
%! Style    : fancy.css
%! Encoding : iso-8859-1
%! Options  : --toc --enum-title

Hi! This is my test document.
Its content will end here.
```

4.2 头部域

位置：

- 固定位置：文件的头三行。
- 固定位置：文件的第一行，并且为空白行。这意味着“没有头部”。

头部是唯一固定位置和行数的区域，总是出现在源文件的头三行。各行的内容可以随意，并没有特别的要求。不过，对于大多数的文档，推荐以下的内容作为头部：

- 第一行：文档标题
- 第二行：作者姓名和（或）邮件地址
- 第三行：文档的编写日期和（或）版本号（推荐使用宏%%date）

请记住：源文件的头三行将会是目标文件的头三行，在目标文件中会以高的对比与文档内容相区分（比如：大号字体，粗体）。如果允许分页，头部将会单独的在第一页居中显示。

较少（或无）头部

有些时候作者会希望少于三行的头部，仅仅给出文档标题和（或）日期信息，这时只需将第二行和（或）第三行留为空行，这部份就不会在目标文件中显示。但是，请切记：即使是留为空白，这些行仍然是头部的一部分，主体域总是在这三行之后开始。

只有标题（第一行）是必须的，但是你也可以把它留白，这意味着文档**没有头部**，而主体域就紧接着从第二行开始。如果你打算在完成转换之后再来自定义头部信息，那么没有头部的选择是很有用的。命令行选项`--no-headers`通常用来完成这一操作。

总结

简而言之：“**头部只是位置，而非内容**”

源文件的第一行会出现在目标文件的第一行，二三行同样。

4.3 设置域

位置：

- 紧接着头部区域之后开始
 - 如果定义了头部则从**第四行**开始
 - 如果无头部，则从**第二行**开始
- 结束于主体域开始
 - 结束于一个非属性设定，空白或者注释行

设置域是可选的。普通用户甚至可以编写出许多txt2tags文档，却不知道这个区域的存在，但是有经验的用户往往受益于它提供的有效的控制功能。设置域用来记录具体文档的设置，这样就省略了转换时繁杂的命令行选项。比如说，你可以设定好缺省的目标文件类型和字符编码。在“[设定](#)”一节中对此有详细的叙述。

4.4 主体域

位置:

- 开始于第一个有效行
 - 头部、设置域以及注释都不是有效的行。
- 结束于文件的末尾 (EOF)

除了头部和设置域以外的所有内容都是主体域。主体域包含来了文件的内容以及txt2tags支持的所有格式和文档结构的信息。在主体域内你也可以放置注释，作为*TODOs* 和自提示。

命令行选项`--no-headers` 会忽略头部，只转换主体域的内容。这是很有用的：你可以用单独的文件来定义头部，然后与转换后的主体域连接起来。

4.5 设定

设定 (Settings) 是特殊的配置选项，位于文档的设置域内，用以控制转换的过程。设定的语法如：

`$! 关键词 : 设定值`

以下是有效的关键词列表：

关键词	描述
target	设定缺省的目标文件格式。
options	设定缺省的转换选项，格式与命令行选项相同。
style	设定文档样式。主要用作为HTML/XHTML文档定义外联的CSS样式表，以及为LaTeX文档加载宏包。
encoding	设定字符编码。当文档包含国际化字符和非ASCII字符时需要指定编码。
preproc	输入文件处理。设定应用于源文件的主体域中的“查找/替换”规则。
postproc	输出文件处理。设定应用与输出文件中的“查找/替换”规则。

示例：

```

%! Target   : html
%! Options  : --toc --toc-level 3
%! Style    : fancy.css
%! Encoding: iso-8859-1
%! PreProc  : "AMJ"           "Aurelio Marinho Jargas"
%! PostProc: '<BODY.*?>'    '<BODY bgcolor="yellow">'

```

4.6 命令行选项

改变txt2tags默认行为最快捷的方式是使用命令行选项。顾名思义，这些选项只能应用于命令行方式，不适用于GUI和Web界面。

与其他工具一样，程序接受一串预定义的选项，每个选项都是由一个单连字符引导的字母，或者两个连续的连字符引导的一个或多个单词，比如-t和--target。“target”选项是唯一的一个必须的选项，其他的都是可省略的选项。

常用的选项包括：--outfile，指定输出文件名；--toc，自动生成目录表；--encoding，设定文档的字符编码。大多数的选项可以通过前缀一个“no-”来关闭，比如：--no-encoding 和--no-toc。

使用%!options 的格式，你可以在源文件的设置域设定需要的选项，这样在命令行中就可以省略掉这些选项了。比如：%!options: --toc -o mydoc.html。目标文件格式的指定是个例外，它有自己的格式：%!target: html。

使用--help 选项能获得完整的选项列表。

4.7 用户配置文件（RC文件）

通用的设定存储于用户配置文件（又称为RC文件）中。对于在每一个源文件中反复使用的设定，你可以把它们写到RC文件中，以便于所有的源文件使用。这个文件的缺省位置取决于你使用的操作系统，你也可以通过定义环境变量来指定其位置。

RC 文件的位置	
Windows	%HOMEPATH%_t2trc
Linux and other	\$HOME/.txt2tagsrc
User defined	T2TCONFIG 环境变量

设定的格式与源文件设置域中使用的选项格式完全相同。以下是包含在程序包的doc/txt2tagsrc 的示例文件：

```
% my configs

%% Always use CSS-friendly tags in HTML
%!options(html): --css-sugar

%% Change the default TOC depth for all targets
%!options: --toc-level 4

%% Set the default encoding for all documents
%!options: --encoding iso-8859-1
```

除去空白行、注释行和有效配置行之外的任何一个多余的行，都会使得txt2tags在运行时产生错误，因此编辑这个文件时请尽量小心。

在执行转换时，程序自动将RC文件的内容应用于源文件，如果你想要关闭这一行为，则可使用命令行选项--no-rc。

4.8 配置的优先级和加载顺序

有三种方式告诉txt2tags需要使用的选项和设定，其读取和应用的顺序是：

1. 用户配置文件的设定
2. 源文件设置域的设定
3. 命令行选项

程序首先读取RC文件的内容（如果存在的话），并将相应的设置应用于当前源文件。然后它检查源文件的设置域的设定，如果找到，应用之并且覆盖掉RC文件的设定以壁面冲突。最后是命令行选项，它的效力比前两者都高。

这即是说，如果文档的字符编码同时用三种方式来设定，最终被应用的是命令行选项的设定。

4.9 %!include 指令

`include` 命令用于将外部文件的内容包含进源文件中。它不是一个配置选项而是一个命令，对于主体域区域来说，则是一个有效的行。

`include` 命令可用于将大型的文档分割为较小的片段（比如整本书中的章节），或者是将外部文件的全部内容包含进源文件中。比如：

```
My first book
Dr. John Doe
1st Edition

%!include: intro.t2t
%!include: chapter1.t2t
%!include: chapter2.t2t
...
%!include: chapter9.t2t
%!include: ending.t2t
```

文件名接在字符串`%!include`后面。支持可选的“`target`”选项，因此下面的格式也是合法的：

```
%!include(html): file.t2t
```

注意：`include` 命令将被包含文件的主体域区域读入源文件中，而头部和设置域则被忽略。所以，你可以单独转换这个文件，或者将它包含进主文档中一同转换。

此外，还有另外三种形式的包含：

- 完全引用（Verbatim）包含
- 原文（Raw）包含
- 带标记（Tagged）包含

完全引用（Verbatim）包含会保留读入文件原始的空格和格式，就像是txt2tags文件的完全引用（`VERB`）域一样。要实现这种类型的包含，请将文件名置于反引号中：

```
%!include: ``/etc/fstab``
```

原文 (Raw) 包含会照原样包含读入文件，不会寻找和解析其中的任何标记，就像之余原文 (RAW) 域中一样。要实现这种类型的包含，请将文件名置于双引号中：

```
%!include: "nice_text.txt"
```

带标记 (Tagged) 包含会将读入文件内容直接传递给目标文件，txt2tags不会进行任何解析和控制。这种方式适用于在文档中包含额外的带标记的内容，对于缺省的头部或脚注信息以及txt2tags不支持的复杂标记十分有用：

```
%!include: 'footer.html'
```

注意：请将文件名置于两个单引号中间。由于插入的内容是已经格式化的，你必须确认目标文件格式以防止出错。

4.10 %!includeconf 指令

`includeconf` 命令用于将外部配置文件读入当前文件中。这个命令仅在源文件的设置域内出现才是合法而有效的。

当多个文件共享同样的配置时，你可以将这些配置集中放置，然后用这个指令来读入。请在每个需要读入外部集中配置的文件中使用`includeconf` 命令。比如：

```
My First Document
```

```
John Doe
```

```
July, 2004
```

```
%!includeconf: config.t2t
```

```
Hi, this is my first document.
```

外部配置文件使用的格式与RC文件相同。

第五章 掌握标记

txt2tags的全部标记概览：

基本		美化	
头部	First 3 lines	粗体	**words**
标题	= words =	斜体	//words//
带编号的标题	+ words +	下划线	_words_
段落	words	删除线	-words-
链接	[label url]	等宽	“words“
图片	[filename.jpg]	原始文字	””words””
其他			
引用	<TAB>words	分隔线	——...
列表	- words	粗分隔线	=====...
编号列表	+ words	表格	cell1 cell2 cell3...
定义列表	: words	锚点 (标签)	= title =[anchor]
注释行	% comments	注释域	%%% \n comments \n%%%
完全引用行	““ word	完全引用域	““ \n lines \n ““
原文行	””” words	原文域	””” \n lines \n ”””

通用规则：

- 头部是文档的头三行，其中的标记不会解析。
- 标题是包围标题文字的对称的“=”或“+”字符。字符越多，标题级数越深。
- 美化符会忽略标记与其中的内容之间的空格。
- 注释符号“%”必须位于每一行的开头（第一列）。
- 图片文件名必须以GIF、JPG、PNG或者类似的扩展名结束。

- 完全引用（Verbatim）和原文（Raw）中的标记不会被解析。
- 分隔线/粗分隔线必须是连续20个字符以上。
- 引用和列表的嵌套/解嵌套通过缩进来定义。
- 表格标题行通过位于行首的双管道符“||”来定义。

5.1 头部

- 描述：确定文档头部
- 属性：多行，空格自由（可自由包含空格，下同。原文：FreeSpace），无对齐，无嵌套
- 可包含：宏
- 语法：
 - 源文件的头三行
 - 第一行留空则表示无头部。适合于用户定制的头部（原文：Nice for command line oneliners or customized headers）。
 - 第二行和/或第三行留空，将忽略头部的部分内容。
- 细节：
 - 头部域的标记不会被解析
 - 源文件的头三行也是目标文件的头三行，在目标文件中与主体域以高对比来显示，或者置于单独的第一页上（如果允许分页的话）。
 - 头部的内容可以随意，没有硬性规定的信息，然而推荐使用下面的写法：
 - * 第一行：文件标题
 - * 第二行：作者姓名和/或电子邮件
 - * 第三行：文档日期和/或版本号（建议使用`%%mtime`宏）

5.2 标题与编号的标题

- **描述:** 确定一个编号或者不编号的章节标题
- **属性:** 非多行, 空格自由, 无对齐, 无嵌套
- **可包含:** 原始文本 (Raw)
- **语法:**
 - 对于所有的规则, 用 “+” 替代 “=” 可得到编号的标题
 - 包围文字的符号数目保持对称, =像这样=
 - 符号数目增加, 章节深度随之增加: =title=, ==subtitle==, ===subsubtitle===, ...
 - 最大深度为5级标题, =====像这样=====
 - 数目不对称的标记不构成标题, =像这样===
 - 标记包围的内部可以自由添加空格, = 像这样=
 - 标题可以有锚点 (anchor), =像这样=[anchor]。可以创建一个到此锚点的链接, 形如[本地链接#anchor]
 - 锚点名称只能由字母、数字、下划线和连字符构成 (A-Za-z0-9_-)
- **细节:**
 - 标题中的其他标记不会解析
 - 标题中的宏不会解析

5.3 段落

- **描述:** 确定一个自然段
- **属性:** 多行, 空格自由, 无对齐, 无嵌套
- **可包含:** 宏, 美化符, 原始文 (Raw), 链接, 图片, 注释
- **语法:**
 - 由空行分隔的多行文本构成段落
 - 其他的块, 如列表、引用域、表格或完全引用域等, 也可作为段落的结束

5.4 注释

- **描述:** 用于插入不会写入目标文件的文本
- **属性:** 非多行, 非空格自由, 无对齐, 无嵌套
- **可包含:** 无
- **语法:**
 - 以一个百分号% 起始的行 (%位于第一列) 构成一个注释行, %像这样
 - 不能有前导空格 (按: 这就是所谓“非空格自由”)
- **细节:**
 - 注释文本不会写入转换后的文件
 - 注释是非块状的, 因此每一个注释行都必须以% 作为前导
 - 可用于TODO 和FIXME 提示, 以及作者自提示

5.5 粗体, 斜体, 下划线和删除线

- **描述:** 用于在段落、表格、列表和引用域内插入粗体/斜体/下划线/删除线字体
- **属性:** 非多行, 非空格自由, 无对齐, 可嵌套
- **可包含:** 宏, 美化符, 原始文 (Raw), 链接, 图片
- **语法:**
 - 双星号包围的文字产生**粗体**, **像这样**
 - 双斜杠包围的文字产生**斜体**, //像这样//
 - 双下划线包围的文字产生**下划线**字体, __像这样__
 - 双连字符包围的文字产生**删除线**字体, --像这样--
 - 标记必须与内部的文字紧密贴合 (不能有空格): ** 像这样** 是非法的
- **细节::**

- 源文件中每一对美化符都必须位于同一行，中间不能有换行符
- 美化符内可以包含宏，比如：`**%%date**`
- 美化符可以混合使用，如：`**__like__ //this//**`

5.6 等宽字体

- **描述：**用于在段落、表格、列表和引用域内插入等宽字体的文本
- **属性：**非多行，非空格自由，无对齐，无嵌套
- **可包含：**无
- **语法：**
 - 用双反引号包围文字，‘‘像这样‘‘
 - 标记必须与内部文字紧密贴合（不能有空格）：‘‘ 像这样‘‘ 是非法的
- **细节：**
 - 内部的标记不会解析
 - 内部的宏不会解析
 - 所有的等宽文字必须位于单一行内，不能有换行符
 - 在部分目标文件中，内部的空格会被保留下来；在其他的目标格式中，连续的空格则被压缩为一个。
 - 还可以得到粗体的等宽字体，只需放置于粗体标记内部：`***monobold**`。这一规则对其他的美化符同样适用，比如：`//italic//`，`__underline__`。

5.7 完全引用行和完全引用域

- **描述：**用于插入程序源代码或者其他预格式化的文本，保留空格和换行符，并以等宽字体显示
- **属性：**多行，非空格自由，无对齐，无嵌套
- **可包含：**无

- 语法- 完全引用行：
 - 以三个相继的反引号加一个空格引导，后接文字，““ 像这样
 - 反引号必须位于每行的第一列，不能有前导空格
- 语法- 完全引用域：
 - 第一行为三个相继的反引号““，从下一行开始为单行或多行文本，最后一行为三个相继的反引号““
 - 标记不能有前导空格，也不能有后随的空格
- 细节：
 - 内部的标记不会解析
 - 内部的宏不会解析
 - 如果到达源文件末尾（EOF），尚未闭合的完全引用域将自动关闭

5.8 隔离行和粗隔离行

- 描述：生成水平线隔离行或者粗隔离行
- 属性：非多行，空格自由，无对齐，无嵌套
- 可包含：无
- 语法：
 - 隔离行由多个虚线“-”或者下划线“_”构成
 - 粗隔离行由多个等号“=”组成
 - 至少包括20个相应的字符
 - 线可以有前导和后随的空格
 - 在同一行的任何其他字符将破坏标记
- 细节：
 - 如果目标格式不支持水平线，自动转化为一个注释行
 - 粗隔离行在某些目标格式中可能有不同的作用：
 - * 更粗的隔离行
 - * 放映暂停，比如在MagicPoint 中
 - * 分页，比如在LaTeX 中

5.9 链接和命名链接

- **描述:** 确定一个指向互联网或本地的链接
- **属性:** 非多行, 非空格自由, 无对齐, 无嵌套
- **可包含:** 宏, 原始文 (Raw), 图片
- **语法:**
 - 合法的URL、ftp、新闻组或电子邮件地址能被程序自动识别并转换为超链接
 - 协议 (http, https, ftp等) 可省略, 如: `www.likethis.com`
 - 可以为链接命名, 如: `[click here www.url.com]`
 - 可以创建图片链接: `[[image.jpg] www.url.com]`
 - 链接地址允许宏替换: `[see source %%infile]`
 - 链接名字允许宏替换: `[mirror of %%outfile www.url.com]`
 - 全部链接标识符必须位于源文件的同一行内, 不能有换行符
- **细节:**
 - 如果目标格式不支持链接, 则链接文字仅仅加上下划线

5.10 引用

- **描述:** 确定一个引用 (缩进) 行
- **属性:** 多行, 非空格自由, 无对齐, 可嵌套
- **可包含:** 宏, 美化符, 引用, 原始文, Bars, 链接, 图片, 注释
- **语法:**
 - 以一个制表符 (TAB) 开始的一行
 - 更多的制表符则增加引用缩进的深度
 - 引用行能不允许列表和表格
- **细节:**

- 如果到达文件末尾（EOF），尚未闭合的引用自动关闭
- 某些目标格式不支持引用嵌套，则子嵌套自动外移到与外层嵌套同级
- 引用深度原则上没有限制，但某些目标格式可能会有限制，则超过最大深度的子嵌套自动外移。

5.11 列表，编号列表和定义列表

- **描述：** 确定列表项开始
- **属性：** 多行，非空格自由，无对齐，可嵌套
- **可包含：** 宏，美化符，列表，表格，完全引用，原始文，Bars，链接，图片，注释
- **语法：**
 - 以单个虚线“-”/加号“+”/冒号“:”紧接单个空格开始一行
 - 列表项的第一个字符不能是空格（定义列表除外）
 - 可选的前导空格（普通空格，非制表符）将开始一个子列表（列表嵌套）
 - 子列表以较少深度的表项或者空表项结束
 - 所有开放的列表以两个相继的空行结束
- **细节：**
 - 如果到达文件末尾（EOF），所有尚未闭合的列表自动关闭
 - 列表可以混合使用，比如编号列表内部可以嵌套定义列表
 - 某些目标格式可能不支持列表嵌套，则子列表自动外移到与外层列表同级
 - 嵌套深度原则上没有限制，但是某些目标格式可能会有限制，则超过最大深度的子列表自动外移

5.12 图片

- **描述：** 插入一副图片

- **属性:** 非多行, 非空格自由, 可对齐, 无嵌套
- **可包含:** 宏
- **语法:**
 - 将图片的文件名放在一对方括号中, 如: `[likethis.jpg]`
 - 文件名必须以PNG, JPG, GIF 等图形格式扩展名结束, 不区分大小写
 - 文件名可以含有符号 (原文: Symbols are allowed on the filename), 如: `[likethis!~1.jpg]`
 - 文件名允许宏替换, 比如: `[report-%%date(%Y-%m-%d).png]`
 - 文件名不能包含空格, 如: `[like this.jpg]`
 - 文件名与方括号之间不能有空格, 如: `[likethis.jpg]`
- **细节:**
 - 如果目标格式不支持插入图片, 图片名称将置于一对圆括号中显示
 - 插入图片标记在一行中的位置定义了图形的对齐方式:
 - * `[LEFT.jpg]` 左对齐左对齐左对齐
 - * 居中居中居中 `[CENTER.jpg]` 居中居中居中
 - * 右对齐右对齐右对齐 `[RIGHT.jpg]`

5.13 表格

- **描述:** 限定有任意列的表格行
- **属性:** 多行, 空格自由, 可对齐, 无嵌套
- **可包含:** 宏, 美化符, 原始文, 链接, 图片, 注释
- **语法:**
 - 一个前导的管道符 “|” 确定一个表格行
 - 一对前导的管道符 “||” 确定一个表格的标题行
 - 首个管道符前的前导空格确定表格居中
 - 单元格以管道符间隔 (形如: 空格|空格)

- 表格第一行以一个管道符结束，表示边框可见
- 其余行结尾的管道符可以忽略（仅作装饰用）
- 单元格以多余一个管道符结束获得“跨列”效果：“||”跨两列，“|||”跨三列，以此类推
- 单元格内的空格决定内部的对齐方式
- 示例：| 包含| 5列| 的| 表格| 行|

- 细节：

- 每一个表格行的内容必须在源文件内的同一行内，不能有换行符
- 带列对齐的目标格式（比如SGML和LaTeX）以第一行的对齐方式作为所有行的缺省对齐方式
- 任意的非表格行将结束一个表格，注释行除外
- 单元格数目没有限定，不同的行可以包含不同数目的单元格
- 目前尚无法实现“行跨越”
- 如果目标格式不支持表格，则表格域视为完全引用域

5.14 原始文，原文行和原文域

- 描述：用以“保护”某些文字不被解析，其内部的标记和宏均不会处理
- 属性：非多行，非空格自由，无对齐，无嵌套
- 可包含：无
- 语法- 原始文：
 - 连续的两个双引号包围文字，如：`"like this"`
 - 标记必须与内容紧密详解（不能有空格）
- 语法- 原文行：
 - 以3个相继的双引号开始的行，`""" like this`
 - 双引号必须位于行的第一列，不能有前导空格
 - 双引号后接一个空格，以分隔开标记与文本
- 语法- 原文域：

- 第一行为3个相继的双引号，下一行开始为单行或多行文本，最后一行为3个相继的双引号作为结束
- 标记不能有前导和后继空格

- 细节:

- 内部的标记不会解析
- 内部的宏不会解析
- 如果到达源文件末尾 (EOF)，尚未闭合的原文域自动关闭

第六章 宏

宏是一些特殊的关键词，在转换的时候进行展开和替换。宏可以用来插入动态的信息，比如源文件的编辑日期或其他信息。

宏的格式为：双百分号%%引导，后面紧跟着宏名，如：`%%date`。某些宏可以后接一对圆括号，里面是可选的格式选项字符串，诸如：`%%date(%Y-%m-%d)`，格式选项字符串由百分号%后接一个字符构成，如果未给出格式选项，则使用缺省的格式。

宏名	展开为	默认格式
<code>%%date</code>	当前日期	<code>%Y%m%d</code>
<code>%%mtime</code>	源文件修改时间	<code>%Y%m%d</code>
<code>%%infile</code>	源文件路径	<code>%f</code>
<code>%%outfile</code>	输出文件路径	<code>%f</code>
<code>%%toc</code>	展开为目录表	-

一般规则：

- 宏名不区分大小写，所以`%%date`、`%%DaTe`和`%%DATE`完全等效
- 宏放置于文档的头部和体区域都是有效的，只有`%%toc`例外，它只能放在主体域区域。
- 在设置域内的宏，意味着设置域结束，体区域开始
- 宏可以位于一行的任何位置，同一行可以包含各种不同的宏（`%%toc`除外，它必须独占一行）
- 链接和插图标记也可以使用宏（`%%toc`除外）
- 标题、完全引用和原文域内的宏不会解析

一个完整的例子（粗体部分由宏展开得到）：

本文是《Txt2tags用户指南》，由源文件**txt2tags-userguide-zh.t2t** 通过Txt2tags转换得到，输出文件名**tex**。转换完成于**2009-03-23 15:07:29**，而最后一次修改则是在**2009-03-23 15:07:25**。源文件和目标文件均位于目录**t2t-userguide-cn**下。

（译者按：宏%%infile和%%outfile再Windows下无法正常展开。）

6.1 %%date

宏%%date 将被替换为当前的日期和时间，可以用于在文档的头部或尾部注明生成的时间。如果要得到源文件最近一次编辑的时间，请参见%mtime 宏一节。

宏展开时可以有若干不同的显示格式，完整的列表请参看Python主页，以下是其中最常用的几种：

格式	描述
%a	星期名的缩写
%A	星期名的全称
%b	月份名的缩写
%B	月份名的全程
%c	日期和时间
%d	一个月中的第几天（01-31的十进制数）
%H	24小时制的小时数（00-23的十进制数）
%I	12小时制的小时数（01-12的十进制数）
%m	一年中的第几个月（01-12的十进制数）
%M	分钟数（01-59的十进制数）
%p	本地格式的上午或者下午
%S	秒数（01-59的十进制数）
%x	本地的日期代称（原文：Locale's appropriate date representation）
%X	本地的时间代称
%y	两位数的年份表示（00-99的十进制数）
%Y	完整的年份表示
%%	百分号“%”

示例：

宏	->	对2009, Mar 23 at 15:07 的展开效果
%%date(Converted on: %c)	->	Converted on: Mon Mar 23 15:07:29 2009
%%date(%Y-%m-%d)	->	2009-03-23
%%date(%I:%M %p)	->	03:07 PM
%%date(Today is %A, on %B.)	->	Today is Monday, on March.

6.2 %%mtime

宏%%mtime 将被展开为源文件最近一次修改的时间。它可算是宏%%date的“姊妹篇”，其格式指令与之完全相同。

举一个例子，本文的源文件最近一次修改是在Mon Mar 23 15:07:25 2009，这一日期由%%mtime(%c)展开而来。

6.3 %%infile

宏%%infile展开为源文件在系统中的路径信息，可用于生成html中“查看本页面源文件”的链接。

该宏支持的格式指令如下：

指令	描述	本文源文件的宏展开输出
%f	文件名	txt2tags-userguide-zh.t2t
%F	文件名（不含扩展名）	txt2tags-userguide-zh
%e	文件扩展名	t2t
%p	文件绝对路径	/home/rookie/tmp/editing/t2t-userguide-cn/txt2tags-userguide-zh.t2t
%d	文件所在目录	/home/rookie/tmp/editing/t2t-userguide-cn
%D	文件父目录	t2t-userguide-cn
%%	百分号	%

示例：

源代码	->	展开结果
This Guide parent dir is %%infile(%D).	->	This Guide parent dir is t2t-userguide-cn.
I do use the %%infile(%e) file extension.	->	I do use the t2t file extension.
[See the source %%infile]	->	See the source
Converted to XHTML, I'll be %%infile(%F).xhtml	->	Converted to XHTML, I'll be txt2tags-userguide-zh.xhtml

注：若源文件为标准输入STDIN，则宏展开为“-”。

6.4 %%outfile

宏%%outfile将展开为目标文件的路径，可用于将文件名包含于文档的头部和主体内。它可算是[宏%%infile](#)的“姊妹篇”，格式指令也完全相同。

示例：

源代码	->	展开结果
You are reading the %%outfile file.	->	You are reading the txt2tags-userguide-zh.tex file.
txt2tags -t %%outfile(%e) -i %%infile -o %%outfile	->	txt2tags -t tex -i txt2tags-userguide-zh.t2t -o txt2tags-userguide-zh.tex

注：若输出为标准输出STDOUT，则宏展开为“-”。

6.5 %%toc

宏%%toc将展开为目录表。你可以把它放在文档主体域内的任何地方，甚至不止使用一次，比如你可以把它放在文档的末尾。本指南就是使用这个宏来放置了目录表。

与其他的宏不同，%%toc没有格式字串，而使用规则也有不同：

- 只在文档主体域中可用
- 必须独占一行（前后可以有空格）
- 必须与命令行选项`--toc`配合使用，否则将被忽略
- 如果使用了`%toc`，缺省的目录表位置和格式将被覆盖

第七章 设定

设定项位于文档的设置域内，在转换执行时生效。设定项均为可选的，普通用户完全可以不使用它们而完成所有工作；然而，一旦你学会使用设定项，我打赌你一定会上瘾，因为它们实在太好用了！

设定行是特殊的注释行——由一个“!”引导，以此区别于普通的注释行。设定项的语法很简单，由关键词和价值构成，二者由一个冒号（“:”）分隔。

%! 关键词: 值

语法细节：

- “%!”要写在一起，中间不能有空格，且置于行首
- 关键词和分隔符之间可以有空格
- 关键词和价值均不区分大小写

规则：

- 只有位于设置域内的设定项才生效，而位于主体域内的设定项将被视为普通的注释文本。
- 若相同关键词的设定项在设置域内出现多次，则使用最后一次的设定。例外：options, preproc和postproc，这三个设定是累积的。
- 关键词写错的设定被视为普通注释文本。
- 设定项的优先级高于RC文件，但低于命令行选项。

7.1 %!Target

用来定义缺省的目标文件格式，如：

```
%!target: html
```

此时用户只需调用命令：

```
$ txt2tags file.t2t
```

即可将文档转换为预设的目标格式。

本设定不支持指定可选的特定目标格式（原文：optional target specification），比如：`%!target(tex): html` 是无效的。

7.2 %!Options

每次都输入冗长的命令行选项既累人又容易出错，使用Options设定项，用户可将转换选项与源文件放在一起，这同时也利于确保文档总能以相同的选项和相同的方式转换。

书写这些选项时请确保不要有语法错误，就像使用命令行方式一样。当然，对程序“txt2tags”的调用、目标格式的指定和源文件名就不必写进来了。

比如，假设你在命令行进行如下的转换：

```
$ txt2tags -t html --toc --toc-level 2 --enum-title file.t2t
```

把这些选项写入源文件：

```
%!target: html
```

```
%!options(html): --toc --toc-level 2 --enum-title
```

现在你只需简单的使用命令“`txt2tags file.t2t`”即可达到同样的目的，这样也便于你在编辑器中执行转换。比如，在Vi里面可以这样调用：

```
:!txt2tags %
```

7.3 %!Encoding

Encoding选项为非英语用户所需，因为他们需要使用本地语言的某些特殊字符以及细节，因此目标文件的**字符集**需要自行指定（如果允许的话）。

有效的Encoding值与HTML所使用的**字符集**名称相同，比如：*iso-8859-1*、*koi8-r*等。如果你对此不确定，可以参见[完整的字符集列表](#)。

LaTeX使用的是字符集的别名，这对于用户来说并不是问题，因为txt2tags会替你完成名称的转换，比如：

txt2tags/HTML	>	LaTeX
windows-1250	>>>	cp1250
windows-1252	>>>	cp1252
ibm850	>>>	cp850
ibm852	>>>	cp852
iso-8859-1	>>>	latin1
iso-8859-2	>>>	latin2
koi8-r	>>>	koi8-r

就算设定的值txt2tags无法识别也可以通过，反正由用户自行定制。

7.4 %!PreProc

PreProc选项在源文件读入后还没解析前，进行“查找和替换”的工作。

可以用来定义一些常用词的缩写，如：

```
%!preproc JJS           "John J. Smith"
%!preproc RELEASE_DATE "2003-05-01"
%!preproc BULLET       "[images/tiny/bullet_blue.png]"
```

源文档中的一行：

```
Hi, I'm JJS. Today is RELEASE_DATE.
```

txt2tags在进行转换时会将其视为：

```
Hi, I'm John J. Smith. Today is 2003-05-01.
```

这相当于对源文件调用外部Sed/Perl进行过滤，然后传给txt2tags:

```
$ cat file.t2t | preproc-script.sh | txt2tags -
```

txt2tags在PreProc处理结束后开始对源文件进行解析。

注：预处理仅仅对主体域进行，不会包含头部域和设置域。

7.5 %!PostProc

PostProc选项在txt2tags进行解析和处理结束后，对于转换后的内容进行“查找和替换”的工作。

可以用来对生成的文档进行进一步加工和微调，比如修改标记、添加额外的文本和标记，等等。比如：

```
%!postproc(html): '<BODY.*?>' '<BODY BGCOLOR="green">'  
%!postproc(tex) : "\\newpage" ""
```

作用是讲HTML文件的背景颜色改为绿色，以及移除LaTeX文件里的分页符。

这相当于对转换后的内容调用外部Sed/Perl进行过滤，再生成目标文件：

```
$ txt2tags -t html -o- file.t2t | postproc-script.sh > file.html
```

在引入这一特性前，我们往往得使用一些小脚本来“调整”txt2tags处理的结果，这些脚本实际上多是大量的sed（或类似工具）替换命令。现在，这些繁琐的替换字符串可以放在文件中，并使用Python强大的正则表达式机制来进行匹配和处理。

7.6 %!Style

- 用于在HTML和XHTML目标文件中指定样式表文件。
- 用于在LaTeX目标文件中使用\usepackage加载宏包。
- 效果等同于命令行选项--style。
- 命令行选项--style 优先级高于设定项%!style。

7.7 针对特定的目标格式进行设定

除了`%!target`之外的所有设定项均可作用于某一特定的目标格式，语法如：`%!key(target): value`。这使得用户可以针对不同的目标格式进行不同的设置。

这一特性对所有支持它的设定项都有效，只不过在Pre/PostProc过滤器中显得尤其有用。举个例子，你可以对HTML和LaTeX文件设定不同的样式：

```
%!style(html): fancy.css
%!style(tex) : amssymb
```

这一特性使得用户便于对转换后的文件进行微调：

```
%!target: sgml
%!options(sgml): --toc
%!options(html): --style foo.css
%!options(txt ): --toc-only --toc-level 2
```

在这个例子中，缺省的目标格式为SGML，并且生成目录表。如果用户调用命令“`txt2tags -t html file.t2t`”，则仅仅使用针对HTML定义的选项，生成的文件会使用外联的样式表“`foo.css`”，并且不生成目录表。

7.8 关于PreProc和PostProc过滤器的一些细节

- 按行进行“查找和替换”（类似与Sed）。
- 后定义的过滤器不会覆盖先定义的过滤器，而是累加的。因此你可以定义许多过滤器，它们会按照定义的顺序依次执行。
- 与其他的设定项不同，一般的和针对特定目标格式的过滤器均会被执行。在下面这个例子中，两个过滤器均会作用于HTML目标文件上：

```
%!postproc      :   this   that
%!postproc(html):   that   other
```

- 过滤器必须指定两个参数。
- 特定的转义字符，如换行符`\n`和制表符`\t`，均将被解释。

- 利用空字符进行替换，实现删除文本的功能：

```
%!postproc: "undesired string" ""
```

- 使用PostProc替换标签时最好总是指明对应的目标格式，以避免出现问题：
“%!PostProc(target): <thsi> <that>”
- PreProc在行读入后立即起作用，而PostProc则是在所有的解析和处理过程结束后起作用，他们起作用的位置相当于：

```
$ cat file.t2t | preproc.sh | txt2tags | postproc.sh
```

- 过滤器的“查找”过程使用正则表达式，而不是常规字符串。不理解什么是正则表达式也没关系，你只需记得某些特殊字符要使用反斜线“\”来进行“转义”就够了，这些特殊字符包括：

```
\* \+ \. \^ \$ \? \( \) \{ \[ \| \\
```

- 支持Python的正则表达式（与Perl的正则表达式相似）。比如：在HTML中把标签“B”都替换为标签“STRONG”：

```
%!postproc(html): '(</?)B>' '\1STRONG>'
```

- 过滤器参数有3种方式进行传递：
 1. 单个未加引号的单词，诸如：FOO（不含空格）
 2. 放在双引号中的字符串，诸如：“FOO”
 3. 放在单引号中的字符串，诸如：'FOO'
- 如果模式表达式包含双引号，将它放在一对单引号中保护起来，反之亦然。一些合法的模式表达式示例：

```
%!postproc: PATT REPLACEMENT
%!postproc: "PATT" "REPLACEMENT"
%!postproc: 'PATT' 'REPLACEMENT'
%!postproc: PATT "REPLACEMENT"
%!postproc: "PATT" 'REPLACEMENT'
```

第八章 黑魔法

本章不推荐初级用户阅读。我们将在txt2tags的过滤器中使用一些复杂的模式和正则表达式，以此来实现一些“奇怪”的功能。

请注意：执行以下过程前建议谨慎，它们可能坏事，甚至于源文件中的某些文本在转换中会丢失，而不会出现在目标文件中。请在仅当你确实需要并且明白自己在干什么的情况下才使用这些技巧。

注：过滤器是强大却也危险的特性！错误的过滤器会差生意想不到的结果。

请切记！

8.1 使用%!PostProc插入多行文本（比如CSS规则）

过滤器中，用于替换的模式表达式可以包含多行（以\n作为换行符）。这便于在HTML目标文件中插入少量几行CSS规则，而毋需创建一个独立的外联样式表：

```
%!postproc: <HEAD>      '<HEAD>\n<STYLE TYPE="text/css">\n</STYLE>'  
%!postproc: (</STYLE>) 'body    { margin:3em           ;} \n\1'  
%!postproc: (</STYLE>) 'a       { text-decoration:none ;} \n\1'  
%!postproc: (</STYLE>) 'pre,code { background-color:#ffffcc ;} \n\1'  
%!postproc: (</STYLE>) 'th      { background-color:yellow ;} \n\1'
```

如此，其余的过滤器通过替换一个插入的字符串而与第一个过滤器联系起来。现在，原来简单的一个“<HEAD>”变成了：

```
<HEAD>
<STYLE TYPE="text/css">
body    { margin:3em          ;}
a       { text-decoration:none ;}
pre,code { background-color:#ffffcc ;}
th      { background-color:yellow ;}
</STYLE>
```

8.2 使用%!PreProc创建“特定目标类型”的内容

有时候你打算在某一特定格式的目标文件中插入一些文本，而其他的格式则不作改变，这一目标可以通过PreProc的一些小技巧来实现。

方法是：将这些文本作为注释插入源文件中，做上一些标记，而使用针对特定目标格式的过滤器将它们“反注释”掉。

举个例子来说，我们想要在HTML输出中加入一个额外的段落，于是把这些文本写成特殊的注释行，比如：

```
%html% This HTML page is Powered by [txt2tags http://txt2tags.sf.net].
%html% See the source TXT file [here source.t2t].
```

这些行以%开头，属于普通的注释行，在转换时被忽略掉。但是当你添加一个过滤器之后：

```
%!preproc(html): '^%html% ' ''
```

通过显示的声明，这一过滤器仅仅在HTML作为目标格式时被应用。于是开头的百分号被移除，这些行被“激活”，不再是注释。

8.3 使用%!PreProc改变txt2tags标记

对于某些正则表达式控来说，他可以自行定制源文件的语法，将txt2tags的默认标记修改为自己喜欢的标记。

举个例子，位于行首的制表符是默认的引用标记，如果用户不喜欢，或者其编辑器关于制表符有奇怪的设定，他可以定义一个新的标记来标记引用域。比如他选择用位于行首的“>>>”来作为引用标记，那么他只需定义一个简单的过滤器：


```
%!PreProc: '>>>' '\t'
```

源文件里面的引用文本形如：

```
>>> This is a quoted text.  
>>> The user defined this strange mark.  
>>> But they will be converted to TABs by PreProc.
```

在解析之前，这些诡异的“>>>”符号会被自动的替换为制表符，txt2tags则识别出引用标记。

请注意：极端的PreProc规则会改变整个标记语法，甚至会在标记之前产生冲突，因此在使用前务必小心谨慎。

第九章 软件历史

Txt2tags的第一个公开发表的版本version 0.1 是在2001年7月，然而其原型则比这个时间早上至少一年……

本章将带领你纵览软件从最初的原型到当前版本的发展过程。

9.1 1999年1月：史前历史

作者曰：“我最初关于编写一个文本转换工具的尝试始于1999年，起初只是一个简单的、功能有限的Bourne Shell脚本，用来将标记文本转换为HTML页。哦是的，不过是又一个txt2html的工具，已经有无数人干过这个事情了……简单的说，它只能识别简单的标记比如***bold***、*/italic/*、under，以及转义典型的HTML特殊字符如< & >。不怎样对吧，但是！我还年轻呢，;)”

9.2 1999年6月：依然是史前历史

作者还想说：“几个月过后，关于SGML的宣传铺天盖地的淹没了我所任职的Conectiva 公司，于是原来的txt2html变成了txt2sgml脚本。那时我正在学习SED，因此txt2sgml就成了一个110行的Bourne Shell脚本，其中包含了许多SED命令。”

注：SED是UNIX下的流编辑器，用来进行文本的自动化编辑。

这一改进后的SGML版本支持更多的文档结构，比如列表、完全引用域等。在下面这个示例中，已经可以看到txt2tags标记的雏形了：

```
* This was a bold line (BOLD line oriented? Well...)
```

```
--
- bullet list was very similar to txt2tags list
- but with these -- to begin and close a list
--
```

```
-----
Verbatim text was delimited by the =-- pattern.
The other ----- were just cosmetic.
-----
```

依然不怎样对吧。但是！接下来就是划时代的一步了……

9.3 2000年八月：不再是史前历史了

作者又说：“一年以后，那时我完全爱上了SED，于是我将txt2sgml.sh 脚本重写，变成了一个包含350行的纯粹的SED脚本。一些激动人心的特性被加入进来，比如：subsection、URL识别和子列表（列表嵌套）。我用了将近一整年的时间来使用和改进这个脚本。”

一个txt2sgml.sed的示例文件：

```
* Hey, here are the first 3 magic lines
* The document title / author / date
* But they required those asterisks at the beginning
```

```
MAIN TITLE
```

```
Titles were made by uppercase-only lines. Subtitles were identified by
leading spaces. Each space denoted a new sublevel. The beautifiers:
*bold*, **strong**, "italic" and 'typewriter'.
```

```
- lists
+ sublists
  = and subsublists (by identifier, not indentation)
```

Two blank lines to close lists. Links as `www.example.com` and e-mails were recognized automagically by regular expressions. And there was a strange image mark:

```
%image: path/to/image.jpg
```

9.4 2001年5月：Python化和多目标格式的想法

现在是谁在说话呢？“那时我开始动笔写《[Regular Expression book](#)》这本书。我仍然使用`txt2sgml.sed`的标记文本格式，将文件转换为SGML格式，然后用`sgml2html`工具转换为HTML，再在浏览器中“快速的”检查结果。因为“快速”与“`sgml2html`”并不相符，我将自己的SED脚本修改为`txt2html.sed`工具，用来直接产生HTML文档……可是出版商使用Adobe PageMaker来排版书籍，这可苦了我这样的Linux用户。幸好，我发现PageMaker使用的是一种类似于HTML的标记语言，于是最后我写了三个相似的SED脚本，用来将文本转换为SGML、HTML和PageMaker，此外还用一些Shell脚本来生成目录表，以及进行一些后处理。书写到一半的时候，我产生了将这些单一小工具合并起来的想法，并且选择用Python来完成这一工作。于是，TXT2TAGS诞生了。”

9.5 2001年7月：0.x系列：txt2tags的处女秀（公开发布）

是的，就是他在说：“我的那本书的发行时间（7月31日），与txt2tags的第一个版本0.1版的发布时间（7月26日）只相隔几天——其中一个依赖于另一个，却是一起成长的。除了SGML、HTML和PageMaker之外，其他的目标格式也一并实现了：MoinMoin、MagicPoint以及纯文本。接下去的一年多时间里，0.x系列的更多版本发布了，程序也逐渐长大：新的目标格式UNIX手册页，宏`%%date`，表格支持，基于网页的使用接口，图片智能对齐和生成目录表，这些功能一一实现。在0.2版本中，一个Shell脚本被添加进来作为外包，进行文件操作和命令行选项的处理——这样做的原因无非是因为我是个Shell老手，却是一个Python菜鸟。一直到0.9版本，txt2tags才又回到100%的Python代码。”

9.6 2002年9月：1.x系列：成长

你是否已经没耐心听下去了？“txt2tags的设想被证明是好的，我决定认真的来对待之。需要进行的下一步是散发程序，将它告诉全世界：我需要文档！于是，程序的网站上线了，邮件列表（英语和葡萄牙语）建立起来，用户指南也诞生了。用户群成长起来，并且为项目作出许多贡献。是的，我想，这证明这玩意儿是有用的。1.x版本添加的新特性包括：图形用户接口（GUI），Windows和Mac系统的移植，LaTeX目标格式，`%!style`设定项，`include`命令和强大的Pre/Post过滤器。”

9.7 2004月7月：2.x系列：成熟

好吧，我也说得口干舌燥了：“成长是奇特而又艰难的。我说过我是个Python菜鸟对吧？现在我变强了，然而过往的错误累积起来，于是无可避免的，源代码必须动一次大手术。这一次重写使得一些标记的兼容性变差了，于是我得创建一个升级脚本。这一次耗费了很长的时间，但是2.0版本终于还是发布了，它带来了数不清的新特性，诸如：XHTML目标格式，W3C验证码，`i18n`，以及RC文件。翻译团队将程序本身以及文档翻译成多种语言。LOUT目标格式被添加进来，此外还有新的宏：`%%mtime`、`%%infile`、`%%outfile`，以及`%%toc`。未完待续……”

The End ([查看源文件](#))

译文最后一次修订于：2009-03-23 15:07:25

