Some documentation of the rtf2LaTeX filter

The filter rtf2LaTeX consists of two modules, the reader (just called reader) and the writer (called rtf2LaTeX), and needs two files, one with the extension "code" and the other with "land". (see later on)

The reader

is written by Paul DuBois and consists except besides some initialition stuff of two major parts: The procedures RTFReadToken   and   RTFRouteToken. RTFReadToken reads unformatted text, "control words", "control symbols" and "groups" from an rtf-file and creates tokens depending on what it has read.

Tokens are associated with up to three classification numbers:

Class number: Broadest (least detailed) breakdown.
Major/minor numbers: Within their class, tokens have a major number, and may also have a minor number to further distinguish tokens with the same major number.
Tokens that can't be classified are put in the "unknown" class. For these tokens, the major and minor numbers are meaningless, although rtfTextBuf may be of interest then.
Text tokens are a single character, and the major number indicates the character value (note: can be non-ascii, i.e., greater than 127). There is no minor number.
Control symbols may have a parameter value, which will be found in rtfParam.   If no parameter was given, rtfParam = rtfNoParam.
for further details see the includefile "rtf.h".

The RTFRouteToken calls a procedure depending   on the class of the token and the last call of RTFSetClassCallback. These procedures are in the writer, so the reader can be used by any filter, which reads an rtf-file.

Changes from the distribution 1.06a2

I have made some change in the following procedures of the reader:
RTFGetToken:
Not only newlines and carriage returns, but also NULL characters, which can appear by some transports, will be silently discarded.
ReadStyleSheet:
In some German versions of WORD the Normale style is called Standard style!
And there are some RTF control-words which aren't described in the RTF-Specification I have, but nevertheless appear in RTF-files which are created by the versions of WINWORD   and Macintosh's WORD I use. So I have created a few new RTFKeys, which are shown in the following table:

rtfKMajor (major number)	rtfKMinor (minor number)	rtfKStr (symbol name)	rtfKHash (symbol name hash value)
rtfDestination,	rtfIVerscomm,	verscomm,	0
rtfTblAttr,	rtfClshdng,	clshdng,	0

for \makebackup I have taken the same RTFKey as it is taken for makeback (rtfMakeBackup), and for \cs and \ds the same as for \s (rtfStyleNum)

In rtf2LaTeX

there are the following procedures:
UnknownClass, GroupClass, ControlClass for the rtf-classes with the same name.
start_para for the text-class at the beginning of each paragraph, and TextClass elsewhere.
The procedure UnknownClass deals as the name implies with unknown rtf-control-words.
GroupClass handles with groups. (Groups are built both in rtf and LaTeX   with {})

In the procedure ControlClass a case statement calls some   procedures depending on the major-number. In these procedures it is easy to add some more lines to make a better version of this filter, which handles more rtf-control-words than mine.
(Many keywords aren't dealt with, e.g. the two rtf-control-words \clshdng, \verscomm, I have added in the reader.)

The two translation-files

Two translation-files are need for the rtf2LaTeX filter:
The one with the extension "code" tells the filter how to treat characters above 128.
The other has the extension "land" and tells the filter how to translate some special WORD styles like footnote, heading 1, heading 2, ...

The character translation file

The default file is mac.code, which is for rtf-files made by Macintosh programs. (These files have the rtf-control-word \mac at the beginning.) You have to use the file ansi.code for rtf-files made by DOS or WINDOWS (rtf-control-word \ansi)) Both of these files comprise the most important characters. If you need more, you will be able to add more in a very easy way. You can create your own file as follows:

A comment must be written in the first line of the file. Then the lines with the characters follow. The rtf-number of the character (from 128 to 255) is at the beginning of each line. Then the LaTeX command (a string) follows after a single space. A comment can be written after the 25th column. You needn't include all numbers from 128 to 256. There are no tab-stops allowed. Here is an example of a file: (for an example of a whole file see mac.code)

nr	LaTeX-command	comment	
193	\'{A}	/* e1	*/
194	\^{A}	/* e2	*/
195	\~{A}	/* e2	*/

The special WORD style translation file

Some WORD-styles will be translated in LaTeX-commands (e.g. heading 1 to section, heading 2 to subsection, ...) Others will be ignored (if the flag -d is off), because LaTeX treats them in a better way than WORD (e.g. footnote reference, footnote text) The files with the extension .land tells the filter how they should be translated.

The default file is english.land, which is for rtf-files made by the English versions of WORD and WINWORD. (These files use the styles heading 1, heading 2, ... footer, header, footnote text, ...) You have to use the file german.land for rtf-files made by German versions of WORD and WINWORD. (style names: Gliederung 1, Fusszeile, ...) Both of these files comprise the most important special WORD styles. If you need more, you will be able to add more in a very easy way. You can create your own file as follows:

A comment must be written in the first line of the file. Then the lines with the styles follow. The name of the style is at the beginning of each line (Must be a string of 20 characters). It may be followed by the LaTeX-begin-command (a string of 20 characters) and the LaTeX-end-command (20 characters). (If there are no LaTeX commands the style will be ignored.) A comment can be written after the 60st column. (This comment will be added to the LaTeX output file) There are no tab-stops allowed.
Here is an example of a file: (for an example of a whole file see english.land)

word-name	latex-begin-command	latex-end-command	comment
heading 1	\section{	}	heading 1 -> \section
heading 2	\subsection{	}	heading 2 -> \subsection

heading 4

heading 4 ignored