

rtf2latex2e Documentation

version 2.0.1

Ujwal S. Sathyam
Scott Prahl

31 Jan 2011

1 Introduction

rtf2latex2e is an RTF \rightarrow L^AT_EX converter that takes as its input RTF files produced by Microsoft Word and comparable word processors such as Star Office and generates a T_EX-able “*.tex*” file. It has the capability to handle fairly complex RTF files containing figures, tables, and equations. *rtf2latex2e* is written using standard *C* and should compile on any platform supporting a *C* compiler. It has been tested on Mac OS X, Linux (Intel), and Windows.

rtf2latex2e uses the generic RTF reader framework by Paul DuBois. The framework is a general purpose tool for processing RTF files and may be configured in a well-defined manner to allow it to be used with a variety of writers generating different output formats. This provides a method for generating RTF-to-XXX translators. Paul seems to have stopped developing the reader code, therefore Ujwal Sathyam (in 1999) had to adapt it quite a bit to handle the latest version of RTF. More code evolution took place during development of version 2.0.

If you expect a WYSIWYG reproduction of your RTF file, you may be disappointed. Our main concern has been translating the essential features of the RTF file such as characters, figures, tables, and equations. Visual formatting such as ruler positions, tabs (until I figure out a good way of doing this), paragraph indentations, and other fluff has been largely ignored. The translated L^AT_EX 2_ε file will require some manual editing to put the finishing touches. We just want to make that task a little easier. In our opinion, expecting a WYSIWYG reproduction is not practical and misses the point entirely.

2 Installation

The source code is available at SourceForge as well as compiled binaries for Windows systems. Users of MacOSX, Unix, and linux systems will have to compile the source code, which is a fairly straightforward process.

2.1 Unix

Change to the *rtf2latex2e* directory and type:

```
prompt> make
```

This will compile the sources and create a binary called **rtf2latex2e**. You can choose to install the binary in a convenient location. If you want to install into the default directory */usr/local/bin*, Installation is done by:

```
prompt> sudo make install
```

otherwise edit the `$PREFIX` variable in *Makefile* to install the files where you would like.

2.2 Mac OS X

Just follow the above direction, but an added feature is available. Since so many Mac RTF files come with embedded *.pict* images, it is possible to include automatic translation of *.pict*→*.pdf* in the binary by uncommenting (removing the *#*) these lines in the *Makefile* before compiling

```
#CFLAGS :=$(CFLAGS) -m32 -DPIC2PDF
#LDFLAGS :=$(LDFLAGS) -m32 -framework ApplicationServices
```

On a Mac, this will causes *rtf2latex2e* to automatically convert the PICT images to PDF images and change the `\includegraphics` file names.¹ Obviously the proper framework is not available on other systems so this option is commented out by default.

2.3 Windows

Windows users get a pre-compiled binary of *rtf2latex2e* to be run from the MS-DOS prompt. Just run it with the RTF file to be converted as the argument. To compile, uncomment the following line in the *Makefile*

```
PLATFORM?=-DRTF2LATEX2E_DOS # Windows/DOS
```

3 Using *rtf2latex2e*

This is a command-line program, which may be run with a variety of options

¹It is understandable that has deprecated the PICT format and is pushing PDF. PICT is an old format, true, but I am always stumbling across legacy files with PICT images. The only way to properly convert the PICT vector image to a PDF vector images is to use the `QDPictCreateWithURL()` call. This function is not available when compiling 64-bit binaries and therefore the `-m32` option is required while compiling an linking.

```

prompt> rtf2latex2e [options] file.rtf

prompt> rtf2latex2e foo                convert foo.rtf to foo.tex
prompt> rtf2latex2e -p 33 -t 4 foo      minimal latex mark-up
prompt> rtf2latex2e -e 15 foo           debug failed eqn conversion
prompt> rtf2latex2e foo-eqn003.eqn      debug third equation
prompt> rtf2latex2e -D foo              put files in foo-latex dir
prompt> rtf2latex2e foo.rtf             convert to foo.rtf/TXT.tex

```

3.1 The trivial options

The trivial options are

```

-h          display help
-v          version information

```

and should require no explanation, so I won't bother.

3.2 The new directory option

The option to create a new directory is interesting and useful

```

-D          make a new directory for latex and extracted images

```

because RTF files often contain dozens of image files. All these files are extracted during conversion and this will clutter your current working directory. The `-D` option avoids this and create a single location for all the new latex files by locating all of them in a new directory. For example

```

prompt> rtf2latex2e -D file.rtf

```

creates a new directory called *file-latex* and places the converted latex file *file.tex* inside. The file names are adjusted so that you can `cd` to that directory and type

```

prompt> cd file-latex
prompt> pdflatex file.tex

```

to create a PDF file, for example.

3.3 The preference path option

This is important.

```

-P path/to/preferred/preference/directory

```

Probably more important than it should be. *rtf2latex2e* reads a bunch of font encodings *rtf-encoding.**, *rtf-ctrl*, and *latex-encoding* before it can convert anything. These are all located in the preference directory.² In addition to these

²Arguably, all these files could be compiled into the binary. Certainly, the only file that a user might actually modify is the *latex-encoding* file, but now that we have UTF8, why bother?

critical files, some user configurable files *r2l-head*, *r2l-map*, and *r2l-pref* are also read from this location.

There are three ways to specify the location of this directory,

- using the default compiled in location as specified in the *Makefile*. As shipped, *rtf2latex2e* defaults to */usr/local/share/rtf2latex2e*.
- using the shell environment variable `$RTFPATH`. This can be a handy thing to put in your *.bash_profile* file if you want to use some other directory than the default one above. This option is searched before the above.
- using the `-P path` option. This gets tried first. For example, during development, this option is used extensively because the local *pref* directory often differs from the system installed version.

3.4 The text conversion option

Obviously text can have a bunch of formatting applied to it. **color**, **bold**, *italic*, underlined, font selection, and font size. Basically, the specific font sizes specified in the RTF file get mapped to the nearest relative latex sizes like `\small` and `\large`. All these can be turned off with the `-t` option. Just add the numbers together to get a combination of items

<code>-t #</code>	text conversion options
<code>-t1</code>	font size
<code>-t2</code>	font color
<code>-t4</code>	font formatting
<code>-t8</code>	replace tabs with spaces

Thus `-t 6` will cause *rtf2latex2e* to convert formatting and color, but ignore size changes and replace all tabs with spaces.

Disabling font size conversion is often a good idea because L^AT_EX usually makes reasonable assumptions about the font size. It is a nuisance if every paragraph is bracketed with `{\Large ... }` because the RTF document was written in 14 point font.

No support for font changes exists in *rtf2latex2e* version 2.0 — maybe in another ten years.

3.5 The paragraph conversion option

Paragraphs are nasty beasts. Just consider indenting. In an RTF file, indenting can be achieved

- by using the `\fi` tag to indicate the indentation of the first line.
- by tabbing to the first tab stop
- by using spaces

- by using a style

Well, that was not as bad as I remembered it being. But that is just the first indent. There is also line spacing before and after the paragraph, line spacing, left and right margins, and overall alignment.

-p #	paragraph conversion options
-p1	'heading 1' style -> '\section{'}
-p2	indenting
-p4	space between paragraphs
-p8	line spacing
-p16	margins
-p32	alignment

If all the options are on then a ton of cruft is added to the latex document. For example, using -p 63 each paragraph might come out looking like

```
\vspace{6pt}
\leftskip=15pt
\parindent=-15pt
This architecture allows the reader to remain constant,
so that different translators can be built by supplying
different writer and driver code....
```

Although, the L^AT_EX'ed file looks great, it is a nuisance to edit. Since presumably, this is the only reason that you are converting to L^AT_EX, you probably don't want the extra markup. I personally like -p 33 which is the default for the -n option described below.

I should describe the -p 1 option, but I really am tired of this document. So I won't.

3.6 The equation conversion option

The most common source of the RTF files is Microsoft Word. Equations in Word are created in Equation Editor (MathType), and when saved into an RTF file, the equation is saved as MTEF embedded in an OLE object. *rtf2latex2e* uses the cole library³ to extract the embedded equations from the OLE structured format. The equation is then converted into L^AT_EX format. If everything goes well, then the conversion can be surprisingly good. If the native equation conversion fails, or if the option to convert equations is disabled, *rtf2latex2e* reads that picture and outputs the equation as a picture file.

Not surprisingly, sometimes the conversion fails. One of the major bug fixes in version 2.0 was to properly extract the MTEF record of the equation from the encoded OLE. During processing, *rtf2latex2e* saves this as a separate file. For example, the thirty-third equation when translating *file.rtf* might get saved as *file-eqn033.eqn*. This file is then processed to produce a string that is inserted

³originally available in 2000 from <http://arturo.directmail.org/filtersweb>

into the converted latex document. If something goes wrong with processing this particular equation then it is very handy to be able to access this equation directly. So how do you know it was equation 33 and not equation 19? Use `-t 15` according to

<code>-e #</code>	equation conversion options
<code>-e1</code>	convert to latex
<code>-e2</code>	insert image
<code>-e4</code>	keep intermediate eqn file
<code>-e8</code>	insert eqn file name in latex document

Then in the converted document (or more likely the partially converted *file.ltx* document, insert a final `\end{document}` so that you can see the converted latex file. The `-e 2` allows you to visually compare the converted latex with an image of what it should look like. The `-e 8` option will insert the name of the associated *.eqn* file. Then it is just an issue of debugging the equation using

```
prompt> rtf2latex2e file-eqn033.eqn
```

and figuring out which one of the MTEF commands is getting mangled.

3.7 Two useful options

<code>-b</code>	best attempt at matching RTF formatting
<code>-n</code>	natural latex formatting ... easiest to edit

This is what I do

```
prompt> rtf2latex2e -n test.rtf
```

which gives the best balance of minimal mark-up with conversion fidelity.

4 The Preference files

4.1 r2l-pref

rtf2latex2e reads a preference file *r2l-pref* where you can specify various options

- `preambleFirstText`
- `preambleSecondText`
- `preambleDocClass`
- `outputMapFileName`
- `pageWidth`
- `pageLeft`
- `pageRight`

- `convertPageSize`
- `convertParagraphStyle`
- `convertParagraphIndent`
- `convertInterParagraphSpace`
- `convertParagraphMargin`
- `convertParagraphAlignment`
- `convertLineSpacing`
- `convertTextSize`
- `convertTextForm`
- `convertTextNoTab`
- `convertTextColor`
- `convertHypertext`
- `convertEquation`
- `convertAsDirectory`
- `convertTableName`
- `convertPict`

The options are self-explanatory. If not, then read the comment text in *r2l-pref*. These preference selections are overridden by command line options.

4.2 latex-encoding

The default charset for the latex file is Unicode, the text file is written using UTF8. This is specified by the file *latex-encoding*. Others encodings are possible but I think they should all be dropped. Unicode works with L^AT_EX so why bother with any other representation. Consider the others deprecated.

4.3 r2l-head

rtf2latex2e also reads a file (if present) called *r2l-head*. In this file, you can specify any additional packages that you want to use in your L^AT_EX file, e.g., a babel hyphenation package or a font encoding. The contents of this file are just copied into the preamble of the L^AT_EX file.

4.4 r2l-map

rtf2latex2e also reads a file (if present) called *r2l-map*. In this file, you can customize mappings for section headings. Other style conversion may be possible, but it should be considered experimental (aka, flakey) at the moment.

5 Features

rtf2latex2e is designed to convert journal articles, reports, and letters written in Microsoft Word. That means I would like it to handle the following:

5.1 Figures

rtf2latex2e can read figures of format PICT, EMF, WMF, PNG, and JPEG embedded in RTF files. These are the most common formats encountered in RTF files. When *rtf2latex2e* encounters an embedded figure, it reads out the figure into a separate file. The output format of the figure is the same as the format it was embedded in. On the Mac, *rtf2latex2e* will attempt to convert PICT to PDF files.⁴

5.2 Tables

Yeah, it does tables!! However, this is the weakest link in the chain and the messiest part of the code. This is largely due to the fact that RTF does not have a separate “Table” group. It is also due to the fact that TeX likes to know in advance the number of columns in the table, and RTF does not tell us that. Ujwal spent a lot of time to support tables to this extent. Some of the test files have tables in them. To get an idea of the type of tables that *rtf2latex2e* can handle, take a look at test/table.rtf. The package *longtable.sty* is used for generic table handling to take care of tables that span several pages.

Tables have way more mark-up than desired.

5.3 Character mapping:

Character mapping is largely complete for the most common latin scripts. Characters are translated by referencing character set maps and the output map file *latex-encoding*. The platform and locale dependent character set, e.g. latin-2 (Eastern European), is converted to an internal platform-independent representation by reading the appropriate character map file, in this case *rtfencoding-cp1250*. For example, character 192 (hex 0xC1) represents Á in the latin-2 character set. *rtf2latex2e* uses *rtfencoding-cp1250* to translate this character to *Aacute*. Finally, *Aacute* is translated into the L^AT_EX representation `\’{A}`

⁴It used to be that all images were converted to EPS files. This capability was removed because it other utilities do it better, I prefer EPS, and *pdflatex* most formats directly. It is better that you convert files as you would like, rather than providing a poor conversion to EPS.

using the file *latex-encoding*.⁵ This two-step character mapping allows for easy addition of support for additional character sets such as latin-5 (Turkish) or cp1251 (Russian).

5.4 Test files

There are several test files in the *test* directory of the *rtf2latex2e* distribution that demonstrate the capabilities of the converter. You can also download a larger set of test files to see how the program behaves. These test files are in a tarred gzipped archive in the same place where you downloaded the *rtf2latex2e* distribution. “*RTF-test-files*” contains several RTF files that have been successfully tested on *rtf2latex2e*. By success, I mean that *rtf2latex2e* processes the RTF file without any problems (except maybe giving a few warnings) and produces a “.tex” file that is L^AT_EX 2_ε-able!! It does not mean that the L^AT_EX 2_ε output file will look exactly the same as the RTF input file. In fact, most of the time, it will not. Some features like I do not care to convert, others like Unicode support will be implemented in future versions.

6 Acknowledgements

I would not even have attempted this thing had it not been for Paul DuBois’ very nicely designed RTF tool. I did not have to bother with parsing the RTF tokens and understanding it. All I had to do was write code to act upon the token. Thanks, Paul, for simplifying it.

Steve Swanson of Mackichan Software, makers of Scientific Word and Workplace, contributed the early equation converter code. With this ability, *rtf2latex2e* has advanced to version 1.0. Hopefully, this essential feature addition along with *rtf2latex2e*’s other capabilities will make this program the *de facto* tool for converting word processor documents to L^AT_EX 2_ε.

7 Legalese

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.

The initial equation converter capability was provided by Steve Swanson from Mackichan Software, makers of Scientific Word and Workplace.

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. If you format your hard disk, or do anything else inconvenient, it’s not my fault.

The reader part of this code is copyright Paul DuBois.

⁵Although now it is just translated to the Unicode value 0xE1. Anyhow, you get the idea.

If you make any modifications that you think makes this program better, please send me the modifications so that I can incorporate them in later versions. Please do not distribute modified versions. I plan to keep working on this project, and anybody is welcome to help.