The LaTeX2HTML translator: An Overview

Herbert W. Swan

dprhws@Arco.com

LaTeX2HTML is, as its name implies, a translator which converts a standard LaTeX document into Hypertext Markup Language (*HTML*), for incorporation into the World-Wide Web. Like LaTeX, it is freely available software, supported by highly dedicated volunteers. Unlike LaTeX, it is currently available only on UNIX platforms. Complete documentation concerning this software may be browsed from `<http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual>`. A European mirror is reported at `<http://www.rzg.mpg.de/rzg/software/latex2html/>`. From this online document, a PostScript version of the complete document may easily be obtained for hardcopy viewing. This article (the one you are reading) represents a condensation of that more complete documentation.

LaTeX2HTML replicates the basic structure of a LaTeX document as a set of interconnected *HTML* files which can be explored using automatically generated navigation panels. The cross-references, citations, footnotes, the table-of-contents and the lists of figures and tables, are also translated into hypertext links. Formatting information which has equivalent “tags” in *HTML* (lists, quotes, paragraph-breaks, type-styles, etc.) is also converted appropriately. The remaining heavily formatted items such as mathematical equations, pictures or tables are converted to images which are placed automatically at the correct positions in the final *HTML* document.

LaTeX2HTML also extends LaTeX by supporting arbitrary hypertext links and symbolic cross-references between evolving remote documents. It allows the specification of *conditional text* and the inclusion of raw *HTML* commands. These hyper-media extensions to LaTeX are available as new commands and environments from within a LaTeX document.

System Requirements

Before you consider obtaining LaTeX2HTML, you should first ensure that your UNIX system includes the following utilities:

- *Perl* version 4.036 or later. Since LaTeX2HTML is written in the *Perl* language, this utility is essential. Since *Perl* version 5 makes more efficient use of dynamic memory, it is recommended over version 4.
- Your UNIX system must support some form of keyed database management system, such as *DBM, NDBM, SDBM,* or *GDBM.* To find out whether your system supports one of these standards, do a “**man dbm**,” etc. If you are using *SDBM* or *GDBM*, use must use *Perl* version 5.
- LaTeX2HTML can translate both LaTeX2$_\varepsilon$ documents, as well as the older LaTeX 2.09 documents. However, if you wish to use all of the features of the LaTeX2HTML translator, you should upgrade your LaTeX program itself to the newer version 2e.
- If you want to be able to use LaTeX to translate equations, figures and arbitrary environments into GIF images for web browsing, you will also need the following UNIX utilities available and working:
  - *dvips* version 5.516 or later or *dvipsk.* It is also recommended that your installation of *dvips* permits users to generate their own fonts through *Metafont* and the *MakeTeXPK* script. This will cause equations and other GIF text to be much more screen-readable.
  - *Ghostscript* version 2.6.1 or later, with the *ppmraw* device driver built in. (To see whether your version of *Ghostscript* contains this driver, type
    gs
    devicenames ==
    quit
  - The *pbmplus* or *netpbm* library. Some of the image filters in these libraries are used during the PostScript to GIF image conversion (described below).
  - If you dislike the white background color of the generated inlined images then you should get either the *netpbm* library (instead of the older *pbmplus*) OR install the *giftrans* filter by Andreas Ley `<ley@rz.uni-karlsruhe.de>`. Version 1.10.2 is known to work without problems, but later versions should also be OK.

Obtaining LaTeX2HTML

One way LaTeX2HTML may be obtained is through one of the three primary Comprehensive TeX Archive Network (CTAN) sites nearest you. They are located in the United States `<ftp.shsu.edu>`, the United Kingdom `<ftp.tex.ac.uk>`, and Germany `<ftp.dante.de>`. It can be found under the `tex-archive/support/latex2html` directory.

The CTAN version will always be the latest stable revision of LaTeX2HTML, currently V96.1-b. However, it will not always contain the latest patches submitted by the user community. If you wish to be on the leading edge, you are better off obtaining the program from `<ftp://www-dsed.llnl.gov/files/programs/unix/latex2html/sources>` (European mirror `<ftp://ftp.rzg.mpg.de/pub/soft/latex2html/sources>`. This directory will contain all of the recent LaTeX2HTML releases, including the latest major release, 96.1, dated January of 1996. After downloading the file `latex2html-96.1.tar.gz`, you will have to unzip it with "**gzip -d**," and extract the files with "**tar xvf latex2html-96.1.tar**." You should then proceed to download the latest revision (currently Rev. e, dated April 8, 1996), *latex2html-96.1.reve.tar.gz.* Simply unzip and extract this file on top of the original release file for the latest patches. It will include *.diff* files which will show you what patches were made since the previous major release.

Installation

Once the LaTeX2HTML source files are obtained, installation consists of:

1. Editing the *Perl* scripts, to tell the system where the *perl* translator resides;
2. Editing the `latex2html.config` script to tell LaTeX2HTML where the necessary UNIX utility programs reside;
3. Running the *install-test* script, which mainly verifies the previous two steps;
4. Making local copies of the LaTeX2HTML icons so that they are accessible to your WWW server. The installation variable `$ICONSERVER` in `latex2html.-config` must then be set to this location.

   Warnings: If you cannot do that bear in mind that these icons will have to travel from Livermore, California!!! Also note that several more icons were added that were not present in previous versions of LaTeX2HTML.

5. Customizing the installation, as necessary. This is accomplished by setting various "installation variables" (described later) in the file `latex2html.config` Individual users may also override the system-wide configuration, either globally for all their documents, for separately for individual documents.

   For a "per user" initialization file, copy the file `dot.latex2html-init` in the home directory of any user that wants it, modify it according to her preferences and rename it as `.latex2html-init`. At runtime, both the `latex2html.config` file and `$HOME/.latex2html-init` file will be loaded, but the latter will take precedence.

   You can also set up a "per directory" initialization file by copying a version of `.latex2html-init` in each directory you would like it to be effective. An initialization file `/X/Y/Z/.latex2html-init` will take precedence over all other initialization files if `/X/Y/Z` is the "current directory" when LaTeX2HTML is invoked.

Based on recent postings to the LaTeX2HTML mailing list (described below), the top three installation difficulties that new users encounter are:

Image conversion Although LaTeX2HTML is able to process ordinary text adequately, it fails to "find" the images to process. One common cause of this problem is that the *ppmraw* device is not built into *Ghostscript.* Another cause involves the TEXINPUTS environment variable. During normal operation, LaTeX2HTML creates a subdirectory under the current one to contain the translated *HTML* files. It is from this subdirectory that LaTeX and *dvips* are run during image conversion. If the TEXINPUTS environment variable that is visible to these programs does not include "..", then image conversion will fail. At some sites, "latex" and "dvips" are really scripts which set TEXINPUTS and some other environment variables and call te actual programs. If this is the case, the the user including ".." in his or her TEXINPUTS will not solve the problem.

Database problems Sometimes LaTeX2HTML will simply die if the *Perl* interface to the UNIX database routines is not working. In *Linux* systems, this problem is typically solved by uncommenting the "**use GDBM_File**" line in the LaTeX2HTML and *install-test* scripts. On other systems it may be necessary to recompile *Perl* with the proper database libraries.

File globbing failure This problem manifests itself by LaTeX2HTML creating only some temporary files, but no valid *HTML* files. This problem can be traced to the inability of *Perl* to locate the *csh* program. Look for a line similar to "`csh='csh'`" in the *Perl* `Config.pm` file. If it is absent, reconfigure *Perl* .

Normal Operation

To use LaTeX2HTML simply type: "**latex2html <file>.tex**." The "**.tex**" suffix is optional and will be supplied by the program if omitted by the user. This will create a new directory called <file> which will contain the generated *HTML* files, some log files and possibly some images. To view the result use an *HTML* viewer such as NCSA Mosaic or Netscape Navigator on the main *HTML* file which is `file/file.html`. This file will contain navigation links to the other parts of the generated document.

The LaTeX2HTML script includes a short manual which can be viewed by typing "**nroff -man latex2html**."

If a GIF image (of an equation or a figure, etc.) is used more than once in the document, it will be generated only once. Furthermore, on subsequent calls to LaTeX2HTML on the same document, most images need not be regenerated at all, but are recycled from one run to the next. Their names might change, but their contents do not. The only exception are images which are *order-sensitive,* meaning that their content depends upon their location. Examples of order-sensitive images are `equation` and `eqnarray` environments. This is because their figure numbers are part of the image. Figures and tables with captions, on the other hand, are order-insensitive because the figure numbers are handled by *HTML* and are not part of the image itself.

For the most part, LaTeX2HTML acts as expected, processing *m*ost of the commands standard to LaTeX. `\ref` and `\label` pairs are translated as as anchor/hyper-references in the *HTML* document. LaTeXlists are converted to *HTML* lists. Footnotes are collected onto one page and displayed as hypertext. Tables and certain inline equations are translated to their *HTML* equivalents, providing the selected *HTML* version supports these constructs.

Command-line options

It is possible to customize the output from LaTeX2HTML using a number of command-line options with which you can specify how to break up the document, where to put the generated files, what the title is, what the signature at the end of each page is, what to put in the navigation panel, what kind of extra information to include about the document, whether to retain the original LaTeX section-numbering scheme, the version (below) of *HTML* to generate, etc.

The command-line options described below can be used to change the default behaviour of LaTeX2HTML. Alternatively the corresponding environment variables in the initialization file `.latex2html-init` may be changed, in order to achieve the same result.

-split <**num**> Stop splitting sections into separate files at this depth. A value of `0` will put the document into a single *HTML* file. The default is `8`.

-link <**num**> Stop revealing child nodes at each node at this depth. (A node is a `\part` or `\chapter` or `\section` or `\subsection` or `\subsubsection` etc.). A value of `0` will show no links to child nodes, a value of `1` will show only the immediate descendents, etc. A value at least as big as that of the `-split` option will produce a table of contents for the tree structure, rooted at each given node. The default is `4`.

-nolatex Disable the mechanism for passing unknown environments to LaTeX for processing. This can be thought of as "draft mode" which allows faster translation of the basic document structure and text, without fancy figures, equations or tables.
This option has been superseded by the `-no_images` option (see below).

-external_images Instead of including any generated images inside the document, leave them outside the document and provide hypertext links to them.

-ascii_mode Use only ascii characters and do not include any images in the final output. In ascii mode the output of the translator can be used on character-based browsers which do not support inlined images (the *HTML* IMG tag).

-t <**top-page-title**> Name the document using this title.

-up_url <**URL**> Specifies a universal resource locator (URL) to associate with the *"UP"* button in the navigation panel(s).

-up_title <**string**> Specifies a title associated with this URL.

-prev_url <**URL**> Specifies a universal resource locator (URL) to associate with the *"PREVIOUS"* button in the navigation panel(s).

-prev_title <**string**> Specifies a title associated with this URL.

-down_url <**URL**> Specifies a URL for the *"NEXT"* button in the navigation panel(s).

-down_title <**string**> Specifies a title associated with this URL.

-contents <**URL**> Specifies a URL for the *"TABLE OF CONTENTS"* button, for document segments that would not otherwise have one.

-index <**URL**> Specifies a URL for the *"INDEX"* button, for document segments that otherwise would not have an index.

-external_file <**filename-prefix**> Specifies the prefix of the `.aux` file that this document segment should read. The `.aux` extension will be appended to this prefix. This file is necessary for document segments to obtain figure and section numbers from LaTeX.

-dir <**output-directory**> Redirect the output to this directory.

-no_subdir Place the generated *HTML* files in the current directory. The default behaviour is to create (or reuse) another file directory.

-prefix <**filename-prefix**> The <filename-prefix> will be prepended to all `.gif`, `.pl` and `.html` files produced, except for the top-level `.html` file. This will enable multiple products of LaTeX2HTML to peacefully coexist in the same directory. However, do **not** attempt to *simultaneously* run multiple instances of LaTeX2HTML using the same output directory, else various temporary files will overwrite each other.

-font_size <**size**> This option provides better control over the font size of environments processed by LaTeX. <size> must be one of the font sizes that LaTeX recognizes; i.e. `10pt`, `11pt`, `12pt`, etc. The default size is the same as the LaTeX document. Whatever size is selected, it will be magnified by the installation variables `$MATH_SCALE_FACTOR` or `$FIGURE_SCALE_FACTOR`.

-no_tex_defs If specified, the translator will make no attempt to interpret raw TeX commands. This feature will enable sophisticated authors the ability to insert arbitrary TeX commands in environments that are destined to be processed by LaTeX anyway; e.g. figures, theorems, etc.

-ps_images Use links to external PostScript images rather than inlined GIF images.

-address <**author-address**> Sign each page with this address.

-no_navigation Disable the mechanism for putting navigation links in each page.

-top_navigation Put navigation links at the top of each page.

-bottom_navigation Put navigation links at the bottom of each page as well as the top.

-auto_navigation Put navigation links at the top of each page. Also put one at the bottom of the page if the page exceeds `$WORDS_IN_PAGE` number of words (default = 450).

-index_in_navigation Put a link to the *index-page* in the navigation panel if there is an index.

-contents_in_navigation Put a link to the *table-of-contents* in the navigation panel if there is one.

-next_page_in_navigation Put a link to the *next logical page* in the navigation panel.

-previous_page_in_navigation Put a link to the *previous logical page* in the navigation panel.

-info <**string**> Generate a new section *About this document...* containing information about the document being translated. The default is to generate such a section with information on the original document, the date, the user and the translator. An empty string (or the value `0`) disables the creation of this extra section. If a non-empty string is given, it will be placed in the contents of the *About this document...* section instead of the default information.

-reuse <**reuse_option**> The <reuse_option> specifies how or whether image files are to be shared or recycled, and accepts three valid options:

- **`0`** Do not ever share or recycle image files. This choice also invokes an interactive session prompting the user about what to do about a pre-existing *HTML* directory, if it exists.
- **`1`** Recycle image files from a previous run if they are available, but do not share identical images that must be created in this run.
- **`2`** Recycle image files from a previous run and share identical images from this run. (This is the default).

-no_reuse Do **not** share or recycle images generated during previous translations. This is equivalent to `-reuse 0`. (This will enable the initial interactive session during which the user is asked whether to reuse the old directory, delete its contents or quit.)

-init_file <**file**> Load the specified file. This *Perl* file will be loaded after loading `$HOME/.latex2html-init`, if one exists. It can be used to change the default options.

-no_images Do not attempt to produce any inlined images. The missing images can be generated "off-line" by restarting LaTeX2HTML with the option `-images_only`.

-images_only Try and convert any inlined images that were left over from previous runs of LaTeX2HTML.

-show_section_numbers Show section numbers. By default the section numbers are not shown in order to encourage the use of particular sections as stand-alone documents. In order to be shown, section titles must be unique and must not contain inlined graphics.

-html_version <**version**> This specifies the *HTML* version (See below). to generate. Currently, versions 2.0, 2.1, 2.2, 3.0 and 3.1 are available. The default version is 2.0.

-vs Print the current version of LaTeX2HTML.

-debug Run in debug mode, using the *Perl* debugger, if it is installed.

-h Print out the list of options.

HTML version

The Hypertext Mark-up Language is an evolving standard, with different versions supporting different features. In order to make your documents viewable by the widest possible audience, you should use the most basic *HTML* version in common usage. This is currently version 2.0 which is the default version for LaTeX2HTML. It supports images, server-side image maps, interactive forms, and minimal typographic elements (bold, italic and teletype). This version adheres to the ISO–Latin–1 (ISO–8879) character set. Other *HTML* versions supported by LaTeX2HTML are:

Version 2.1 *HTML* version 2.1 is essentially identical to *HTML* version 2.0, with some extensions for internationalization. Most importantly, the default character set is no longer ISO–8859–1 but ISO–10646, commonly known as *Unicode*. This is a 16-bit character set and can thus display a much larger set of characters.

Version 2.2 This version supports all the features of version 2.1, plus the *HTML* 3 Table Model. This feature is already available in many *HTML* browsers, including Netscape Navigator V1.2 and later.

Version 3.0 This version adds to version 2.2 some of the *HTML* 3.0 textual formatting features, including centering, flush right, flush left and underlining.

Version 3.1 LaTeX2HTML revision "d" and later support *HTML* subscripts and superscripts, which may be browsed by Navigator V2 and later.

Version 3.2 In addition to all of the features of version 3.1, this version adds support for math markup. Currently the only browsers which can display this mark-up are *Arena* (`http://www.w3.org/hypertext/WWW/Arena`) and *UdiWWW* for *Windows 95* (`http://www.uni-ulm.de/~richter/udiwww`.)

Very few native TeX commands are supported. The ones that are include: `\newdimen`, `\newbox`, `\vskip`, `\hskip`, `\char`, and some limited forms of `\def`. Furthermore, `\def` and `\renewcommand` may not behave as expected, if they define the same entity more than once in the same document. Furthermore, macro definitions cannot contain any verbatim-like environments.

Internationalization

A special variable `$LANGUAGE_TITLES` in the initialization or configuration files determines the language in which some section titles will appear. For example setting it to
`$LANGUAGE_TITLES = ''french'';`
will cause LaTeX2HTML to produce “Table des matières” instead of “Table of Contents”. Furthermore, the value of `\3rd June 1996` is presented in a format which is customary in that language.

The only languages currently supported are `french`, `english` and `german`, but it is trivial to add support for another language in the file `latex2html.config`. As a guide here is the entry for the French titles:

```
sub french_titles {
 $toc_title = "Table des mati\\'eres";
 $lof_title = "Liste des figures";
 $lot_title = "Liste des tableaux";
 $idx_title = "Index";
 $bib_title = "R\\'ef\\'erences";
 $info_title =
       "\\'A propos de ce document...";
 $abs_title = "R&eacute;sum&eacute;";
 $pre_title = "Pr&eacute;face";
 $app_title = "Annexe";
 @Month = ('','janvier','f&eacute;vrier',
         'mars', 'avril', 'mai',
         'juin', 'juillet', 'ao&ucirc;t',
         'septembre', 'octobre',
         'novembre', 'd&eacute;cembre');
}
```

In order to provide full support for another language you may also want to replace the navigation buttons which come with LaTeX2HTML (which are by default in English) with your own. As long as the new buttons have the same filenames as the old ones, there should not be a problem.

Hypertext Extensions to LaTeX

One of the greatest appeals of the World-Wide Web is its high connectivity through hyperlinks. LaTeX2HTML implements several LaTeX extensions designed to rapidly exploit this connectivity. These extensions are made known to LaTeX by using the *html* package (style file) included with the LaTeX2HTML distribution.

The simplest (but most tedious) way to insert a hypertext link into your document is by use of the `\htmladdnormallink` command. This command takes two arguments: the text to highlighted as hypertext, and the URL that that hypertext will take you if selected. Images may be inserted in your document via `\htmladdimg`, which takes as its single argument the URL of the image to be inserted. These commands are time-consuming for the author, because exact URLs must be supplied, and because they must be manually updated if the destination URLs are ever changed.

Part of the author’s burden of maintaining hyperlinks is relieved by the use of symbolic links. The LaTeX `\label` and `\ref` mechanism is one way to providing such links. If the target link is another LaTeX2HTML-processed document, this mechanism may be extended by:

1. Substituting the `\externalref` command for the `\ref` command, and
2. Including an `\externallabels` command somewhere in the document to may a correspondence between the URL of the target document and its LaTeX2-HTML table of symbols. Such a symbol table is automatically generated by LaTeX2HTML for every document.

Still another way of cross referencing text, either within or between documents, is with the `\htmlref` command. This macro is similar to `\htmladdnormallink`, except that the second argument is replaced by a symbolic LaTeX label, defined by `\label`. If the label not found within the current document, it may instead appear in any of the external LaTeX documents listed in an `\externallabels` command. There is also a `\hyperref` construction which produces a different wording of the link in the printed version of the document than in the electronic one.

If a portion of the text is to appear only in the *HTML* version of the document, it may be enclosed within the `htmlonly` environment. Short text segments destined only for *HTML* may alternately be placed in the `\html` command. Conversely, portions of the document intended only for the printed version may be enclosed within the `latexonly` environment, or the `\latex` command.

Any other *HTML* construct may be inserted by way of the the `rawhtml` environment. Any text contained within this environment is passed directly to the *HTML* file without modification. This text could include forms, frames, or any other *HTML* object.

Navigation panels

The navigation panels are the strip containing “buttons” and text that appears at the top and perhaps at the bottom of each generated page and provides hypertext links to other sections of a document. Some of the options and variables that control whether and where it should appear have already been mentioned.

A simple mechanism for appending customized buttons to the navigation panel is provided by the command `\htmladdtonavigation`. This takes one argument which LaTeX2HTML appends to the navigation panel. For example,

```
\htmladdtonavigation{\htmladdnormallink
   {\htmladdimg{http://server/URL}}{%
   http://server/link}}
```

will add an active button `mybutton.gif` pointing to the specified location.

Apart from these facilities it is also possible to specify completely what appears in the navigation panels and in what order. As each section is processed, LaTeX2HTML assigns relevant information to a number of global variables. These variables are used by the subroutines `top_navigation_panel` and `bottom_navigation_panel`, where the navigation panel is constructed as a string consisting of these variables and some formatting information.

These subroutines can be redefined in a system or user configuration file (respectively, `LATEX2HTMLDIR/latex2html.config` and `\$HOME/.latex2html-init`). Any combination of text, *HTML* tags, and the variables mentioned below is acceptable.

The control-panel variables are:

Iconic links (buttons):

PREVIOUS Points to the previous section;
UP Points up to the "parent" section;
NEXT Points to the next section;
NEXT_GROUP Points to the next "group" section;
PREVIOUS_GROUP Points to the previous "group" section;
CONTENTS Points to the contents page if there is one;
INDEX Points to the index page if there is one.

Textual links (section titles):

PREVIOUS_TITLE Points to the previous section;
UP_TITLE Points up to the "parent" section;
NEXT_TITLE Points to the next section;
NEXT_GROUP_TITLE Points to the next "group" section;
PREVIOUS_GROUP_TITLE Points to the previous "group" section.

If the corresponding section exists, each iconic button will contain an active link to that section. If the corresponding section does not exist, the button will be inactive. If the section corresponding to a textual link does not exist then the link will be empty.

The number of words that appears in each textual link is controlled by the variable `$WORDS_IN_NAVIGATION_PANEL_TITLES` which may also be changed in the configuration files.

Figure 1 shows an example of a navigation panel specification. (The "." is the *Perl* string concatenation operator and "#" signifies a comment).

Converted images

LATEX2HTML copies the contents certain commands and and all environments that it was not specifically programmed to handle into the file `images.tex` for GIF file conversion. This conversion is done by way of LATEX, *dvips, Ghostscript,* and the portable bitmap library. The contents of these commands and environments are not altered by LATEX2HTML, except for macro expansion, but this can be defeated if necessary. The complete LATEX preamble from the user's document is also passed to `images.tex,` so that user macros, packages and graphics options (like `\graphicspath`) are made available to LATEX processing the images. The *environments* which are processed as images include:

- Inline equations, excepts those which can be processed by LATEX2HTML according to its *HTML* version, specified as above;
- `equation` and `eqnarray`, except when running under *HTML* version 3.2.
- `table` and `tabular`, except under *HTML* version 2.2 or higher;
- `figure`, `floatfig`, `wrapfig`, and other floating-figure environments;
- `theorem` and `minipage` environments;
- Any user-defined environment (as described below) specified to LATEX2HTML that should be passed to LATEX.

The standard LATEX *commands* which are converted to images include `\fbox`, `\framebox`, `\parbox`, `\dag`, `\ddag`, `\oe`, `\OE`, and certain special accents not defined in *HTML*. Also converted to images are the `graphics` package commands, `\rotatebox`, `\scalebox`, `\reflectbox`, `\resizebox`, `\includegraphics`, `\epsfig`, `\psfig`, `\epsffile`, and `\epsfbox`.

```
sub top_navigation_panel {
    #  Start with a horizontal rule (3-d dividing line)
       "<HR> ".
    # Now add few buttons with a space between them
      "$NEXT $UP $PREVIOUS $CONTENTS $INDEX $CUSTOM_BUTTONS" .
    # Line break
      "<BR>\n" .
    # If ''next'' section exists, add its title to the navigation panel
      ($NEXT_TITLE ? "<B> Next:</B> $NEXT_TITLE\n" : undef) .
    # Similarly with the ''up'' title ...
       ($UP_TITLE ? "<B>Up:</B> $UP_TITLE\n" : undef) .
    # ... and the ''previous'' title
       ($PREVIOUS_TITLE ? "<B> Previous:</B> $PREVIOUS_TITLE\n" : undef) .
    #  Horizontal rule (3-d dividing line) and new paragraph
       "<HR> <P>\n"
}
```

Figure 1: Sample Perl subroutine to be inserted in the LATEX2HTML configuration or startup file to customize the document's top navigation panel

On the other hand, the following text color commands are processed as ordinary text by LaTeX2HTML, if they occur outside of a command or environment that would otherwise be passed to LaTeX for image conversion: `\color`, `\textcolor`, `\pagecolor`, `\colorbox`, `\fcolorbox`.

Macro interactions

If a LaTeX graphics package requires one or more TeX definitions to made within each graphics environment (e.g. `eepic`), there are currently several ways to proceed:

1. If it is possible to define the required macros only once, do that at the top of the document. LaTeX2HTML will expand each reference to the macros, before passing them on to LaTeX for image conversion.
2. If the TeX macros need to *change* with each figure, it is not possible to make them within each figure. LaTeX2-HTML will expand all references to the macros according to their *last* definition in the document. Instead, place the entire figure, macros and all, into a separate file, say `chart.eepic`. Then in your main document, insert "`\fbox{\InputIfFileExists{chart.-eepic}}`." The command `\InpuIfFileExists` is not reqognized by LaTeX2HTML, and macros contained within it will be left untouched. It *is* recognized by LaTeX, however, and will be processed by it correctly.
3. Another method of having multiple, variable macro definitions is to make them with native TeX `\def` commands, and to use the `-no_tex_defs` command-line switch. With this switch, TeX `\def`'s are not recognized by LaTeX2HTML, and will be left unexpanded as they are copied to `images.tex`.

Scaling and orientation

Images created by this mechanism can be divided into two categories: "small" and "large." For purposes of discussion …

"small images" refers to equations, special accents and any other image generating LaTeX commands; while …

"figures" applies to any image-generating LaTeX environments (e.g. `figure`, `table`, `minipage` etc.)

The size of all "small images" depends on a configuration variable `MATH_SCALE_FACTOR` which specifies how much to enlarge or reduce them in relation to their original size in the PostScript version of the document. For example a scale factor of 0.5 will make all images half as big while a scale factor of 2 will make them twice as big. Larger scale factors result in longer processing times and larger intermediate image files. A scale factor will only be effective if it is greater than 0. The configuration variable `FIGURE_SCALE_FACTOR` performs a similar function for "figures". Both of these configuration variables are initially set to 1.6.

For finer control, several parameters affecting the conversion of a single "figure" can be controlled with the command `\htmlimage`, which is defined in `html.sty`. The one argument of `\htmlimage` is a string of options separated by commas. The options are:

- `scale=` <scale factor>
- `external`
- `thumbnail=` <scale factor>
- `map=` <server-side image map URL>
- `usemap=` <client-side image map URL>
- `flip=` <flip_option>
- `align=` <alignment>

The `scale=` option allows control over the size of the final image. It overrides the global `FIGURE_SCALE_FACTOR` for this one figure.

The `external` option will cause the image not to be inlined. (Images are inlined by default.) External images will be accessible via a hypertext link.

The `thumbnail=` option will cause a small inlined image to be placed in the caption. The size of the thumbnail depends on the <scale factor>. Use of the `thumbnail=` option implies the `external` option.

The `map=` and `usemap=` options specify that the image is to be made into an active image map (See below).

The `flip=` option specifies a change of orientation of the electronic image relative to the printed version. The <flip_option> is any single command recognized by *pnmflip*. The most useful of these include:

rotate90 or r90 This will rotate the image clockwise by 90°.

rotate270 or r270 This will rotate the image counter-clockwise by 90°.

leftright This will flip the image around a vertical axis of rotation.

topbottom This will flip the image around a horizontal axis of rotation.

The `align=` option specifies how the `figure` will be aligned. The choices are: `top`, `bottom`, `middle`, `left`, `right` and `center`. The `middle` option specifies that the image is to be left-justified in the line, but centred vertically. The `center` option specifies that it should also be centred horizontally. This option is valid only if the *HTML* version is 3 or higher, or if the `NET_SCAPE_HTML` configuration variable is set. The default value is `bottom`.

In order to be effective the command `\htmlimage` and its options must be placed inside the environment on which it will operate.

Active Image Maps

Image maps are images with active regions in which a Web surfer can click, to send him off to another sector of cyberspace. LaTeX2HTML can design either inline "figures" or external ones (with or without a thumbnail version) to be image maps. However, *HTML* requires a URL of an *HTML map-file* which defines the coordinates of each active region in the map with a destination URL. Usually this map file is kept on the server machine, however *HTML* version 3 also allows it to reside on the client side for faster response. (See `<http://ds.internic.net/internet-drafts/draft-seidman-clientsideimagemap-02.txt>` Both configurations are supported by LATEX2-HTML through the `\htmlimage` options "`map=` " and "`usemap=` ," respectively.

Keeping such a map file up to date manually can be tedious, especially with dynamic documents under revision. An experimental program *makemap* can help automate this process. This program (which is really a *Perl* script) takes one mandatory argument and one optional argument. The mandatory argument is the name of a *user-map* file, defined below. The optional argument is the name of the directory where the *HTML* map file(s) are to be placed.

The best way of describing how this works is by example. Suppose that a document has two figures designated to become active image maps. The first figure included a statement like,

```
\begin{figure}
\htmlimage{map=/cgi-bin/imagemap/Blk.map}
. . .
\end{figure}
```

The second figure had a line like,

```
\begin{figure}
\htmlimage{map=/cgi-bin/imagemap/Flow.map}
. . .
\end{figure}
```

A typical user map file, named `report.map`, might contain the following information[1]:

```
+report/  URL
#
#  Define map #1:
#
Blk.map:
label1  rect    288,145 397,189
label2  rect 307,225 377,252
label2  default
#
#  Define map #2
#
Flow.map:
label3  circle  150,100 200,100
label4  default
```

In this file, comments are denoted by a #-sign in column 1. The line beginning with `+report` states that the symbolic labels are to be found in the `labels.pl` contained in the directory `report/`, and that its associated URL is as stated. Any number of external `labels.pl` files may be so specified. The block diagram image has two active regions. The first is a rectangle bounded by corners (288, 145) and (397, 189), while the second is a rectangle bounded by corners (307, 225) and (377, 252). These coordinates can be obtained with the aid of a program such as *xv*. If the user clicks in the first rectangle, it will cause a branch to the URL associated with symbolic label `label1` defined in the `labels.pl` file found in directory `report/`. The single active region in the flow chart figure is a circle centred at (150, 100) and passing through point (200, 100). Clicking in this region will cause a branch to symbolic label `label3`. Labels `label2` and `label4` will be visited if the user clicks anywhere outside of the explicit regions. If any labels are not defined in any of the `labels.pl` files mentioned, they will be interpreted as URLs without translation.

The *HTML* image maps are generated and placed in directory `report/` by invoking the command: "**makemap report.map report**."

Fancy lists

An optional style file `htmllist.sty` has been provided which produces fancier lists in the electronic version of the document. This file defines a new LATEX environment, `htmllist`, which causes a user-defined item-mark to be placed at each new item of the list, and which causes the optional description to be displayed in bold letters. The mark is determined by the `\htmlitemmark { <item-mark> }` command. This command accepts either a mnemonic name for the <item-mark>, from a list of icons established at installation, or the URL of a mark not in the installation list. The command `\htmlitemmark` must be used inside the `htmllist` environment in order to be effective, but it may be used more than once to change the mark within the list. The item-marks supplied with LATEX2HTML are `BlueBall`, `RedBall`, `OrangeBall`, `GreenBall`, `PinkBall`, `PurpleBall`, `WhiteBall` and `YellowBall`. The `htmllist` environment is identical to the `description` environment in the printed version.

An example of its usage is:

```
\begin{htmllist}
\htmlitemmark{WhiteBall}%
\item[Item 1:] This will have a
               white ball.
\item[Item 2:] This will also have a
               white ball.
\htmlitemmark{RedBall}%
\item[Item 3:] This will have a red ball.
\end{htmllist}
```

Change bars

LATEX2HTML supports the LATEX2$_\varepsilon$ `changebar.sty` package, written by Johannes Braams `<JLBraams@cistron.nl>`, for inserting *change-bars* in a document in order to indicate differences from previous versions. Changed text is surrounded by bracket GIF icons.

[1]This file is designed for an NCSA server. CERN servers use "`rect`" instead of "`rectangle`," specify a radius instead of an outer point in the circle, and enclose point coordinates by parentheses.

Document Segmentation

As we have seen, the LATEX author can provide these links either manually or symbolically. Manual links are more tedious because a URL must be provided by the author for every link, and updated every time the target documents change. Symbolic links are more convenient, because the translator keeps track of the URLs. Earlier releases of LATEX2HTML required the entire document to be processed together if it was to be linked symbolically. However, it was easy for large documents to overwhelm the memory capacities of moderate sized computers. Furthermore, processing time could become prohibitively high, if even a small change required the entire document to be reprocessed.

For these reasons, program segmentation was developed. This feature enables the author to subdivide his document into multiple *segments*. Each segment can be processed independently by LATEX2HTML. Hypertext links between segments can be made symbolically, with references shared through auxiliary files. If a single segment changes, only that segment needs to be reprocessed (unless a label is changed that another segment requires). Furthermore, the entire document can be processed without modification by LATEX to obtain the printed version. The top level segment that LATEX reads is called the *parent* segment. The others are called *child* segments.

Program segmentation does require a little more work on the part of the author, who will now have to undertake some of bookkeeping formerly performed exclusively by LATEX2HTML. The following four LATEX extensions carry out segmentation:

`\segment{`<file>`}{`<sec-type>`}{`<heading>`}`
: This command indicates the start of a new program segment. The segment resides in `<file>.tex`, represents the start of a new LATEX sectional unit of type <sec-type> (e.g., `\section`, `\chapter`, etc.) and has a heading of <heading>. A variation of this command, `\segment*`, is provided for segments that are not to appear in the table of contents. These commands perform the following operations in LATEX:

  1. The specified sectioning command is executed.
  2. LATEX will write its section and equation counters into an auxiliary file, named `<file>.ptr`. It will also write an `\htmlhead` command to this file. This information will tell LATEX2HTML how to initialize itself for the new document segment.
  3. LATEX will then proceed to input and process the file `<file>.tex`.

  The `\segment` and `\segment*` commands are ignored by LATEX2HTML.

`\internal[`<file>`]{`<prefix>`}`
: This command directs LATEX2HTML to load inter-segment information of type <type> from the file `<prefix><type>.pl`. Each program segment must be associated with a unique filename-prefix, specified either through a command-line option, or through the installation variable `$AUTO_PREFIX`. The information <type> must be one of the following:

  internals This is the default type, which need not be given. It specifies that the internal labels from the designated segment are to be input and made available to the current segment.

  contents The table of contents information from designated segment are to be made available to the current segment.

  sections Sectioning information is to be read in. Note that the segment containing the table of contents requires both contents and sections information from all other program segments.

  figure Lists of figures from other segments are to be read.

  table Lists of tables from other segments are to be read.

  index Index information from other segments are to be read.

`\startdocument` The `\begin{document}` and `\end{document}` statements are contained in the parent segment only. It follows that the child segments cannot be processed separately by LATEX, without modification. However they can be processed separately by LATEX2HTML, provided it is told where the end of the LATEX preamble is; this is the function of the `\startdocument` directive. It substitutes for `\begin{document}` in child segments, but is otherwise ignored by both LATEX and LATEX2HTML.

`\htmlhead{`<sec-type>`}{`<heading>`}` This command is not normally placed in the document at all. It is automatically passed from parent to child via `<file>.ptr`. It identifies to LATEX2HTML that the current segment is a LATEX sectional unit of type <sec-type>, with the specified heading. This command is ignored by LATEX itself.

A segmentation example

The best way to illustrate document segmentation is through a simple example. Suppose that a document is to be segmented into one parent and two child segments. Let the parent segment be `report.tex`, and the the two child segments be `sec1.tex`, and `sec2.tex`. The latter are translated with filename prefixes of `s1` and `s2`, respectively. This example is included with the version 96.1 distribution of LATEX2HTML, with more prolific comments than are shown here.

The text of `report.tex` is given below:

```
\documentclass{article}
\usepackage{html,makeidx,color}

\internal[figure]{s1}
\internal[figure]{s2}
\internal[sections]{s1}
\internal[sections]{s2}
\internal[contents]{s1}
\internal[contents]{s2}
```

```
\internal[index]{s1}
\internal[index]{s2}

\begin{document}
\title{A Segmentation Example}
\date{\today}
\maketitle
\tableofcontents
\listoffigures

% Process the child segments:

\segment{sec1}{section}{Section 1 title}
\segment{sec2}{section}{Section 2 title}
\printindex
\end{document}
```

This file obtains the information necessary to build an index, a table of contents and a list of figures from the child segments. It then proceeds and typesets these.

The first child segment, `sec1.tex`, is shown below:

```
\begin{htmlonly}
\documentclass{article}
\usepackage{html,color,makeidx}
\input{sec1.ptr}
\end{htmlonly}
\internal{s2}
\startdocument
Here is some text.
\subsection{First subsection}
Here is subsection
    1\label{first}.
\begin{figure}
\colorbox{red}{Some red
    text\index{Color text}}
\caption[List of figure
    caption]{Figure 1 caption}
\end{figure}
Reference\index{Reference} to
    \ref{second}.
```

The first thing this child segment does is establish the LATEX packages it requires, then loads the counter information that was written by the `\segment` command that invoked it. Since this segment contains a symbolic reference (`second`) to the second segment, it must load the internal labels from that segment.

The final segment, `sec2.tex`, is shown below:

```
\begin{htmlonly}
\documentclass{article}
\usepackage{html,makeidx}
\input{sec2.ptr}
\end{htmlonly}
\internal{s1}
\startdocument
Here is another section\label{second}.
Plus another\index{Reference, another}
reference\ref{first}.
\begin{figure}
\fbox{The figure}
\caption{The caption}
\end{figure}
```

This segment needs to load internal labels from the first one, because of the reference to `first`. These circular dependencies (two segments referencing each other) are either not allowed or handled incorrectly by the Unix utility `make`, without resorting to time stamps and some trickery. A *time-stamp* is a zero-length file whose only purpose is to record its creation time. Besides evaluating segment interdependence, another function of `make` is to provide intersegment navigation information.

A sample `Makefile` is included in the distribution. This correctly generates the fully-linked document. The first time it is invoked, it runs:

- LATEX on `report.tex` twice;
- `dvips` to generate `report.ps`;
- LATEX2HTML on `sec1.tex`;
- LATEX2HTML on `sec2.tex`. At this point, `sec2.html` is completely linked, since the labels from the `sec1` were available;
- LATEX2HTML on `sec1.tex` to pick up the labels from `sec2`;
- LATEX2HTML on `report.tex`.

Proper operation of `make` depends on the fact that LATEX2-HTML updates its own internal label file only if something in its current program segment causes the labels to change from the previous run. This ensures that LATEX2HTML is not run unnecessarily. It is also usual for the information page to be suppressed by specifying `-info` on all but the top-level program segment.

Extending LATEX2HTML

As the translator covers only partially the set of LATEX commands and because new LATEX commands can be defined arbitrarily using low level TeX commands, the translator should be flexible enough to allow end-users to specify how they want particular commands to be translated.

Adding Support for Specific Style Files

LATEX2HTML provides a mechanism where code to translate specific style files is automatically loaded, if such code is available. When use of a style, such as `german.sty`, is detected in a LATEX source document, the translator looks for a file `LATEX2HTMLDIR/styles/german.perl`. If one is found, then it will be loaded into the main script.

This mechanism will help to keep the core script smaller as well as make it easier for others to contribute and share solutions on how to translate specific style files. The current distribution includes the files listed in Table 1. These will provide good examples of how you can create your own extensions to LATEX2HTML.

`.perl` *file*	*Description*
alltt	Supports the LATEX2$_\varepsilon$'s `alltt` package.
changebar	Provides rudimentary change-bar support.
color	Causes colored text to be processed as ordinary text by LATEX2HTML.
french	Special support for the French language.
epsfig	Processes embedded figures not enclosed in a `figure` environment.
floatfig	Processes floating figures.
german	Special support for the German language.
graphics	Supports commands in the `graphics` package.
graphicx	Supports the alternate syntax of graphics commands.
heqn	Alters the way displayed equations are processed.
htmllist	Provides support for fancy lists.
makeidx	Generates the index.
texdefs	Supports some raw TEX commands.
wrapfig	Supports wrapped figures.

Tabel 1: Supported LATEX2HTML style files.

The problem however, is that writing such extensions requires an understanding of *Perl* and of the way LATEX2-HTML is organised. Interfaces that are more "user-friendly" will be investigated.

At the moment a rudimentary mechanism is provided so that a user can ask for particular commands and their arguments either to be ignored or passed on to LATEX for processing (the default behaviour for unrecognized commands is for their arguments to remain in the *HTML* text). Commands that are passed on to LATEX are converted to images which are either "inlined" in the main document or become accessible via hypertext links. Simple extensions using the commands above may be included in the `$LATEX2HTMLDIR/latex2html.config` file or in each personal `$HOME/.latex2html-init` initialization file.

Adding ignored Commands

Commands that should be ignored may be specified in the `.latex2html-init` file as input to the `ignore_commands` subroutine. Each command which is to be ignored should be on a separate line followed by compulsory or optional argument markers separated by #'s e.g.[2]:

```
<cmd_name>#{}# []# {}# [] ...
```

`{}`'s mark compulsory arguments and `[]`'s optional ones.

Some commands may have arguments which should be left as text even though the command should be ignored (e.g. `\mbox`, `\`
,
etc.). In these cases arguments should be left unspecified.

Here is an example of how this mechanism may be used:

```
&ignore_commands( <<_IGNORED_CMDS_);
documentstyle # [] # {}
linebreak# []
center
<add your commands here>
_IGNORED_CMDS_
```

Passing Commands to LATEX

Commands that should be passed on to LATEX for processing because there is no direct translation to `HTML` may be specified in the `.latex2html-init` file as input to the `process_commands_in_tex` subroutine. The format is the same as that for specifying commands to be ignored. Here is an example:

```
&process_commands_in_tex
  (<<_RAW_ARG_CMDS_);
fbox # {}
framebox # [] # [] # {}
<add your commands here>
_RAW_ARG_CMDS_
```

New Features

A number of new features were introduced in the 96.1 version of LATEX2HTML. This changes were provided by the users of this program. Some of these have already been discussed. The more significant ones include the following:

Font generation: The quality of equation bitmaps is greatly improved by enabling the `PK_GENERATION` configuration variable. When this is done *Metafont* will be invoked through `dvips` to generate fonts more suitable to screen viewing. Ideally, this should be done by setting the `mode=` switch to `dvips`, but unfortunately not all versions of `dvips` support this option. For those that don't, a `.dvipsrc` file is supplied with this distribution.

Active image-maps: Both server and client-side maps are supported. Image-maps can either be inline or external. External maps can be associated with a thumbnail image. A separate script `makemap` helps resolve external URLs in image-maps.

Improved image recycling: The older version of image recycling often caused images to overwrite each other due to confusion in the bookkeeping. This has been fixed. It is also no longer necessary to keep generating the same image which is being used repeatedly in a document. Furthermore, images with thumbnails can now be recycled, as well as active image-maps. Only images of the correct size are recycled.

[2]It is possible to add arbitrary Perl code between any of the argument markers which will be executed when the command is processed. For this however a basic understanding of how the translator works and of course *Perl* is required.

Document segmentation: Large documents can now be divided into independently processed segments. Additional command-line switches provide inter-segment navigation information, while automatically generated *Perl* files pass symbolic references and counter information between segments.

Command parsing: Constructions such as `\hello2` are now treated as macro `\hello` followed by argument '2', rather than as macro '`\hello2`'. Added `\makeatletter` and `\makeatother` commands.

Graceful termination: When LaTeX2HTML is interrupted by a termination signal, all child tasks are also terminated. This provides a more orderly shut-down than that offered by previous versions.

More inline-math: Small inline equations which can be typeset in *HTML are* typeset in *HTML*. This further reduces the number of GIF files that need to be generated.

External hypertext: Commands `\html\-ref` and `\hyper\-ref` now accept labels defined in an external document, if the internal reference is not found.

Additional style files:

htmllist Defines a fancier list environment which embellishes each item of a description list with a user-selected icon. (It is the same as the `description` environment in the paper version.)

heqn Redefines the `equation` environment so that equation numbers are handled in *HTML*. This causes equations in this environment to be recyclable. It also causes equation arrays to be recyclable if their equation numbers do not change from the previous run.

floatfig Provides support for this environment by making it look like an ordinary figure in the electronic version.

wrapfig (Same comment as for `floatfig`).

graphics Defines elements of the standard LaTeX2$_\varepsilon$ `graphics.sty` package.

LaTeX2$_\varepsilon$ support: Provided support for the `\ensuremath` command. The last *option* to the `babel` package is interpreted as the LaTeX2HTML language style-file to load. User-defined commands and environments can now have an optional argument. Stubs have been provided for `\enlargethispage` and `\suppressfloats`. LaTeX2$_\varepsilon$ packages `alltt`, `graphics`, `graphicx`, `color`, `changebar` and `epsfig` were provided.

Figure orientation: The `flip=` option of `\htmlimage` causes the program `pnmflip` to be called prior to the generation of the GIF file. This allows the electronic version of the image to be oriented differently than the paper version.

Null images: If for some reason an image produced a null GIF file, then no reference is made to that file and the program proceeds gracefully. Furthermore, such null images do not need to be regenerated. This would occur in *HTML* 3.0 images whose caption is enclosed in a `\parbox`, for example.

Navigation panels: There are now separate subroutines to control the top and bottom navigation panels.

Section headings: Labels, equations and images are now permitted inside a section heading, and in the `\title` command. However, the equations may not look very good because of the size difference.

Future directions

Work is currently underway on a complete rewrite of LaTeX2HTML. The new version will be based much more on TeX's grammar and parsing. According to Marcus E. Hennecke, doctoral student at Stanford University, the rewrite (probably to be named V96.3) "does away with any kind of preprocessing, thus eliminating *texexpand*, `DBM` files, etc. and instead works directly on the raw LaTeX code. I would imagine that it should be possible to run that version on a PC except for the image generation, which will be the only thing requiring external helper programs." One minor drawback of V 96.3 is that it will *require* users to upgrade to *Perl* 5. For this reason, versions based on V96.1 will continued to be supported in parallel for some time to come.

Hennecke emphasizes that LaTeX2HTML V96.3 is not, by itself, a port to non-UNIX platforms. However, it should make such a port easier to achieve, since it will store all system dependencies in variables, rather hard-coded into the source. Anyone contemplating working on a port from UNIX should corroborate with him directly at `<marcush@crc.ricoh.com>`.

User Support

To keep in touch with other users of LaTeX2HTML, to get additional assistance, or to suggest or provide bug fixes, you can join the online mailing list, provided through the Argonne National Labs. To subscribe to this list, send a message to: `<latex2html-request@mcs.anl.gov>` with the contents: **subscribe** . To be removed from the list send a message to the same address with the contents: **unsubscribe** . All recent postings to this discussion group are archived in a web-browsable at `<http://www.rosat.mpe-garching.mpg.de/mailing-lists/LaTeX2HTML/>`. You may not get immediate answers to all your question, but most inquiries are answered eventually. Furthermore, comments, suggestions and critiques are all helpful for the the continuing development of this evolving program.

[3]`<http://cbl.leeds.ac.uk/nikos/tex2html/>`

Acknowledgements

LaTeX2HTML was originally written by Nikos Drakos[3] while he was with the Computer Based Learning Unit of the University of Leeds. The material in this article was drawn heavily on the LaTeX2HTML user's documentation, as revised most recently by myself, Ross Moore[4] and Michel Goossens[5]. Thanks also go to Donald Arseneau[6] for his `url.sty` LaTeX package, which was instrumental in formatting this article.

[4]email: `<ross@mpce.mq.edu.au>`
[5]email: `<goossens@cern.ch>`
[6]email: `<asnd@trumf.ca>`