

Getting Started

Unpacking the Distribution Files

Macintosh

The Macintosh distribution file is a binhex'ed stuffit archive. Depending on how you obtained the file, some parts of this install process may have already been done for you (by your browser for example.)

- Σ If the distribution file ends with .hqx, it is in binhex format. Open the file with Stuffit Expander (either drag and drop the file onto Stuffit Expander, or choose Expand from the File menu.) Stuffit Expander will convert the file to a Stuffit archive. When this is done, Stuffit Expander may automatically run step 2 as well, depending on your preferences.
- Σ If the distribution file ends in .sit, you have a Stuffit archive. Open the file with Stuffit Expander (either drag and drop the file onto Stuffit Expander, or choose Expand from the File menu.) Stuffit Expander will convert the file to a folder.
- Σ If you have a folder, you are done!

Unix

The Unix distribution file is a compressed tar archive. Depending on how you obtained the file, some parts of this install process may have already been done for you (by your browser for example.)

1. If the distribution file ends with .Z, it is in compressed format. Type:
`uncompress xxxxxxxxx.Z`
where xxxxxxxxx.Z is the distribution file name. This will give you a .tar file for step 2.
2. If the distribution file ends in .tar, you have a tar archive. Type
`tar -xf xxxxxxxxxxxx.tar`
where xxxxxxxxx.Z is the distribution file name.
3. The resulting directory contains all of the files you need to begin translating. Set the environment variable RTFLIBDIR to the full path name of the directory. You also need to add this directory to your execution path (path variable in csh, PATH in bourne shell).
4. You may modify the shell script 'rtftohtml' to suite your preferences, this allows you to add options to the command line. You don't need to do this now, just remember it for later.

Windows 3.1/DOS/Windows 95/Windows NT

The Windows/DOS distribution file is a zip archive. Depending on how you obtained the file, some parts of this install process may have already been done for you (by your browser for example.)

5. If the distribution file ends with .zip, a zip archive. Use PKUNZIP or comparable utility to unzip the file.
6. Add the directory containing r2hwin3.exe (Win3.1) or r2hw95.exe(Win95/WinNT) to your PATH
(set this in AUTOEXEC.BAT)
7. set RTFLIBDIR to the directory containing rtftohtm
(set this in AUTOEXEC.BAT)!
8. You may modify the batch file R2H.BAT (Win3.1) or R2H95 (Win95/WinNT) to suite

your preferences, this allows you to add options to the command line. You don't need to do this now, just remember it for later.

Setting up Your Translation Environment


There are several changes you will need to make to the files `html-trn` and `nav-panl`, to customize them to your site and your preferences. You can make these changes now, or stay with the defaults until you have had an opportunity to try the filter out on some documents. If you stay with the defaults, you will need to convert your documents in the directory `docs`. This is because the default navigation buttons will be referenced by the URL: "images/xxx.gif" where xxx.gif is the actual file name of the graphic. For example:

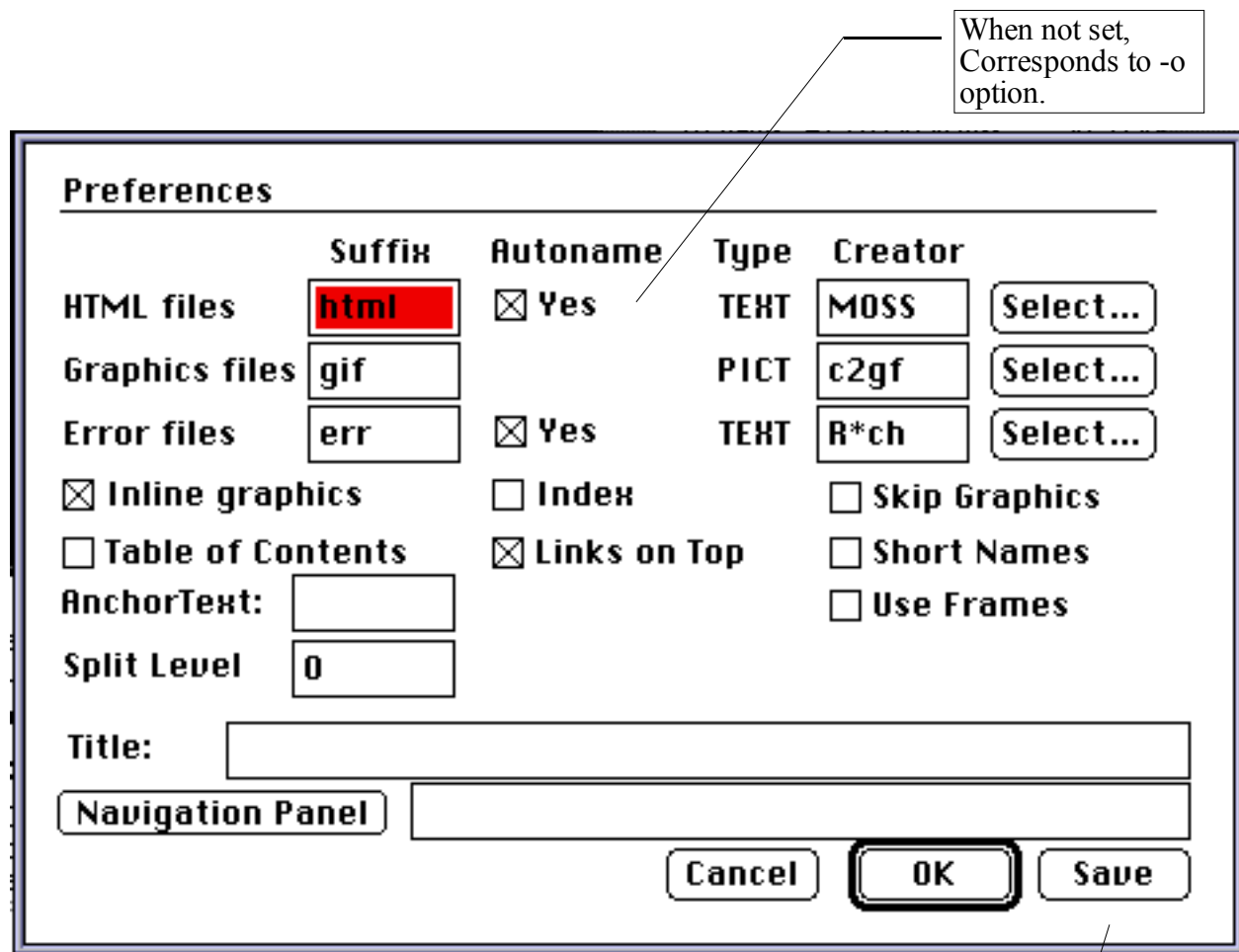
```
docs
  yourfile.rtf
  images
    leftg.gif
    rightg.gif
    ...
```

Converting a document

Macintosh

The Macintosh version supports conversions by dragging your RTF file onto the the filter. The The Macintosh version of RTFtoHTML allows you to set **Command line options** by starting RTFtoHTML and choosing "Preferences..." from the Edit menu. If you use the **OK** button, those preferences will be saved for the current invocation. If you use the **SAVE** button, they will be saved for all invocations. Once your preferences are set, you can drag all your RTF files onto the

application Icon  and they will be translated.



These preferences can be saved as the default

For complete information on the preferences, and translation files, see [Customizations](#) and [FAQ](#).

When you have converted your document you may also find image files with names like xxxx.pict or xxxx.wmf. These are graphics from your RTF document. You need to convert these images to .gif format so that browsers can display them.

You will probably see several messages beginning with “Unknown Paragraph style”. This is normal and you can ignore them. But you can improve your translation by reviewing your input RTF file, and deciding what HTML markup would be best suited to display those paragraphs. Then add a line for each style to the end of the html-trn file. See [Unknown Paragraph style](#) for details.

All warning/error messages will be displayed in the status window, and also written to an .err file.

Macintosh Tips

Σ If you are converting documents **produced** on a Macintosh, set the graphics file creator (in

preferences) to Clip2Gif. Obtain Clip2Gif from <http://hyperarchive.lcs.mit.edu/HyperArchive.html> or your favorite net location. Then run Clip2Gif and set Clip2Gif's preferences to (in the Options menu) to automatically convert files, and always rename them. Then when RTFtoHTML creates a graphic file, you can simply double click it, and Clip2Gif will open and automatically generate the GIF file. Clip2Gif is freeware (see it's distribution for details.) See [Graphics and rtfhtml](#) for more information.

Σ If you are converting documents **produced** on a Windows machine, set the graphics file creator to GraphicConverter. Obtain GraphicConverter from <http://hyperarchive.lcs.mit.edu/HyperArchive.html> or your favorite net location. GraphicConverter can translate Windows MetaFile format images (.wmf).Clip2Gif is freeware (see it's distribution for details.) See [Graphics and rtfhtml](#) for more information.

Σ If you want to keep your HTML documents in a different location than the RTF input, turn off the autoname checkbox in the Preferences. Then for each document you convert, the filter will prompt you for an output file name. You can change folders for the output file, and the filter will place the HTML and graphic files in that folder.

Unix

9. To run the filter on Unix, type
rtfhtml filename.rtf

10. For a complete description of the command line options, see: [Command line options](#).
When you have converted your document you may also find image files with names like xxxx.pict or xxxx.wmf. These are graphics from your RTF document. You need to convert these images to .gif format so that browsers can display them.

You will probably see several messages beginning with "Unknown Paragraph style". This is normal and you can ignore them. But you can improve your translation by reviewing your input RTF file, and deciding what HTML markup would be best suited to display those paragraphs. Then add a line for each style to the end of the html-trn file. See [Unknown Paragraph style](#) for details.

All warning/error messages will be displayed to standard error, and also written to an .err file.

Unix Tips

Σ If you are converting documents **produced** on a Macintosh, obtain NetPBM - a freeware conversion suite. NetPBM is able to convert PICT files to gif format. Alternatively, you may consider Apple's MAE (Macintosh Application Environment) which would allow you to run Clip2Gif on your UNIX box. See [Graphics and rtfhtml](#) for more information.

Σ If you are converting documents **produced** on a Windows machine, you may want to look at SoftWindows for your UNIX box. This would allow you to run a Windows graphic file converter on your UNIX box. I don't know of any native UNIX graphic converters that can handle Windows MetaFile format images. See [Graphics and rtfhtml](#) for more information.

Σ If you want to keep your HTML documents in a different location than the RTF input, specify the -o option to save th HTML and image files in a different location. You could also modify the rtfhtml shell to perform some mapping of input to output file names.

Win3.1/DOS

In Windows environments, you can either

11. Open a DOS command line window

12. cd to the directory containing your RTF files
13. run rtfhtml as follows:
r2h filename.rtf

Or you can use the "File Manager" to give you a drag-n-drop interface to the filter:

14. Open a "File Manager" window displaying the batch file R2H.BAT.
15. Open a second window displaying your RTF input file.
16. Drag the RTF file onto the R2H.BAT file.

You will need to force "File Manager" to refresh the window containing your RTF file before it will show you that a .HTML output file has been created.

You will probably see several messages beginning with "Unknown Paragraph style". This is normal and you can ignore them. But you can improve your translation by reviewing your input RTF file, and deciding what HTML markup would be best suited to display those paragraphs. Then add a line for each style to the end of the html-trn file. See [Unknown Paragraph style](#) for details.

All warning/error messages will be displayed in the DOS window, and also written to an .err file.

Windows/DOS Tips

Σ If you are converting documents **produced** on a Macintosh, obtain NetPBM - a freeware conversion suite. NetPBM is able to convert PICT files to gif format. Alternatively, you may consider Apple's MAE (Macintosh Application Environment) which would allow you to run Clip2Gif on your UNIX box. See [Graphics and rtfhtml](#) for more information.

Σ If you are converting documents **produced** on a Windows machine, you can use Paint Shop Pro , or HiJaak Pro to convert the WMF image files to .GIF format. See [Graphics and rtfhtml](#) for more information.

Σ If you want to keep your HTML documents in a different location than the RTF input, specify the -o option to save the HTML and image files in a different location. You could also modify the R2H.BAT batch file to perform some mapping of input to output file names.

Windows 95/Windows NT

In Windows environments, you can either

17. Open a DOS command line window
18. cd to the directory containing your RTF files
19. run rtfhtml as follows:
r2h95 filename.rtf

Or you can drag-n-drop your RTF files onto the R2H95 batch file.

You will probably see several messages beginning with "Unknown Paragraph style". This is normal and you can ignore them. But you can improve your translation by reviewing your input RTF file, and deciding what HTML markup would be best suited to display those paragraphs. Then add a line for each style to the end of the html-trn file. See [Unknown Paragraph style](#) for details.

All warning/error messages will be displayed in the DOS window, and also written to an .err file.

Windows 95/ Windows NT Tips

- Σ If you are converting documents **produced** on a Macintosh, obtain NetPBM - a freeware conversion suite. NetPBM is able to convert PICT files to gif format. Alternatively, you may consider Apple's MAE (Macintosh Application Environment) which would allow you to run Clip2Gif on your UNIX box. See [Graphics and rtfhtml](#) for more information.
- Σ If you are converting documents **produced** on a Windows machine, you can use Paint Shop Pro , or HiJaak Pro to convert the WMF image files to .GIF format. See [Graphics and rtfhtml](#) for more information.
- Σ If you want to keep your HTML documents in a different location than the RTF input, specify the -o option to save th HTML and image files in a different location. You could also modify the R2H95.BAT batch file to perform some mapping of input to output file names.

License

All versions of RTFtoHTML are copyrighted by Chris Hector. RTFtoHTML versions 3.0 and higher are subject to this license agreement. If you do not agree to be bound by the terms of this license agreement, you must promptly remove this product from your computer system.

RTFtoHTML may be run for 30 days from the date of acquisition of the software for the purpose of evaluating the software. After 30 days, you must either purchase a license for the software, or remove it from your computer system.

A **single user license** for RTFtoHTML entitles the owner to:

- Σ use the Software on a single computer
- Σ use the Software on a network providing that each user of the software
- Σ has a license.
- Σ use the Software on a second computer as long as only one copy is being
- Σ used at a time
- Σ make copies of the software for archival purposes

A **server license** for RTFtoHTML entitles the owner to:

- Σ use the Software on a single computer
- Σ use the Software to provide document translations for users on the
- Σ network who do not have a single user license for the Software.
- Σ make copies of the software for archival purposes

A **site license** for RTFtoHTML entitles the owner to:

- Σ any number of user and/or server licenses for the servers and
- Σ individual PC's on a single site
- Σ (can include multiple buildings within a five mile radius) and

Σ that are owned or leased, and that are operated by the licensee

You may not:

- Σ permit other individuals to use the Software except under the terms
- Σ above rent, lease, grant a security interest in or otherwise transfer
- Σ rights to the Software
- Σ modify, translate, reverse engineer, decompile, disassemble or create
- Σ derivative works based on the Software

Note

The 'demo mode' is not a different version of the filter. The version you have is the full and complete filter. When you purchase the product you will receive a license key. Once the filter finds the license key on your system, it will run without displaying the 'demo mode' message, and will not have the 20-second delay.

Disclaimer of Warranty

This software is provided on an "AS IS" basis without warranty of any kind, including without limitation the warranties of merchantability, fitness for a purpose and non-infringement. The entire risk as to the quality and performance of the Software is borne by you. Should the Software prove defective, you and not Chris Hector assume the entire cost of any service and repair. **SOME JURISDICTIONS DO NOT ALLOW EXCLUSIONS OF AN IMPLIED WARRANTY, SO THIS DISCLAIMER MAY NOT APPLY TO YOU AND YOU MAY HAVE OTHER LEGAL RIGHTS THAT VARY BY JURISDICTION.**

CD Distributions

RTFtoHTML may not be re-distributed on CD's or other media without express written authorization.

Internet sites

Internet sites may re-distribute binaries of RTFtoHTML under the following conditions:

- Σ The software is only stored on the computer for the purpose of re-distribution unless licensed for use as described above
- Σ The complete distribution (for a platform) is provided (documentation, binaries and license agreement)
- Σ I encourage the site to send email to chris@sunpack.com so that I can provide information on updates and bugfixes.

Copyright

RTFtoHTML is copyright © 1996 by Chris Hector.

Pricing

The pricing for RTFtoHTML 3.x is as follows:

- Σ User-\$29.00
- Σ Server - \$290.00
- Σ Site - \$950.00

See the **License** section for descriptions of User, Server and Site Licenses. There is no special Government, Educational or Non-Profit pricing - everyone gets these same low prices!

Order Form

I can accept payment by Visa, American Express, MasterCard, Discover, US check or by wire transfer.

I cannot accept checks drawn on banks outside of the US, as it costs me more to cash the check than I charge for the software!

Please use the accompanying Order form to order this product. It uses secure transactions, so your credit card number is safe. If you are sending a check or using wire transfers you can register with this form, and reference your check# or Wire Transfer confirmation number - this will allow me to get your license key out to you as fast as possible.

If for some reason you cannot order online, Please print and fill out the the accompanying Order form. Send the order form by regular mail to

Mailing Address

Chris Hector 9937 Goodhue St. NE. Blaine, MN. 55449

or Fax it to

FAX and Phone Number

FAX:612-785-2210 Phone:612-785-2505 ext 616
--

Purchase Orders and Invoices

If your order must be processed by your purchasing department please print and fill out the accompanying Order form and use it as an INVOICE for your internal processing. This will cut WEEKS off of the time it takes to get your order processed. Also make sure that ALL

CORRESPONDENCE CONTAINS YOUR EMAIL ADDRESS!!!! Your license key will arrive by email.

Wire Transfers

Please use the accompanying Order form to order this product. **THIS IS IMPORTANT!** **Include the date, amount and bank of your wire transfer as reference information on the order form. When I receive a wire transfer I cannot trace it back to a sender, only to a bank. I need your order form to complete the transaction and get you a license key.**

Register with the form below, and send your wire payment to:

First Bank NA
601 2nd Ave So.
Minneapolis, MN 55402
USA

via SWIFT line:

number is : FNBMUS44IMT

New York

ABA number is: 091000022
Account is:
Christopher J or Nancy L Hector
account number: 116397617568

Users Guide

What is rtftohtml?

rtftohtml is a tool to turn your, say, Word documents into documents which may be read from within the World Wide Web. The format of these documents is called **HyperText Markup Language** (HTML). rtftohtml is able to automatically convert documents stored in RTF (**Rich Text Format**) to HTML. Most word processors in use on UNIX, Macintosh, PC or NeXT systems can export their documents in RTF format (hint: have a look at the "Save as..." dialog box of your favorite word processor).

In processing text, rtftohtml chooses HTML markup based on three characteristics. These are

20. The destination of the text. Example destinations are header, footer, footnote, picture.
21. The paragraph style. Paragraph styles are user-definable entities, but some are pre-defined by the word processing package. For Microsoft Word (on the Macintosh) examples are "Normal" and "heading 1" or ("<berschrift 1" when using a german version).
22. The text attributes. Examples of text styles are bold, courier, 12 point.

The filter has built-in rules for dealing with destinations. For paragraph and text styles, the rules for translation are contained in a file called **html-trn**. By modifying this file, you can train rtfhtml to perform the correct translations for your documents. The most common change that you will need to make is to add your own paragraph styles to html-trn.

rtfhtml should produce reasonable HTML output for most documents. Here is what you can expect:

- Σ Your output should appear in a file called "xx.html" where "xx" or "xx.rtf" was your input file name.
- Σ Bold, italic and underlined text should appear with ,<i> and <u> markup
- Σ Courier font text should appear with <tt> markup
- Σ Tables will be formatted with HTML table markup.
- Σ Footnotes will appear at the end of the current document with hypertext links to them.
- Σ Table of contents, indexes, headers and footers are discarded.
- Σ Table of Contents entries and paragraphs with the style "heading 1..6" will generate a hypertext Table of Contents in a separate file. Each table of contents entry will link to the correct location in the main document.
- Σ All paragraph styles used in your document should appear in the file "html-trn". This allows you to create a mapping from any paragraph style to any HTML markup. There are many pre-defined styles in html-trn, including "heading 1..6". (If a paragraph style is not found, a warning will be generated and the text will be written to the HTML file with no special markup.)
- Σ Each graphic in your file will be written out to a separate file. The filename will be "xxn.ext" where "xx" or "xx.rtf" was your input, "n" is a unique number and "ext" will be either "pict" for Macintosh PICT format graphics or "wmf" for Windows Meta-Files format graphics. The HTML file will create links to these files, using either "<A HREF=" or "<IMG SRC=" links. **SINCE most WWW browsers do not understand "wmf" or "pict" format files, the link will be to xxn.gif.** This presumes that you will run some **other** filter to translate your graphic files to gif.
- Σ Text that is connected with copy/paste-link constructs, or tagged with some special text attributes will generate hypertext links.

RTFtoHTML converts a linear RTF-Document (that may contain cross references, index entries and footnotes) into a fully hypertexted set of HTML-Documents. This document, that you are reading right now, is an example of what you can expect from using rtfhtml.

The 3.0 release added the following features to rtfhtml:

- Σ splitting of the document:
RTFtoHTML uses the headlines to determine where to split (the splitting depth is adjustable by the user, see also section **Headings**)
- Σ fully customizable navigation panels on top and bottom of each page (see sections **Headings** and **Navigation panel**)
- Σ active **Cross references** to headings, mail addresses and URLs
- Σ automatic generation of an active table of contents

automatic generation of an active [Index](#) support for a few of Netscape's HTML extension, such as the `<center>` tag and background images (see also section [Navigation panel](#)) and of course: the [Users Guide](#)

Supported platforms

rtftohtml is available for UNIX, Macintosh (Power PC and 68000), DOS, Windows 3.x, Windows 95 and Windows NT systems. The Windows systems are currently supported by the DOS application which can be run in all of those environments. This now includes the rttoweb extensions on all platforms.

Command line options

Under UNIX and DOS versions rtttohtml is invoked by a command like this:

```
rtftohtml [options] file [[options] file2...]
```

`file` is the name of the RTF-file to convert. By default the HTML-output will be written to a file with the same basename, but with the `.rtf`-extension replaced with `.html`.

The most common options are (see below for more detailed descriptions of these):

<code>-hlevel</code>	to split the document at headings of level <i>level</i> ,
<code>-c</code>	to create a table of contents, and
<code>-x</code>	to create an index (of course, this requires index entries to be in the RTF document)

These should suffice in most cases. When invoking rtttohtml without any arguments, a list of all available options is printed. Following is a complete and detailed list of all available command line options, sorted alphabetically:

<code>-c</code>	Generate a table of contents on a separate page (the title page itself normally only contains the level-1-headings).
<code>-F</code>	Turns on Frames Generation for the Table of Contents.
<code>-G</code>	Indicates that no graphics files should be written. The hypertext links to the graphics files will still be generated. This is a performance feature for when you are re-translating a document and the graphics have not changed.
<code>-hlevel</code>	This tells rtttohtml to split the generated HTML document at all headings of level <i>level</i> , thus creating a separate HTML file for the contents of every section at that level. If <i>level</i> is 0, only one HTML file will be created with a table of contents at the top of the HTML file. If <i>level</i> is omitted, rtttohtml will create a separate HTML file for every section.

For example, `-h1` only splits the HTML-File when a level-1-heading is encountered. All lower level headings will be internally referenced from the top of the respective HTML-File.

- `-i` This option is also required for the `-c` and `-x` options. Indicates that imbedded graphics should be linked into the main document using an IMG tag. The default is to use an HREF style link.
- `-N file` If the `-h` option has been given, this option allows you to tell `rtftohtml` that it should read the description for the navigation panels it produces from file *file*. Navigation panels are described in more detail in section [Navigation panels](#).
- `-o filename` Indicates that the basic output file name should be *filename*. If any other files are created (such as for graphics), the basename of the other files will be *filename* without ".rtf" if it is present in the name.
- `-P extension` Use *extension* as the extension for any links to graphics files. The default for this is "gif".
- `-s` Use short filenames when splitting the HTML output. In this case the HTML-files are simply numbered. If this option is not present the file-names contain the first eight characters of the first heading contained in the respective files.
- `-t` Place external references to headlines near the top of the page. (Default is that they occur at the and of the page.)
- `-T title` Use *title* as the document title. This overrides any title supplied within the RTF-file (see also section [Supplying a title](#)).
- `-x` Generate an index. This works by gathering all the index entries of the RTF-File and inserting an invisible anchor at that place.
- `-X text` By default, `rtftohtml` inserts index anchors without any text, thus producing "empty anchors" which most Web browsers have no problems with. Unfortunately there is one prominent exception to this statement: NCSA Mosaic. Mosaic requires anchors to contain at least one non-blank character, otherwise they are not recognized. This option lets you use *text* as the text of such anchors when using Mosaic as your Web browser, e.g. "`-X ·`".

Using `rtftohtml`

To convert a document from RTF (Rich Text Format) to HTML, `rtftohtml` requires the contents of the RTF-file to be formatted with a certain set of paragraph styles. For example, headings at level 1 must be formatted with the paragraph style “heading 1” (which is the built-in default for headings anyway; german heading styles may be called “<berschrift xy”, but they appear in the RTF file as “heading xy”, too), lists must be formatted with a paragraph style such as “numbered list” etc. The reason for this is that `rtftohtml` needs to know which paragraph styles it should map to which HTML tags. This mapping between styles and tags can be customized by editing the file `html-trn` in `rtftohtml`’s library directory (see section [html-trn](#) for more), to create a mapping from your own individual paragraph styles to HTML-tags. Although this is not as complicated as it might seem, I personally prefer to adjust my Word-documents to use only (or at least mostly) the paragraph styles recognized by `rtftohtml` by default. In this chapter I will stick to this strategy. See section “[Unknown Paragraph style](#)” for a few words on how to customize `rtftohtml` to correctly interpret your own paragraph style.

Supplying a title

To determine the HTML-Title for the created HTML-Files (the text between the `<title>` and `</title>` tags), `rtftohtml` looks for the `\title`-token inside the `\info`-group of the RTF-File. Thus you should give your RTF-Documents a short, descriptive title in the respective dialog box of your word processor (should be called something like “File information”).

Another way to specify the document title is via the `-T` command line option. For example:

```
rtftohtml -T "My work of art" art.rtf
```

Note that this title will also be automatically inserted by `rtftohtml` into the first created HTML-File as a level-1-heading. That's why you should usually delete the very first heading from your RTF-Document (or at least assign a different paragraph format to that line) and use it as the document title. The reason for this is to prevent *RTFtoHTML* from interpreting the headline of your RTF-Document as a level 1 heading, where it should split.

Character styles

`rtftohtml` automatically recognizes and converts bold, italic and underlined text. If a certain range of text is written using a monospaced font such as Courier, it also automatically creates monospaced HTML-output for that range. What fonts are considered to be monospaced can be configured in the file `html-trn` in section [.TMatch](#) (“monospace fonts -> tt”). By default the fonts “Courier”, “Courier New” are expected to be monospaced.

If you get warning messages such as “no output translation for ...” when running `rtftohtml` you can either replace that character with a less exotic one in your RTF-file or add a translation to the end of `rtftohtml`’s library file `html-map`, such as “*character translation*”.

The newline character (created by Shift-Return) will be automatically converted to the corresponding HTML-tag,

as will the unbreakable space (created by Control-Shift-Space).

Headings

Headings must be formatted with a paragraph style like “heading 1”, “heading 2” etc. (resp. “berschrift 1” etc.) to be automatically recognized by rtfhtml. rtfhtml uses these styles to determine when it should split the HTML-file. The heading level at which splitting should take place can be configured by the command line switch `-h level` (see section [Command line options](#)). If a heading contains no text (i.e. it is empty) it will be ignored by rtfhtml.

If the `-h` switch was present when rtfhtml was invoked, a navigation panel will be inserted at the top and at the bottom of every generated HTML file. This navigation panel will contain the following elements:

- Σ Previous, if there is a previous section at the same heading level,
- Σ Next, if there is a next section at the same heading level,
- Σ Up, if the current section is not at level 1,
- Σ Title, to go to the title document,
- Σ Contents, if a table of contents has been generated, i.e. the `-c` switch had been given to rtfhtml
- Σ Index, if an index has been generated, i.e. the `-x` switch had been given to rtfhtml

rtfhtml will try to use the language of the RTF-file for labelling the navigation panel. Currently there is support for english, spanish, french and german. However, if you would like a more fancy-looking panel, with buttons etc., you can tell rtfhtml (by writing a simple configuration file) what HTML-code it should use for the individual panel elements. The creation of such configuration files is described in detail in section [Navigation panels](#).

Lists

rtfhtml knows about the following lists (in braces is the name of the respective paragraph style it expects such lists to be formatted with):

- | | |
|----------------------------|---|
| numbered (“numbered list”) | items start with a tab and end with a paragraph mark
(numbers before the tab are ignored) |
| unnumbered (“bullet list”) | items start with a tab and end with a paragraph mark
(bullets etc. before the tab are ignored) |
| Glossaries (“glossary”) | term and definition are separated by a tab, glossary entries
are separated by a paragraph mark |

Nested lists can be created from an RTF document by using a different style for each level of indentation. The styles "bullet list 1" "numbered list 2" ... represent different levels of nesting, with “bullet list 1” being at nesting level 1. The only rule for use is that no levels of nesting are skipped. For example, a "numbered list 3" paragraph must not appear immediately after a "Normal" paragraph. It must follow a paragraph with a nesting level of 2 or higher.

An example sequence of paragraph styles to produce a nested list might look like this:

```
numbered list
  bullet list 1
    bullet list 2
    glossary 2
  bullet list 1
    numbered list 2
```

Tables

rtftohtml is able automatically convert tables to HTML 3.0 tables.

Images

Graphics are imbedded in RTF in either a binary format or an (ASCII) hex dump of that binary. I have never seen a binary format graphic - I don't think that the filter will process binary correctly. It does handle the hex format of graphics, by converting the hex back into binary and writing the binary to a file. The file extension is chosen by looking at the original type of the graphic. The following list shows the file types and their extensions:

Macintosh PICT	.pict - also, 256 bytes of nulls are prepended to the graphic. This is to conform to the PICT file format.
Windows Meta-files	.wmf
Windows Bit-map	.bmp

In addition, the filter produces a link to the file containing the graphic. Now, since the above graphic formats are not very portable, the filter assumes that you will convert these files to something more useful, like GIF. So the format of the link is:

```
<a href="basenameN.ext">Click here for a Picture</a>
```

where

- Σ • `basename` is the name of the input document (without the `.rtf` extension)
- Σ • `N` is a unique number (starting at 1)
- Σ • `ext` is an extension. This defaults to GIF, but can be overridden with the `-P` command line option.

Since most Web browser only support images in GIF-format, you will have to convert the generated PICT- and WMF-files to GIF. For PICT there is `picctoppm/ppmtogif`, but for WMF? I don't know of any WMF translators for Unix; for DOS there is `wmf2bmp`, whose output could then be converted to GIF via the `pbmplus-tools`. From what I understand, WMF is not a pixel- but a vector-graphic format, so maybe it would be easier to translate WMF to Postscript and then let Ghostscript do the job of converting to GIF. Any volunteers for writing a `wmftops` utility?

You can also change the link to an IMG form. If you specify the `-I` command line option, all links to graphics will be of the form:

```
<IMG src="basenameN.ext">
```

There is one other special case. If a graphic is encountered when the filter is in the process of generating a link, the IMG form of the link is used even without the -I command line option.

Cross references

All kinds of cross references can be created from within the RTF-file. The reference itself must be formatted with the attributes “double-underline/hidden” and must follow the standard HTML-conventions, such as “http://www.w3.org” or “file.txt” or “#mark1”. The “hot” text, that is the text that will appear “clickable” in your Web-browser, immediately follows the reference and must be double-underlined, but not hidden.

Anchors for internal cross references (such as “mark1”, corresponding to the example above) must be formatted either with the attributes “hidden/outline” or “hidden/superscript”. For example [this link](#) will bring you to the list of other features.

If you just want to create a reference to a certain heading resp. section, it is sufficient to simply format the reference with the color red. The text of the reference must match the beginning *n* characters of the heading, so the references “[Supplying](#)” and “[Supplying a title](#)” point to the same section.

If an email address such as chris@sunpack.com is colored red, rtfhtml will automatically produce a cross reference of type “mailto”.

The same work for all other kinds of URLs, so if the URL <ftp://ftp.rzn.uni-hannover.de/pub/> is colored red, rtfhtml will automatically produce a reference pointing to that URL.

Index entries and footnotes

If your RTF document contains footnotes or endnotes, the filter will place the text of the footnote in a separate HTML document. At the footnote reference mark, the filter will generate a hypertext link to the text of the footnote. This works with either automatically numbered footnotes¹, or user supplied footnote reference marks²⁺

If you insert index entries into your RTF-document and give rtfhtml the -x-option, rtfhtml will generate a hypertext’ish index for the generated HTML-documents. Note that when using NCSA-Mosaic as your Web browser you should also tell rtfhtml to insert some text into the generated anchors by using the command line switch -X *text* (see section [Command line options](#)).

Other features

Horizontal lines

The paragraph style “hr” can be used to produce a horizontal line in the HTML output (this will be translated to the <hr> tag).

¹ Look, there is one now!

²⁺ There is my mark.

Discarding Unwanted Text

If you have text that you do not want to appear in the HTML output, simply format the text as Hidden and Plain (that is, no underline, outline...)

If you wish to modify the formatting that discards text, you need to change the entry in `html-trn` that specifies "`_Discard`".

Imbedding HTML in a Document

Normally, if your RTF document contained the text "`<cite>hello</cite>`", the translator would output this as: "`<cite>hello</cite>`". This ensures that the text would appear in your HTML output exactly as it appeared in the original RTF document. If, however, you want the `<cite></cite>` to be interpreted as HTML markup, you must format the tags using Hidden and Shadow or Hidden and Strikethrough. The filter will then send the tags through without translation. It is also possible to use the paragraph style "HTML" to let `rtftohtml` interpret a whole paragraph as being literal HTML.

When the `rtftohtml` filter produces HTML markup, it keeps track of the nesting level of tags to ensure that you don't get something like `<cite>hello</cite>` which would be incorrect markup. If you imbed HTML markup in your document, the filter will NOT be aware of it. You must ensure that your markup appears correctly nested.

If you wish to modify the formatting for imbedded HTML, you need to change the entry in `html-trn` that specifies "`_Literal`".

Other paragraph styles

`rtftohtml` understands a few other paragraph styles by default. These are (among others):

address	Will be converted to HTML's <code><address></code> -environment.
blockquote	Will be converted to HTML's <code><blockquote></code> -environment.
pre	Will be converted to HTML's <code><pre></code> -environment. This is useful when spacing is important in a paragraph.

Customizations

Unknown Paragraph style: messages, or Adding paragraph styles

When converting existing documents to `rtftohtml` you often get a lot of warning message telling you that some paragraph styles are unknown. Now you can either

- Σ despair (don't do this, there's no need to),
- Σ ignore the messages: this may work when only the text of the paragraph is of interest, since `rtftohtml` assumes the "Standard" paragraph style when encountering one it does not know,
- Σ adjust the RTF document to use only paragraph styles which `rtftohtml` understands by default

(these are listed in the file `html-trn` in section `.PMatch`),
Σ add your own paragraph style to `rtftohtml`'s library file `html-trn`.

To add a new paragraph style, simply go to the `.PMatch` table contained in the file `html-trn` and add an entry to the end. Put the name of the paragraph style (quoted), the nesting level (usually zero) and the name of the `.PTag` entry that should be used.

html-trn File Format

The file `html-trn` is needed by `rtftohtml` to map character and paragraph styles contained in the RTF-file to corresponding HTML-tags. It must be readable either from `rtftohtml`'s library directory (as set in the file `makefile.rtftoweb`) or from the directory contained in the environment variable `RTFLIBDIR`.

In `html-trn` there are four tables. They are labelled `.PTag`, `.TTag`, `.TMatch` and `.PMatch`. These tables begin with the name (in column one) and continue until the next table starts. All blank lines and lines beginning with a '#' are discarded. '#' lines are typically used for comments. The tables themselves are composed of records containing a fixed number of fields which are separated by commas. The fields are either strings (which should be quoted) integers or bitmasks.

.PTag Table

Each entry in the `.PTag` table describes an HTML paragraph markup. The format is:

`.PTag`

`#"name", "starttag", "endtag", "col2mark", "tabmark", "parmark", allowtext, cannest, DeleteColl, fold, Toc Styl`

name	A unique name for this entry. These names are referenced in the <code>.PMatch</code> table.
starttag	This string will be output once at the beginning of any text for this markup.
endtag	This string will be output once at the end of any text for this markup.
col2mark	This string will be output in place of the first tab in every paragraph (used for lists)
parmark	This string will be output in place of each paragraph mark. (usually <code>
</code> or <code><p></code>). You should think of this a paragraph separator. This will be output BETWEEN any two paragraphs - that have this style.
allowtext	If 0, no text markup will be allowed within this markup. (for example <code><pre></code> or <code><h1></code> don't format well if they contain additional markup.
cannest	If 1, other paragraph markup will be allowed to nest within this markup. (used for nesting lists)
DeleteColl	If 1, all text up to the first tab in a paragraph will be deleted. (used to strip out bullets that when going to

fold	unordered lists (). If 1, the filter will add newlines to the HTML to keep the number of characters in a line to less than 80. For <pre> or <listing> elements, this should be set to 0.
TocStyl	The TOC level. If greater than 0, the filter will create a Table of contents entry for every paragraph using this markup.

Sample .Ptag Entries

```
"h1", "<h1>\n", "</h1>\n", "\t", "\t", "<br>\n", 0, 0, 0, 1, 1
```

This is a level 1 heading. The "\n" in the start and end-tag fields forces a newline in the HTML markup. Since newlines are ignored in HTML (except in <pre>) it's only effect is to make the HTML output more readable. There is no difference between the first tab and any other. They both translate to a tab mark. Paragraph marks generate "
" followed by a newline (just for looks). Text markup (like) is not allowed within <h1> text, because we leave that up to the HTML client. No nesting is allowed - (see the discussion on nested styles). No text is deleted. Every paragraph using this markup will also generate a level-1 table of contents entry.

```
"Normal", "<p>", "</p>\n", "\t", "\t", "<br>\n", 1, 0, 0, 1, 0
```

This is the default for normal text. Regular text in HTML has no required start and end-tags. The "\n" in the end-tag field forces a newline in the HTML markup. Since newlines are ignored in HTML (except in <pre>) it's only effect is to make the HTML output more readable. There is no difference between the first tab and any other. They both translate to a tab mark. This markup will generate <p>par1
par2</p> for a two paragraph sequence. If you want more spacing between paragraphs, add a second
.

```
"ul", "<ul>\n<li>", "</ul>", "\t", "\t", "\n<li>", 1, 1, 0, 1, 0
```

This is the entry for unordered lists. This generates a "\n" at the start of the list and "\n" at the end. There is no difference between the first tab and any other. They both translate to a tab mark. Paragraph marks generate "" preceded by a newline (just for looks). Text markup (like) is allowed, and this entry may be nested - and it allows others to be nested within it. This allows nested lists. No text is deleted.

```
"ul-d", "<ul>\n<li>", "</ul>", "\t", "\t", "\n<li>", 1, 1, 1, 1, 0
```

This entry is identical to the previous except that the DeleteColl field is set to 1. This is used to remove bullets (which really appear in the RTF) because we don't want to see them in the HTML.

.Ttag Table

Each entry in the .Ttag table describes an HTML text markup. The format is:

.Ttag

```
"name", "starttag", "endtag"
```

name

A unique name for this entry. These names are referenced in the .Pmatch table.

starttag

This string will be output once at the beginning of any text for this markup.

endtag This string will be output once at the end of any text for this markup.

Note that unlike the .Ptag table, no text markup should appear more than once. (Of course there is no good reason that it should appear.) If you have two entries with start and end tags, it would be possible to get HTML of the form text. I don't know if this is invalid markup, but it sure is ugly.

.TMatch Table

Each entry in the .TMatch table describes processing for text styles. The format is:

.TMatch

"Font",FontSize,Match,Mask,"TextStyleName"

Font	The name of a Font, or "" if all fonts match this entry.
FontSize	The point-size of the font, or 0 if all point sizes match this entry.,
Match	A bit-mask, where each bit represents a text attribute. These bits are compared to the attributes of the style being output. They must match for this entry to be matched. One in a bit position means that the text style is set, a zero is not set.
Mask	A bit-mask, where each bit represents a text attribute. In comparing the style of the text being processed, to the Match bit-mask, this field is used to select the bits that matter. If a zero appears in a bit-position, then that style attribute is ignored (for the purpose of matching this entry.) Only 1 bits are used in the above comparison.
TextStyleName	This is either the name of an entry in the .Ttag table indicating the HTML markup to use, or it is one of "_Discard", "_Name", "_HRef", "_Hot", or "_Literal".

The order of bits in the Match and Mask bit-maps are:

```
# v^bDWUHACSOTIB - Bold
# v^bDWUHACSOTI - Italic
# v^bDWUHACSOT - StrikeThrough
# v^bDWUHACSO - Outline
# v^bDWUHACS - Shadow
# v^bDWUHAC - SmallCaps
# v^bDWUHA - AllCaps
# v^bDWUH - Hidden
# v^bDWU - Underline
# v^bDW - Word Underline
# v^bD - Dotted Underline
# v^b - Double Underline
# v^ - SuperScript
# v - SubScript
```

Sample .TMatch Entries

```
# double-underline/not hidden -> hot text
# double-underline/hidden -> href
# v^bDWUHACSOTIB,v^bDWUHACSOTIB
```

```
"" , 0, 0010000000000000, 001000100000000, "_Hot"  
"" , 0, 0010001000000000, 001000100000000, "_HRef"
```

The first entry will match any text formatted with double underline EXCEPT if it is hidden text. This is accomplished by using those two bits to compare (the MASK field) and having a 1 in the double underline bit and a zero for the hidden text bit. The second entry will match any text formatted with BOTH double underline and hidden text. Any text that matches the first will be treated as the hot text of a link. Any text that matches the second will be taken as the href itself. (The filter requires that the HRef text immediately precede the Hot text.)

```
# Regular matches - You can have multiple of these active  
# monospace fonts -> tt  
"Courier", 0, 0000000000000000, 0000000000000000, "tt"
```

This will match any text that uses the Courier font and mark it using the HTML text markup appearing in the .TTag table with the entry name "tt".

```
# bold -> bold  
#      v^bDWUIACSOTIB, v^bDWUIACSOTIB  
"" , 0, 0000000000000001, 0000000000000001, "b"
```

This will match any text that has bold attributes and will mark it using the HTML text markup appearing in the .TTag table with the entry name "b". Note that bold text using the Courier font would match both this entry and the previous. This will yeild markup of the form <tt>hi</tt>. Note that "b" is the name of an entry in the .TTag table, not the HTML markup that is used!

.PMatch Table

Each entry in the .PMatch correlates a paragraph style name to some entry in the .PTag table. The format is:

.PMatch

"Paragraph Style",nesting_level,"PTagName"

Paragraph Style

The paragraph style name that appears in the RTF input.

nesting_level

The nesting level. This should be zero except for nested list entries.

PTagName

The name of the .PTag entry that should be used for paragraphs with this paragraph style.

Sample .PMatch Entries

```
"heading 1", 0, "h1"
```

This is a level 1 heading. Any paragraphs with this paragraph style will be mapped to the entry in the .PTag table named "h1".

```
"numbered list", 0, "ol-d"
```

This is used for numbered lists. Any paragraphs with this paragraph style will be mapped to the entry in the .PTag table named "ol-d".

```
"numbered list 2", 2, "ol-d"
```

This is an entry for a nested paragraph style. The nesting level of two is used to indicate that this paragraph should appear in the HTML nested within two levels of paragraph markups. The

paragraph marked with this style may only appear after a paragraph style that has a nesting level of 1 or greater.

.Strings table

The .Strings table sets values to be used by the translator
format is "name","value"

The quotes may be omitted for numbers, but the two lines:

“TheValue”,”-1”

“TheValue,-1

are identical. Any name can be defined by the Strings table, but only those outlined below will have any affect on translations.

String	Meaning
SkipNavPanel	If set to 1, the navigation panel usually produced at the top and bottom of pages will be omitted. This is primarily useful when you are not splitting documents, or producing an index.
SkipLeadingToc	At the start of each HTML file, you will normally find a list of all of the headings contained within that file. By setting SkipLeadingToc to 1, this list will be omitted.
SkipTrailingToc	Same as above only for trailing lists.
SkipDimsOnIMGs	For IMG tags, the filter will generate HEIGHT/WIDTH attributes. Setting this string to 1 will omit these tags.
WMFAdjust	In producing HEIGHT/WIDTH attributes for WMF graphics, a scaling factor of 133% is normal applied (by experimentation this scaling appears to be necessary.) If you find that your graphics are not appearing the same size as in you documents, you may modify this scaling factor. (133 is the default, meaning 133%)
DefaultTitle	If no title is found in the document, and no title is supplied on the command line, this string is used for a title value.
AllBorder	If this is set to 1, all tables will have the BORDER attribute. Normally, only those that have a border on the first cell will have borders.
ExtTarget	Defines a tag that will be included in the tag, for external hrefs. If set to 'TARGET="_top"' clicking on that hyperlink in a framed document will replace the entire window, not just the

	current frame.
FNSep	This tag is inserted before Footnotes at the end of the page
PreferrEmbed	If set, choose embedded graphic over linked
ShortFileDigits	This is the number of digits to use in Short file names (-s option.) If set to 2 (the default) it will allow splitting into 99 files, 3 would support 999 files.
DoAllPars	By default, empty paragraphs are deleted. Setting to 1 will generate all paragraphs (note that empty paragraphs in a list will generate bullet or sequence numbers without any following text.)

.LXForm table

The LXForm table is used when an RTF document contains a link to an image instead of the image itself. Since the link contains a filename which is local to the machine that the RTF file was produced on, some translation of that filename is required. The LXForm allows you to specify a series of transformations to be done to the filename. Each line in the LXForm table is applied in order to the filename.

The format of the table is:
"tag"[,"regexp","regsub"]
Where tag can be

"LOWER"	translate to lower case
"UPPER"	translate to upper case
"ENCODE"	encode "bad" characters as hex
"DECODE"	decode "bad" characters from hex#
"SUBSTITUTE"	"perform a regular expression style substitution where "regexp" is a regular expression search pattern and "regsub" is a substitution pattern

For example, the following table:

```
.LXform
# translate all special characters to %nn format
"ENCODE"
# strip directory from file name
"SUBSTITUTE",".*%5C", ""
# Strip extension if it exists
"SUBSTITUTE","\.[^.]*$", ""
# Add a .gif extension
"SUBSTITUTE", "$", ".gif"
```

would make the following transformations given the filename "C:\ PICTS\CLOWN.WMF"

<u>Transformation</u>	<u>Output</u>
"ENCODE"	C%3A%5CPICTS%5CCLOWN.WMF
"SUBSTITUTE", ". *%5C", ""	CLOWN.WMF
"SUBSTITUTE", "\.[^.]*\$", ""	CLOWN
"SUBSTITUTE", "\$", ".gif"	CLOWN.gif

Navigation panels and Netscape support

If you want the navigation panels produced by `rtftohtml` (see section [Headings](#)) to look more spiffy, e.g. with images as panel buttons, or if you want the generated HTML documents to use images as their background or another text color, this section is for you.

By using the `-N` [Command line option](#) when invoking `rtftohtml`, it is possible to tell `rtftohtml` exactly how you want the created navigation panels to look like. The same configuration file can be used to add a few funny Netscapisms to the generated documents. If no `-N`-option was given, but `rtftohtml` finds a file named `nav-pan1` in its library directory or the directory contained in the environment variable `RTFLIBDIR` it will use this file as the layout customization file. This way you can avoid having to add the `-N` command line options whenever you use `rtftohtml`.

An example for such a customization file is the file `nav-pan1`, which has also been used when this guide was converted to HTML. By looking at this file you should easily see how the layout of your documents can be adjusted to your taste.

Each line of such a customization file contains the definition of a layout element, as long as the first character is not the hash-character (`#`), which introduces comments. Everything that follows the first colon (`:`) in each line will be literally inserted into the HTML-files when needed.

The following elements may be configured:

previous	What to insert into the navigation panel when the “previous” element is to be created.
next	The same for the “next” element.
up	The same for the “up” element.
title	The same for the “title” element.
contents	The same for the “contents” element.
index	The same for the “index” element.
delimiter	What to use as the delimiter between the elements of navigation panels.
hr	What HTML-code to use when it’s time to insert a horizontal line beneath or above navigation panels.
bgimage	Specifies an optional background (GIF-) image that should be used as the document background (requires Netsape).
bgcolor	Specifies an optional background color that should be used in the document background (requires Netsape). Syntax: <code>#rrggbb</code> (hexadecimal values for red, green, blue).

textcolor

The color to use for normal text. Same syntax as for bgcolor.

FAQ - Frequently Asked Questions

Graphics and rtfhtml

rtfhtml will generate a separate file for each picture found in your RTF source. The type of the picture file depends on how it was added to the RTF. If your word processor runs on Microsoft Windows, you will probably get a WMF (Windows Metafile Format) graphic. If your word processor runs on a Macintosh, you will get a PICT file. Since neither WMF nor PICT graphics are supported formats on the WWW, you will want to convert these files to GIF format. The following programs will convert to GIF format.

Name	Platform	Input formats
netpbm	UNIX	PICT
HiJaak Pro	Windows	WMF
Graphic Workshop	Windows	WMF
wmf2bmp	Windows	WMF
Graphic Converter	Mac	PICT and WMF
GIF Converter	Mac	PICT
Paint Shop Pro	Windows	WMF
Clip2Gif	Macintosh	PICT, GIF, TIFF and JPEG

HiJaak Pro

HiJaak Pro is made by
Inset Systems, 71 Commerce Drive, Brookfield CT 06804-3405, Phone
203-740-2400, Fax 203-775-5634 They also have a toll free number
800-374-6738 (800 DR INSET). They have a BBS, 203-740-0063 and are on
CompuServe (GO INSET).

the home pages for HiJaak Pro is at

<http://www.imsisoft.com/hijaak/hijaak.html>

The program runs under Microsoft Windows and says it can

- Σ View, edit, and image-process a file
- Σ Convert a file from one format to another (over 70 formats)
- Σ Capture a screen
- Σ Print an image
- Σ Organize your graphics files

Graphic Workshop

The Graphic Workshop is a shareware program (\$40) for MS Windows. It can be found at <ftp://uunorth.north.net/pub/alchemy/gwswin11.zip>. It looks really nice. It can convert a lot of graphic formats including .wmf.

GIFConverter 2.3.7

Kevin A. Mitchell <74017.2573@compuserve.com>

Kevin A. Mitchell

P.O. Box 803066

Chicago, IL 60680-3066 USA

License: Shareware (\$40 + shipping)

GIFConverter, by Kevin A. Mitchell, reads and writes the following graphics file formats: GIF, MacPaint, PICT, RIFF, RLE, Thunderscan, Startup Screen, TIFF and JPEG (with or without QuickTime). In addition, it writes EPSF files. Also provided are image enhancement, cropping, color table selection, and dithering features.

GraphicConverter 2.0.2

Thorsten Lemke <thorsten_lemke@pe2.escape.de>

Thorsten Lemke

Insterburger Str. 6

31228 Peine

Germany

License: Shareware (\$35)

GraphicConverter, by Thorsten Lemke, imports PICT, Startup-Screen, MacPaint, TIFF (uncompressed, packbits, CCITT3/4 and lzw), RIFF, PICS, 8BIM, 8BPS/PSD, JPEG/JFIF, GIF, PCX/SCR, GEM-IMG/-XIMG, BMP (RLE compressed BMP«s also), ICO/ICN, PIC (16 bit), FLI/FLC, TGA, MSP, PIC (PC Paint), SCX (ColorIX), SHP, WPG, PBM/PGM/PPM, CGM (only binary), SUN (uncompressed), RLE, XBM, PM, IFF/LBM, PAC, Degas, TINY, NeoChrome, PIC (ATARI), SPU/SPC, GEM-Metafile, Animated NeoChrome, Imagic, ImageLab/Print Technic, HP-GL/2, FITS, SGI, DL, XWD, WMF, Scitex-CT, DCX and KONTRON.

GraphicConverter exports PICT, Startup-Screen, MacPaint, TIFF (uncompressed, packbits and lzw), GIF, PCX, GEM-IMG/-XIMG, BMP, IFF/LBM, TGA, PSD, JPEG/JFIF, HP-GL/2, EPSF, Movie (QuickTime), SUN, PICS,

PICT in Resource and PBM/PGM/PPM.

JPEGView 3.3

Aaron Giles <giles@med.cornell.edu>
Aaron Giles
182 E. 95th Street 11E
New York, NY 10128 USA

License: Freeware (Send a Postcard)
Master Site: <ftp://ftp.med.cornell.edu/pub/jpegview>

JPEGView, by Aaron Giles, is a flexible image utility designed to allow quick, high-quality viewing of the most common image formats, including JPEG, JFIF, GIF, PICT, Baseline and LZW-compressed TIFF, Windows BMP, StartupScreen, and MacPaint. JPEGView can also convert between QuickTime JPEG and JFIF-standard JPEG files. Version 3.0 added a number of substantial new features, including full AppleScript support, new high-quality dithering routines, a greatly improved slide show, floating windows, etc.

JPEGView is now distributed only as a "fat binary", i.e. a program containing both 68K and Power PC code, thereby enabling it to run flat-out on both normal Macintoshes and the new Power Macintoshes.

Paint Shop Pro

It is shareware.
I found it on CIS
the home pages for Paint Shop Pro is at

<http://www.jasc.com/psp.html>

Clip2GIF

Clip2GIF is a great tool that integrates well with RTFtoHTML. In the preferences for rtftohtml, change the creator for image files to that of Clip2Gif. Then set Clip2Gif preferences to do conversion automatically (instead of displaying.) Now when you convert, you will see Clip2Gif files appear in your directory. Just double clicking on them will create the gif files that you need.

The following information was taken from the Clip2Gif homepage at <http://iawww.epfl.ch/Staff/Yves.Piguet/clip2gifhome/default.html><http://iawww.epfl.ch/staff/Yves.Piguet/clip2gif-home/clip2gif.html>

clip2gif is a freeware utility for the Macintosh to convert PICT, GIF, TIFF and JPEG images to any

of these formats, written by Yves Pigué. The current version, 0.7.2 (317 K) (cf. release notes), can be obtained at the following places (and all other good info-mac archives):

clip2gif's main features are

- Σ Display of PICT, GIF, TIFF and JPEG files
- Σ GIF output
 - Σ transparency
 - Σ interlacing
 - Σ depth of 1, 2, 4 or 8 bits/pixel
 - Σ gray shades output
- Σ TIFF output
 - Σ bilevel, grayscale, palette-color or RGB
 - Σ Packbits compression when useful
- Σ JPEG output (requires QuickTime)
- Σ Pict output
- Σ scaling
- Σ input from PICT, GIF, TIFF or JPEG files or the clipboard
- Σ simple drawings with AppleScript
- Σ comprehensive on-line help

Character Sets and Translation

1. Introduction

Text characters in an RTF file may be specified as literal characters or using `\xx` notation, where `xx` is the hex value of the character. RTF files also contain a control word that specifies the character set that's used within the document and governs the interpretation of character values. The charset control words are:

```
\ansi  ANSI (default) - Used by Word for Windows
\mac   Apple Macintosh
\pc    IBM PC
\pca   IBM PC page 850, used by IBM Personal System/2
```

Although the four charsets don't appear to differ for characters in the ASCII range (below 128), they differ considerably above the ASCII range (128–255). For example, the ANSI, Macintosh, and PC charsets represent the degree sign (“°”) as `\b0`, `\a1`, and `\f8` respectively. Furthermore, even for a given charset, character values in the Symbol font represent different characters than they do generally. For example “a” in Symbol font is the greek letter alpha.

rtftohtml uses a translation model that divides translation into two parts. When the RTF file is read each character is mapped to an standard character name. This mapping is controlled by the files:

ansi-gen	Input mapping for ANSI, all fonts except symbol
ansi-sym	Input mapping for ANSI, symbol font
mac-gen	Input mapping for Macintosh, all fonts except symbol

mac-sym	Input mapping for Macintosh, symbol font
pc-gen	Input mapping for IBM PC, all fonts except symbol
pc-sym	Input mapping for IBM PC, symbol font
pca-gen	Input mapping for IBM PS2, all fonts except symbol
pca-sym	Input mapping for IBM PS2,, symbol font

General and symbol charset maps are stored in the text files *ansi-gen*, *ansi-sym*, *mac-gen*, *mac-sym*, *pc-gen*, *pc-sym*, *pca-gen*, and *pca-sym*. Each line of a charset file associates an RTF character value (field 2) with the standard character name to which the RTF character corresponds (field 1). Here's a sample from *ansi-gen*:

```
parenleft      (
parenright     )
space         " "
quotedbl      '"'
quoteright    "'"
quoteleft     "`"
a             a
b             b
c             c
bullet        0x95
emdash        0x96
endash        0x97
```

Character values may be given as a single character (in which case the ASCII value is used), or as a hex number 0xyy. Single or double quotes may be used to quote values containing whitespace or quotes (e.g., use single quotes to quote a double-quote).

Lines with a “#” in column one are taken as comments. Comments and blank lines are ignored.

When the HTML file is written, each standard character name is mapped to an ascii string. This mapping is controlled by the file *html-map*. The format of this file is

- (i) field 1 is the standard character name, just as in the charset maps;
- (ii) field 2 is the output sequence to produce for the character named in field 1

Lines with a “#” in column one are taken as comments. Comments and blank lines are ignored.

Here's part of the *html-map* file:

```
a             a
b             b
c             c
ampersand     &amp;
less          &lt;
equal         =
greater       &gt;
trademark     (TM)
AE            &AElig;
Aacute       &Aacute;
Acircumflex  &Acirc;
```

Error Conditions and Remedies

First Things First - Determine the character set.

1. Identify which character set you are using by looking at the top of the RTF file. You should see one of the character sets identifiers, either `\ansi`, `\mac`, `\pc`, or `\pca`. If you do not see any of these then `\ansi` is your character set.

Error: Invalid Character Code `\xx` - outputting `�`

If you get this error, that means that a character appeared in the RTF input that has no standard character name. You fix this by adding that code (in this case the Hex Code `xx`) to your input character mapping file. Which file you use depends on your character set:

```
\ansi  ansi-gen and ansi-sym
\pc    pc-gen and pc-sym
\mac   mac-gen and mac-sym
\pca   pca-gen and pca-sym
```

- 1) Determine what character set you are using as described above.
- 2) Find the character code in your source and determine what it looks like when it is displayed by your word processor. Also check to see if it is a character in the symbol font.
- 3) Add `\e4` to the input translation file:

If you are using `ansi` and the character code is in the symbol font then you should edit `ansi-sym` and add:

```
somename          0xe4
```

The name that you use depends on what `e4` looks like on your screen. If it is a bullet, then use

```
bullet            0xe4
```

This tells the filter to treat this character code as a bullet.

The complete list of standard character names is here:

Error: No output translation for `:bullet`

If you get this error, this means that the filter has no output mapping for the standard character name "bullet". To fix this, edit the `html-map` file and add a line describing what to output for a bullet. For example:

```
bullet *
```

What to do when you want to change a character translation

Start with `html-map`. Find the standard character name that you want to have translated, and then edit the corresponding string.

About RTF

RTF (Rich Text Format) was designed by Microsoft as an open format for interchanging documents between Microsoft Word and other word processing packages. It is supported by WordPerfect, FrameMaker, Interleaf and many other packages on UNIX, Apple, Macintosh, Next, and PC platforms.

The specification for RTF and a set of filters for translating RTF to plain-text, troff and Tex and

LaTeX are available at: <ftp://ftp.primate.wisc.edu/pub/RTF>

The latest version of the RTF specification is Version 1.5 which corresponds to Microsoft Office 97. The specification comes in two parts, the [1.14 RTF Specification](#) and [1.15 RTF Additions](#) .