The LaTeX2HTML Translator

Nikos Drakos
Computer Based Learning Unit
University of Leeds.

June 7, 1996

Documentation revised and updated by:

Ross Moore
Mathematics Department
Macquarie University, Sydney.

This document is up-to-date with LaTeX2HTML V96.1, revision f.[1,2]

(Where the terms 'version 96.2' or 'V96.2' are used, we actually mean this revision f.)

Abstract

LaTeX2HTML is a conversion tool that allows documents written in LaTeX to become part of the World-Wide Web. In addition, it offers an easy migration path towards authoring complex hyper-media documents using familiar word-processing concepts.

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 extends LaTeX by supporting arbitrary hypertext links and symbolic cross-references between evolving remote documents. It also 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.

This document presents the main features of LaTeX2HTML and describes how to obtain, install and use it.

[1]The manuscript was updated to version 96.1, as indicated with change-bars, by Herbert W. Swan `<dprhws@edp.Arco.com>` and converted to LaTeX 2_ε by Michel Goossens `<goossens@cern.ch>`.

[2]For version 96.2, updates and extensive revisions to the manuscript were made by Ross Moore `<ross@mpce.mq.edu.au>`, also incorporating suggestions from Michel Goossens.

Credits

Several people have contributed suggestions, ideas, solutions, support and encouragement. Some of these are Roderick Williams, Ana Maria Paiva, Jamil Sawar and Andrew Cole here at the Computer Based Learning Unit.

CERN The idea of splitting LaTeX files into more than one component, linked with hyperlinks, was first implemented in *Perl* by Toni Lantunen at CERN. Thanks to Robert Cailliau `<cailliau@cernnext.cern.ch>` of the World-Wide Web Project also at CERN for giving me access to the source code and documentation (although no part of the original design or the actual code has been used).

Robert S. Thau `<rst@edu.mit.ai>` has contributed the new version of `texexpand`. Also, in order to translate the "document from hell" (!!!) he has extended the translator to handle `\def` commands, nested math-mode commands, and has fixed several bugs.

Phillip Conrad and L. Peter Deutsch. The `pstogif` *Perl* script uses the `pstoppm.ps` PostScript program, originally written by Phillip Conrad (Perfect Byte, Inc.) and modified by L. Peter Deutsch (Aladdin Enterprises).

Roderick Williams. The idea of using existing symbolic labels to provide cross-references between documents was first conceived during discussions with Roderick Williams `<rodw@cbl.leeds.ac.uk>`.

Eric Carroll `<eric@ca.utoronto.utcc.enfm>`, who first suggested providing a command like `\hyperref`.

Franz Vojik `<vojik@de.tu-muenchen.informatik>` provided the basic mechanism for handling foreign accents.

Todd Little. The `-auto_navigation` option was based on an idea by Todd `<little@com.dec.enet.nuts2u>`.

Axel Belinfante `<Axel.Belinfante@cs.utwente.nl>` provided the *Perl* code in the `makeidx.perl` file, as well as numerous suggestions and bug-reports.

Verena Umar `<verena@edu.vanderbilt.cas.compsci>` (Computer Science Education Project[3]) has been a very patient tester of some early versions of LaTeX2`HTML` and many of the current features are a result of her suggestions.

Ian Foster and Bob Olson. Thanks to Ian Foster `<itf@mcs.anl.gov>` and Bob Olson `<olson@mcs.anl.gov>` at the Argonne National Labs, for setting up the LaTeX2`HTML` mailing list[4].

[3] http://csep1.phy.ornl.gov/csep.html
[4] http://cbl.leeds.ac.uk/nikos/tex2html/doc/mail/mail.html

Recent Developments

Since 1995 the power and usefulness of LaTeX2HTML has been enhanced significantly. The revisions later than V95.1 have been largely due to the combined efforts of many people, other than the original author. Interested users have supplied patches to fix a fault, or implement a feature that previously was not supported. Often a question or complaint to the discussion-group (see Section 2.2) has spurred someone else to provide the necessary "patch".

Arising from this work, special credit is due to:

Marcus Hennecke `<marcush@crc.ricoh.com>` for his many extensive revisions;

Mark Noworolski `<jmn@eecs.berkeley.edu>` for coordinating V95.3;

Sidik Isani `<isani@cfht.hawaii.edu>` for his improvement in GIF quality;

Michel Goossens `<goossens@cern.ch>` was the driving force behind the upgrade to LaTeX 2_ε compatibility, and provides a link to the World-Wide Web Project at CERN;

Herb Swan `<dprhws@edp.Arco.com>` for coordinating V96.1 of LaTeX2HTML, including much of the *Perl* code for the new features that were introduced, and for providing a series of bug-fix revisions prior to V96.2;

Ross Moore `<ross@mpce.mq.edu.au>` who has revised and extended this manual, helped design and test the segmentation strategy, and other revisions for V96.2;

Alex Lopez-Ortiz `<alopez-o@neumann.UWaterloo.ca>` for coordinating the V96.2 release of LaTeX2HTML;

Marcus Hennecke `<marcush@crc.ricoh.com>` again, who has a major revision planned for V96.3. This should greatly improve the speed and efficiency of LaTeX2HTML, as well as allowing for portability to Unix systems on a variety of platforms. However it will require use of *Perl* 5.

Many others, too many to mention, have contributed bug-reports, fixes and other suggestions.

Keep them coming!

Thanks also to Donald Arseneau `<asnd@triumf.ca>` for allowing his `url.sty` to be distributed with this manual. Similarly, thanks to Johannes Braams `<JLBraams@cistron.nl>` for `changebar.sty`.
Both of these are useful utilities which enhance the appearance of the printed manual.In particular, changes introduced with V96.1 are denoted by thin change-bars, while thicker bars denote changes introduced with V96.2.

General License Agreement and Lack of Warranty

This software is distributed in the hope that it will be useful but *without any warranty*. The author(s) do not accept responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all. No warranty is made about the software or its performance.

Use and copying of this software and the preparation of derivative works based on this software are permitted, so long as the following conditions are met:

- The copyright notice and this entire notice are included intact and prominently carried on all copies and supporting documentation.
- No fees or compensation are charged for use, copies, or access to this software. You may charge a nominal distribution fee for the physical act of transferring a copy, but you may not charge for the program itself.
- If you modify this software, you must cause the modified file(s) to carry prominent notices (a `ChangeLog`) describing the changes, who made the changes, and the date of those changes.
- Any work distributed or published that in whole or in part contains or is a derivative of this software or any part thereof is subject to the terms of this agreement. The aggregation of another unrelated program with this software or its derivative on a volume of storage or distribution medium does not bring the other program under the scope of these terms.

This software is made available *as is*, and is distributed without warranty of any kind, either expressed or implied. In no event will the author(s) or their institutions be liable to you for damages, including lost profits, lost monies, or other special, incidental or consequential damages arising out of or in connection with the use or inability to use (including but not limited to loss of data or data being rendered inaccurate or losses sustained by third parties or a failure of the program to operate as documented) the program, even if you have been advised of the possibility of such damages, or for any claim by any other party, whether in an action of contract, negligence, or other tortuous action.

Contents

[5]This feature is supported only for users of LaTeX 2_ε.

List of Figures

List of Tables

[6]http://csep1.phy.ornl.gov/csep.html

1 Overview

This document describes the LaTeX2`HTML` translator which:

- breaks up a document into one or more components as specified by the user[7];
- provides optional, customisable iconic navigation panels on every page which contain links to other parts of the document, or other documents;
- handles inlined equations ($\sum_{i=1}^{n} x_i = \int_0^1 f$), handles equation alignment ($A_{B_{C+D}}$), right-justified numbered equations (see equation 2), tables (see Table 2), figures (see Figure 2), and *any arbitrary environment*[8];
- figures or tables can be arbitrarily scaled and oriented, and shown either as inlined images or "thumbnail" sketches;
- can produce output suitable for browsers that support inlined images or character-based browsers (as specified by the user);
- handles definitions of new commands, environments and theorems even when these are defined in external style-files[9];
- handles footnotes[10], tables of contents, lists of figures and tables, bibliographies and can generate an index;
- translates cross-references into hyper-links and extends the LaTeX cross-referencing mechanism to work not just within a document but *between documents* which may reside in remote locations;
- translates LaTeX accent and special character commands (e.g. Ȧ Øö £©¶) to the equivalent ISO–Latin–1 or Unicode character set, where possible;
- recognises hypertext links (to multi-media resources or arbitrary Internet services such as *sound/video/ftp/http/news*) and links which invoke arbitrary program scripts—all expressed as LaTeX commands;
- recognises *conditional text* which is intended only for the hypertext version, or only for the paper (`.dvi`) version;
- can include raw `HTML` in a LaTeX document (e.g. in order to specify interactive forms);
- can deal sensibly with at least the *Common LaTeX Commands* summarised at the back of the LaTeX blue book[1], and many other LaTeX constructions;
- will try to translate any document with embedded LaTeX commands, irrespective of whether it is complete or syntactically legal;
- can be configured to translate equations and/or tables either as GIF images or as `HTML` 2.1 or 3.0 mark-up, as browsers become available which are suitable for the task;

[7] The user can specify the *depth* at which the document should not be broken up any further.

[8] These are passed to LaTeX and then converted to images which are either included in the document or are made available through hypertext links.

[9] This allows the definition of `HTML` macros in LaTeX !

[10] Like this!

- links symbolic references across document segments which are independently processed.

A selection of documents illustrating different contexts in which LaTeX2HTML has been used is available at `http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/`.

2 Installation and Further Support

2.1 Getting LaTeX2HTML

One way LaTeX2HTMLmay be obtained is through one of the three 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>`. In the directory `tex-archive/support/latex2html` should be the latest version, uncompressed.
Alternatively, a compressed `tar` file of the source and related files may be obtained via *anonymous ftp* to `http://www-dsed.llnl.gov/files/programs/unix/latex2html/sources/latex2html-96.1.tar.gz`.
Two other *ftp*-sites are `ftp://ftp.mpn.com/pub/nikos/latex2html-96.1.tar.gz` and `ftp://ftp.rzg.mpg.de/outgoing/latex2html-96.1.tar.gz`.
Other *ftp*-sites nearer to you can be found using *Archie* at `http://hoohoo.ncsa.uiuc.edu/archie.html` or `http://www.pvv.unit.no/archie/` (faster).

> **Warning:** Some *ftp*-sites may not carry the latest version.

Updates and patches are posted on the LaTeX2HTML server at `http://www-dsed.llnl.gov/files/programs/unix/latex2html/`.
If you get the compressed `tar` version, save it into a file `latex2html-96.1.tar.gz` say, then extract its contents with

```
% gzip -d latex2html-96.1.tar.gz
% tar xvf latex2html-96.1.tar
```

You should then have the following:

- `README` file;
- `Changes` file;
- `latex2html` *Perl* script;
- `texexpand` *Perl* script[11];
- `latex2html.config` configuration file;
- `install-test` *Perl* script, for installation and testing;
- `dot.latex2html-init` sample initialization file;
- `texinputs/` subdirectory, containing various LaTeX style-files;
- `versions/` subdirectory, containing code for specific HTML versions;
- `makemap` *Perl* script;

[11] Written by Robert S. Thau `<rst@edu.mit.ai>`.

- `example/` subdirectory, containing the segmentation example, described in detail in Section 4.6;
- `.dvipsrc` file;
- `pstogif` *Perl* script;
- two `pstoppm.ps` files;
- `docs/` subdirectory, containing the files needed to create a version of this manual;
- `icons/` subdirectory, containing various icons;
- `styles/` subdirectory, containing *Perl* code for handling some style-files.

2.1.1 Requirements

The translator makes use of several utilities all of which are freely available on most platforms. You may use *Archie* to find the source code of any utilities you might need.[12]

The requirements for using LaTeX2HTML depend on the kind of translation you would like to perform, as follows:

1. **LaTeX commands but without equations, figures, tables, etc.**
   - *Perl* `(version 4.0 - RCSfile: perl.c,v - Revision: 4.0.1.8 - Date: 1993/02/05 19:39:30 - Patch level: 36)`.
     Warning: You really *do* need *Perl* at patch level 36 or later.
     Versions of LaTeX2HTML earlier than 0.7a4 work *only* with *Perl* 4 at patch level 36.
     Versions of LaTeX2HTML up to V96.2 work both with *Perl* 4 at patch level 36 and *Perl* 5.
     No version of LaTeX2HTML will work with *Perl* 4 at earlier patch levels.
     Future versions, from V96.3 onwards, will require *Perl* 5.
   - *DBM* or *NDBM*, the Unix DataBase Management system.
2. **LaTeX commands with equations, figures, tables, etc.**
   As above plus ...
   - `latex`;
   - `dvips` (version 5.516 or later) or `dvipsk`;
   - `gs` (*Ghostscript* version 2.6.1 or later);
   - The `pbmplus` or `netpbm` library. Some of the filters in those libraries are used during the PostScript to GIF conversion.
3. **Segmentation of large documentation**
   If you wish to use this feature, you will have to upgrade your LaTeX to LaTeX 2_ε.
4. **Transparent inlined images**[13]
   If you dislike the ugly 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`[14] 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.

[12] http://www.pvv.unit.no/archie/
[13] http://melmac.corp.harris.com/transparent_images.html
[14] ftp://ftp.rz.uni-karlsruhe.de/pub/net/www/tools/giftrans.c

If *Ghostscript* or the `pbmplus` (or `netpbm`) library are not available, it is still possible to use the translator with the `-no_images` option.

If you intend to use any of the special features of the translator (see page 12) then you have to include the `html.sty` file in any LaTeX documents that use them.

Since by default the translator makes use of inlined images in the final `HTML` output, it would be better to have a viewer which supports the `<IMG>` tag, such as *NCSA Mosaic* or *Netscape Navigator*. If only a character-based browser is available, or if you want the generated documents to be more portable, then the translator can be used with the `-ascii_mode` option (see Section 3.1).

2.1.2 Installing LaTeX2HTML

To install LaTeX2`HTML` you **MUST** do the following:

1. **Specify where *Perl* is on your system.**
   In each of the files `latex2html`, `texexpand`, `pstogif`, `install-test` and `makemap`, modify the first line saying where *Perl* is on your system.

   Some system administrators do not allow *Perl* programs to run as shell scripts. This means that you may not be able to run any of the above programs. *In this case change the first line in each of these programs from*

   ```
   #!/usr/local/bin/perl
   ```

   to

   ```
   : # *-*-perl-*-*
       eval 'exec perl -S  $0 "$@"'
       if $running_under_some_shell;
   ```

2. **Specify where the external utilities are on your system.**
   In the file `latex2html.config` give the correct path-names for some directories (the `latex2html` directory and the `pbmplus` or `netpbm` library) and some executables (`latex`, `dvips`, `gs`). Note that it is possible to use LaTeX2`HTML` even if you don't have some of the external utilities.

   While you're at it you may want to change some default options in the same file.

3. **Run `install-test`.**
   This *Perl* script will make some changes in the `latex2html` file and then check whether the path-names to any external utilities (specified during the previous step) are correct. It will not actually install the external utilities.

   Don't forget to make `install-test` executable (using the `chmod` command) before using it if necessary. You may also need to make the files `pstogif`, `texexpand` and `latex2html` executable if `install-test` fails to do it for you.

   If for any reason you have trouble running `install-test` do not despair. Most of what it does is to do with checking your installation rather than actually installing anything. To do a manual installation just change the variable `$LATEX2HTMLDIR`, in the beginning of the file `latex2html`, to point to the directory where the LaTeX2`HTML` files can be found.

This is enough for the main installation but you may also want to do some of the following:

- **To use the new LaTeX commands which are defined in `html.sty`:**
Make sure that LaTeX knows where the `html.sty` file is, either by putting it in the same place as the other style-files on your system, or by changing your `TEXINPUTS` shell environment variable, or by copying `html.sty` into the same directory as your LaTeX source file. If you are still using LaTeX V2.09, you must change the symbolic link `html.sty` from `html2e.sty` to `html209.sty`. *Also make sure that* `TEXINPUTS` *includes '`..`' else* LaTeX2HTML *will not work properly.* On some systems, the command `latex` is really a shell script which sets some environment variables and calls the real LaTeX. If this is so, make sure that this shell script sets `TEXINPUTS` properly. This environment variable is not to be confused with the LaTeX2HTML installation variable `$TEXINPUTS` described next.

- There is an installation variable in `latex2html.config` called `$TEXINPUTS`, which tells LaTeX2HTML where to look for LaTeX style-files to process. It can also affect the input-path of LaTeX when called by LaTeX2HTML, unless the command `latex` is really a script which overwrites the `$TEXINPUTS` variable prior to calling the real LaTeX. This variable is overridden by the environment variable of the same name if it is set.

- The installation variable `$PK_GENERATION` specifies which fonts are used in the generation of mathematical equations. A value of "`0`" causes the same fonts to used as those for the default printer. Because they were designed for a printer of much greater resolution than the screen, equations will generally appear to be of a lower quality than is possible. To cause LaTeX2HTML to dynamically generate fonts that are designed specifically for the screen, you should specify a value of "`1`" for this variable. If you do, then check to see whether your version of `dvips` supports the command-line option `-mode`. If it does, then also set the installation variable `$DVIPS_MODE` to a low resolution entry from `modes.mf`, such as `toshiba`. It may also be necessary to edit the `MakeTeXPK` script, to recognise this mode at the appropriate resolution.

  If `dvips` does *not* support the `-mode` switch, then leave `$DVIPS_MODE` undefined, and verify that the `.dvipsrc` file points to the correct screen device and its resolution.

- The installation variable `$AUTO_PREFIX` allows the filename-prefix to be automatically set to the base filename-prefix of the document being translated. It can be especially useful for multiple-segment documents.

- On certain *Linux* systems it is necessary to uncomment the line "`use GDBM_File`" in the `latex2html` and `install-test` scripts, to define the interface to the database-management routines.

- The `makemap` script also has a configuration variable `$SERVER`, which must be set to either `CERN` or `NCSA`, depending on the type of Web-server you are using.

- **To set up different initialization files:**
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.

Warning: This initialization file is incompatible with any version of LaTeX2HTML prior to V96.1. Users must either update this file in their home directory, or delete it altogether.

- **To make your own local copies of the LaTeX2HTML icons:**
Please copy the `icons/` subdirectory to a place under your WWW tree where they can be served by your server. Then modify the value of the `$ICONSERVER` variable in `latex2html.config` accordingly.

  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.

- **To make your own local copy of the LaTeX2HTML documentation:**
This will also be a good test of your installation.

  Firstly, to obtain the `.dvi` version for printing, from within the `docs/` directory it is sufficient to type: `make manual.dvi`.
  This initiates the following sequence of commands:

```
latex manual.tex
makeindex -s l2hidx.ist manual.idx
makeindex -s l2hglo.ist -o manual.gls manual.glo
latex manual.tex
latex manual.tex
```

  in which the two configuration files `l2hidx.ist` and `l2hglo.ist` for the `makeindex` program, are used to create the index and glossary respectively. The 2nd run of `latex` is needed to assimilate references, etc. and include the index and glossary. (In case `makeindex` is not available, a copy of its outputs `manual.ind` and `manual.gls` are included in the `docs/` subdirectory, along with `manual.aux`.) The 3rd run of `latex` is needed to adjust page-numbering for the index and glossary within the table-of-contents.

  Next, the `HTML` version is obtained by typing: `make manual.html`.
  This initiates a series of calls to LaTeX2HTML on the separate segments of the manual; the full manual is thus created as a segmented document, see 4.6a later section. The whole process may take quite some time, as each segment needs to be processed at least twice, to collect the references from other segments.

  The files necessary for correct typesetting of the manual to be found within the `docs/` subdirectory. They are as follows.

  – style-files: `l2hman.sty`, `html.sty`, `htmllist.sty`, `justify.sty`, `changebar.sty` and `url.sty`
  – inputs: `changes.tex`, `credits.tex`, `features.tex`, `hypextra.tex`, `licence.tex`, `manhtml.tex`, `manual.tex`, `overview.tex`, `problems.tex`, `support.tex` and `userman.tex`
  – graphics: `eform.ps` and `figure.ps`
  – filename data: `l2hfiles.dat`
  – auxiliaries: `manual.aux`, `manual.ind`, `manual.gls`

  The last three can be derived from the others, but are included for convenience.

- **To get a printed version of the 'Changes' section:** Due to the burgeoning size of the `Changes` file with successive revisions of LaTeX2HTML, the 'Changes' section is no longer normally included as part of the printed version of the manual. If you want this, then find the line `\input{changes.tex}`, within `docs/manual.tex`, and comment-out the surrounding htmlonly environment. Remake the manual as in the preceding item; the repeated runs of `latex` are required to adjust the index, glossary and table-of-contents for the extra information. Up to 20 extra pages may be added.

- **To join the community of LaTeX2HTML users:**
More information on a mailing list, discussion archives, bug reporting forms and more is available at `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`

2.2 Getting Support and More Information

A LaTeX2HTML mailing list has been set up at the Argonne National Labs. The LaTeX2HTML mailing list archive[15] is available. (Thanks to Ian Foster `<itf@mcs.anl.gov>` and Bob Olson `<olson@mcs.anl.gov>`.)

To join send a message to: `<latex2html-request@mcs.anl.gov>` with the contents: `subscribe` .

To be removed from the list send a message to: `<latex2html-request@mcs.anl.gov>` with the contents: `unsubscribe` .

3 User Manual

To use LaTeX2HTML simply type: `latex2html <file>.tex`
The `.tex` suffix is optional and will be supplied by the program if it is 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.

It is possible to customise the output from LaTeX2HTML using a number of command-line options (see Section 3.1) 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, etc.

Also, the LaTeX2HTML script includes a short manual which can be viewed by saying:

```
nroff -man latex2html
```

3.1 Command-Line Options

The command-line options described below can be used to change the default behaviour of LaTeX2HTML. Alternatively the corresponding environment variables in the initialisation 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`.

[15] http://cbl.leeds.ac.uk/nikos/tex2html/doc/mail/mail.html

-link <*num*> For each node, stop revealing child nodes 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 `<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 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 LaTeX2`HTML` to peacefully coexist in the same directory. However, do *not* attempt to *simultaneously* run multiple instances of LaTeX2`HTML` 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).

Section 5.5 provides additional information about image-reuse.

-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 to convert any inlined images that were left over from previous runs of LaTeX2HTML.

-show_section_numbers Show section numbers. By default, section numbers are not shown, so as 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 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.

-short_index Creater a shorter Index, using codified links, when the makeidx package has been loaded.

-debug Run in debug-mode, displaying messages and/or diagnostic information from the utilities used by LaTeX2HTML.

More extensive diagnostics, from the *Perl* debugger, can be obtained by appending the string `-w-` to the 1st line of the `latex2html`, and other, *Perl* script(s).

-h Print out the list of options.

3.2 Extending the Translator

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.

3.2.1 Adding Support for Specific Style-Files

LaTeX2HTML provides a mechanism whereby the code to translate specific style-files is automatically loaded, if such code is available. For example, when use of a style such as `german.sty` is detected in a LaTeX source document, the translator looks for the file `$LATEX2HTMLDIR/styles/german.perl`. If this is found, then it will be incorporated into the main script.

This mechanism helps to keep the core script smaller, as well as making 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 provide good examples of how you can create further extensions to LaTeX2HTML.

`.perl` *file*	*Description*
alltt	Supports the LaTeX 2$_\varepsilon$'s alltt package.
changebar	Provides rudimentary change-bar support.
color	Causes colored text to be processed as ordinary text by LaTeX2HTML.
epsfig	Processes embedded figures not enclosed in a figure environment.
floatfig	Processes floating figures.
french	Special support for the French language.
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.

Table 1: Supported LaTeX2HTML style-files.

The problem however, is that writing such extensions requires an understanding of *Perl* and of the way LaTeX2HTML 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 unrecognised 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 `$LATEX2HTMLDIR/latex2html.config` file or in a personal `$HOME/.latex2html-init` initialization file, as described next.

3.2.2 Asking the Translator to Ignore 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.[16]:

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

`{}`'s mark compulsory arguments and `[]`'s optional ones; spaces are ignored.

Some commands may have arguments which should be left as text even though the command should be ignored (e.g. `\mbox`, `\center`, 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 # [] # {}
```

[16] 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.

```
linebreak # []
center
<add your commands here>
_IGNORED_CMDS_
```

3.2.3 Asking the Translator to Pass 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_
```

4 Hypertext Extensions to LaTeX

This section describes how you can define hypertext entries in your HTML documents from within your LaTeX source. These are implemented as new LaTeX commands which have special meaning during the translation to HTML but are mostly ignored when processed by LaTeX.
All the new commands used in the sections below are defined in the file `html.sty`, which is part of the distribution. *This file must be included in any LaTeX document using these features* by one of the following methods:

- Including `html` as an optional argument to the LaTeX `\documentstyle` command;
- Including `html` in the LaTeX 2_ε `\usepackage` command; or
- Pasting the definitions of individual commands directly into the text!

4.1 Hyper-links in LaTeX

Arbitrary hypertext references are available through the commands `\htmladdnormallink` and `\htmladdimg`. These are defined as follows:

```
\htmladdnormallink{<text>}{<URL>}
\htmladdimg{<URL>}
```

The first command expects some text as the first argument and a URL as the second argument. When processed by LaTeX (i.e. in the `.dvi` or `.ps` output files), the URL will have no effect. But when processed by the translator, the URL will be used to provide an active hypertext link (to another file, picture, sound-file, movie, etc.) e.g.

```
\htmladdnormallink{<URL>}
{http://www.ncsa.uiuc.edu/demoweb/url-primer.html}
```

In a similar way, the argument of the `\htmladdimg` command should be a URL pointing to an image. This URL is ignored in the LaTeX hard copy output.

Another command called `\htmladdnormallinkfoot` takes the same two arguments and has the same effect as `\htmladdnormallink` when generating HTML. However when processed by LaTeX it shows the URL as a footnote.

Warning: The tilde (˜) character is commonly used within hyperlink URLs. It is a quirk of LaTeX that it must be generated via `\~{}`, else the (˜) will be interpreted as an accent on the following character.

4.2 Including Arbitrary HTML Mark-up

Raw `HTML` or `HTML+` tags and text may be included using the environment rawhtml. This can be used to take advantage of new `HTML+` facilities as soon as they become available. A particularly good use of this feature is in the creation of interactive electronic forms from within a LaTeX document. When producing the paper (`.dvi`) version of a document the rawhtml environment is ignored. Here is an example:

```
\begin{rawhtml}
<HR>
<FORM ACTION="http://cbl.leeds.ac.uk/nikos/doc/error.html">
<OL>
<LI> <INPUT TYPE="checkbox" NAME="wp" VALUE="word"> Word for
Windows.
<LI> <INPUT TYPE="checkbox" NAME="wp" VALUE="wp"> Word Perfect.

<LI> <INPUT TYPE="checkbox" NAME="wp" VALUE="latex"> LaTeX.
<LI> Plain Text Editors (Please Specify): <INPUT TYPE="text" NAME="other_ed">
</OL>
So, what do think (comments please): <BR>
<INPUT TYPE="text" SIZE=45,4 NAME="other_wp">

<INPUT TYPE="submit" VALUE="submit this form but don't expect much!">
</FORM>
<HR>
\end{rawhtml}
```

The result is shown above.

Warning: Avoid using LaTeX commands involving counters (e.g. numbered figures or equations) in conditional text, because this may disrupt the values of the counters in the electronic version.

4.3 Conditional Text

Conditional text can be specified using the environments latexonly and htmlonly. These allow writing parts of a document which are intended only for electronic delivery or only for paper-based delivery.

This would be useful for example in adding a long description of a multi-media resource in the paper version of a document. Such a description would be redundant in the electronic version, as the user can have direct access to this resource.

Here is an example of the use of the latexonly environment:

```
\begin{latexonly}
\begin{figure}
    \begin{center}
    \fbox{\includegraphics[width=4in]{eform.ps}}
    \end{center}
    \caption{An electronic form. Of course in the online version of this
     document the form above would be active.}
\end{figure}
\end{latexonly}
```

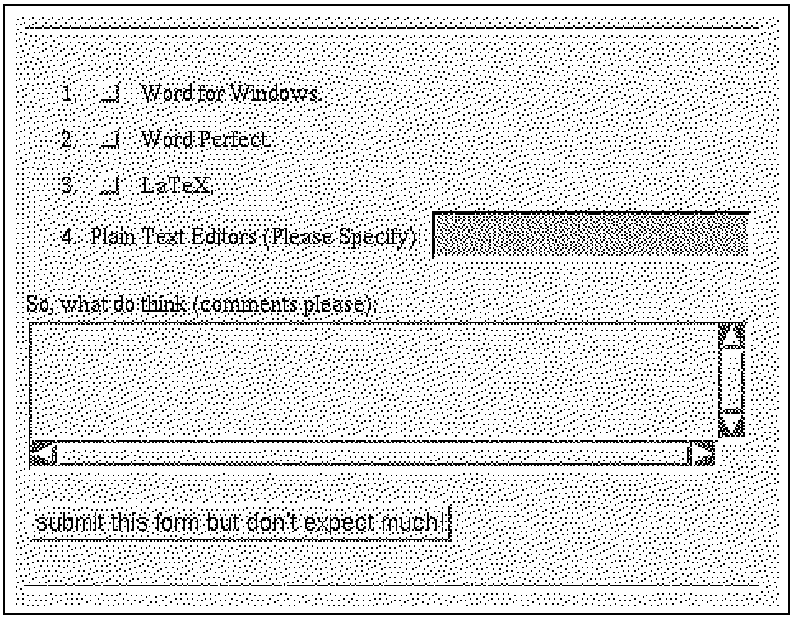

Figure 1: An electronic form. Of course in the online version of this document the form above would be active.

There are also shorthand notations to accomplish the same thing with less typing. The `\latex{..}` command causes everything within the braces to be processed by LaTeX, but ignored by LaTeX2HTML. Conversely, the `\html{..}` command causes everything within the braces to be ignored by LaTeX and processed by LaTeX2HTML. Finally the command `\latexhtml{..}{..}` causes everything within the first set of braces to be processed exclusively by LaTeX, with the contents of the second set of braces processed solely by LaTeX2HTML.

Another variant of the latexonly environment is available, in which everything between `%begin{latexonly}` and `%end{latexonly}` is ignored by LaTeX2HTML. The difference is that the latexonly environment puts the contents into a group, in which all definitions are local. There is no such scoping with the `%begin... %end...` variant.

The following example should clarify what happens:

```
\newcommand{\A}{The letter A.}
\newcommand{\B}{The letter B.}

\begin{latexonly}
\renewcommand{\A}{Not the letter A.}
\end{latexonly}
%begin{latexonly}
\renewcommand{\B}{Not the letter B.}
%end{latexonly}

\begin{document}
\A \B
\end{document}
```

If you process this with LaTeX, the result is: The letter A. Not the letter B.

A further device, provided by `html.sty`, is the `\htmlrule` command. This puts a horizontal rule into the HTML file only; being ignored in the `.dvi` version. It is useful to provide extra visual separation between paragraphs, without creating a new HTML page, such as might warrant extra vertical space within the printed version.

4.4 Symbolic References shown as Hyperized Text

In printed documents cross-references are shown through a *numeric or symbolic indirection* e.g. "see Figure 1" (numeric indirection), or "see section 'Changes'" (symbolic indirection). LaTeX2HTML can mirror this mechanism using the same numeric or symbolic references, or when these are not appropriate by using iconic references.

In a hypertext document however, cross-references can be shown without any indirection, just by highlighting a relevant piece of text. This can make a document more readable as it removes unnecessary information.

A single new LaTeX command `\hyperref` can be used for specifying how a cross-reference should appear, both in the printed document and in the hypertext version. For example, assuming that the label `sec:cond` is defined somewhere within a document, the command `\hyperref`, which takes 4 arguments, can be used in the same document as follows:

```
\emph{Is the concept of
\hyperref
               % This will be highlighted in the hypertext version
{conditional text} % argument #1
               % This will be shown in the printed version
               % followed by a numeric reference ...
{conditional text (see Section }   % argument #2
               % ... followed by this text
{ for more information)}  % argument #3
               % This the common label
{sec:cond} % argument #4
a good idea? }
```

Here is how it will be shown:

> Is the concept of conditional text (see Section 4.4 for more information) a good idea?

In the hypertext version what would appear is:

> Is the concept of conditional text a good idea?

(Of course conditional text would be an active hypertext link.)

Another command also defined in `html.sty` is `\htmlref` which has the same effect as `\hyperref` during the conversion to HTML. It takes two arguments, some text and a label. In the HTML version the text will be "hyperized", pointing to the label. In the paper version the text will be shown as it is and the label will be ignored; e.g.

```
With \verb|\htmlref| \htmlref{it's easy to make links}{fig:example}.
```

In the HTML version it will be shown as:

> With `\htmlref` it's easy to make links.

4.5 Symbolic References between Living Documents

The method of the previous section to generated symbolic hyperized links can easily be extended to *external* documents processed by LaTeX2HTML. When LaTeX2HTML processes a document, it generates a *Perl* file named `<prefix>labels.pl` which contains a list of all the symbolic labels that were defined, along with their locations. The `<prefix>` is empty unless otherwise specified, to allow different document segments to share the same directory. Links to an external document are then possible once a connection is established to that document's `labels.pl` file. This connection is established by the `\externallabels` command:

```
\externallabels{<URL to directory of external document>}%
               {<local copy of external document labels.pl file>}
```

The first argument to `\externallabels` should be a URL to the directory containing the external document. The second argument should be the full path-name to the `labels.pl` file belonging to the external document. Note that for *remote* external documents it is necessary to copy the `labels.pl` file locally so that it can be read when processing a local document that uses it. The command `\externallabels` can be used once for each external document in order to import the *external labels* into the current document. A warning is given if `labels.pl` cannot be found.

If a symbolic reference made in either of the commands described in Section 4.4 is not defined within the document itself, LaTeX2HTML will look for that reference in one of the external files[17]. After any modifications in an external document (sections added/deleted, segmentation into different physical parts, etc.) a new `labels.pl` will be generated. If the `\externallabels` command in another document contains the correct address to an updated copy of the `labels.pl` file, then the cross-references will be re-aligned after running the local document through the translator.

There is also a mechanism analogous to the *label–ref* pairs of LaTeX, which can be used only within a single document. These labels are called *internal labels*, as opposed to the external labels defined above. They are used extensively with the document segmentation strategy described later.

Either type of label is defined with a LaTeX `\label` command. Labels can be referenced *within* a document using a `\ref` command. When processed by LaTeX, each `\ref` command is replaced by the section number in which the corresponding `\label` occurred. When processed by the translator, each `\ref` is replaced by a hypertext link to the place where the corresponding `\label` occurred. This mechanism can be extended to external documents by the `\externalref{}` command:

```
\externalref{<symbolic label in remote document>}
```

The argument to `\externalref` may be any symbolic label defined in the `labels.pl` file of any of the external documents. Such references to external symbolic labels are then translated into hyper-links pointing to the external document.

4.5.1 Cross-Referencing Example

To understand this mechanism better consider how you would maintain a link to this section (of the hypertext version of this document) from one of your documents, without using

[17] Care must be taken to ensure that critical symbolic references are unique across related documents.

labels. Sure enough you can get the name of the physical file that this section is in. This however is quite likely to change, and any links to it would become invalid. To update your link, the name of the new file must be found and your link changed by hand. Also there is no general updating mechanism, so the only way to find out if your document is pointing to the right place is by actually following the link, then doing a manual update[18].

Next consider how it could be done with symbolic labels. First you have to import the labels used in this document by copying the file `labels.pl`, saving it in `/tmp/labels.pl` say, then adding anywhere in your document:

```
\externallabels{http://cbl.leeds.ac.uk/nikos/tex2html/doc/manual}%
               {/tmp/labels.pl}
```

After that you can use the label `crossrefs` defined at the beginning of this section[19] as follows:

```
\externalref{crossrefs}
```

This will be translated into the appropriate hyper-link to this page. If there are any changes in this document and you would like to bring your document up-to date, you have to copy `labels.pl` again and rerun the translator on your document. Of course if I move the directory containing the `HTML` files for this document somewhere else, then you would have to make a change in the argument of the `\externallabels` command to reflect this.

It is obvious that some level of collaboration is required between authors trying to maintain cross-references between different documents. Using symbolic labels makes this a lot easier (especially for documents written by the same author).

4.6 Document Segmentation[20]

One of the greatest appeals of the World-Wide Web is its high connectivity through hyper-links. 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 LaTeX2`HTML` 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 LaTeX2`HTML`. 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 book-keeping formerly performed exclusively by LaTeX2`HTML`. The following four LaTeX extensions carry out segmentation:

[18] Link validation can be done automatically but the updating must be done manually when filenames have changed (assuming no other symbolic label mechanism is available).

[19] You either have to guess the role of each label by looking at the `labels.pl` file or by asking the author!

[20] This feature is supported only for users of LaTeX 2_ε.

\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 \segment* of this command, 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[<*type*>]{<*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.

Note: If extensive indexing is to be used, then it is advisable to keep each <*prefix*> quite short. This is because the hyper-links in the index have text strings constructed from this <*prefix*>, when using the makeidx package. Having long names with multiply-indexed items results in an extremely inelegant, cumbersome index. See Section 5.7.2 for more details.

\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.

4.6.1 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 96.2 distribution of LaTeX2HTML, with more prolific comments than are shown here.

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

```
\documentclass{article}          % Must use LaTeX 2e
\usepackage{html,makeidx,color}

\internal[figure]{s1}            % Include internal information
\internal[figure]{s2}            % from children
\internal[sections]{s1}
\internal[sections]{s2}
\internal[contents]{s1}
\internal[contents]{s2}
\internal[index]{s1}
\internal[index]{s2}

\begin{document}                 % The start of the 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 to typeset 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 inter-segment 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 LaTeX2`HTML` is not run unnecessarily. It is also usual for the information page to be suppressed by specifying `-info 0` on all but the top-level program segment.

In the above example, all segments are built within the same sub-directory `report/` of the directory containing the LaTeX source files. This is achieved simply by using the option

`-dir report` with each. All the GIF images and `<prefix><type>.pl` files are created and stored within this directory.

Sometimes it is desirable to build one or more segments within their own sub-directory. This is especially so when a segment has a large number of GIFs, or if it is required to be part of more than one combined document. In this case the `-dir <dir>` options can be different or omitted entirely. For inter-segment referencing to work, a "relative path" must be included as part of the `<prefix>` with each `\internal` command; e.g.

```
\internal[figure]{../sect1/s1}
```

4.7 Hypertext Links in Bibliographic References (Citations)

If a report or a book that is cited (using the `\cite`) command is available (or there is information about it) on the World-Wide Web then it is possible to add the appropriate hypertext links in your bibliographic database (the `.bib`) file.

Here is an example of a bibliographic entry for the original LaTeX [1] blue book:

```
@string{curiaURL="\htmladdnormallink
{http://curia.ucc.ie/info/TeX/menu.html}
{http://curia.ucc.ie/info/TeX/menu.html}"}

@string{fernURL="\htmladdnormallink
{http://es-sun2.fernuni-hagen.de/info2html?(latex.info)Top}
{http://es-sun2.fernuni-hagen.de/info2html?(latex.info)Top}"}

@book{lamp:latex,
title = "LaTeX User's Guide \& Reference Manual, 2nd edition",
year = 1994 ,
author = "Leslie Lamport",
Publisher = "Addison--Wesley Publishing Company, Inc.",
note = "Online information on {\TeX} and {\LaTeX} is available at "
 # curiaURL # " and " # fernURL }
```

See the bibliography for how this will appear. No other modifications are required—LaTeX and BibTeX should work as normal. Note that it would be sensible to put the `@string` commands into a separate file, `urls.bib` say, loaded with the main file via `\bibliography{urls,...}`.

For those who use the Harvard style for references there exists a special conversion add-on package[21].

4.8 Customizing the Navigation Panels

The navigation panels are the strips 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 in Section 3.1.

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,

[21] http://www.arch.su.edu.au/~peterw/latex/harvard/

```
\htmladdtonavigation{\htmladdnormallink
   {\htmladdimg{http://server/mybutton.gif}}{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.

Below is an example of a navigation panel (the "`.`" is the *Perl* string concatenation operator and "`#`" signifies a comment).

```
sub top_navigation_panel {
    #  Start with a horizontal rule (3-d dividing line)
       "<HR> ".
    # Now add a 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"
}
```

Note that extra buttons may be included by defining suitable code for the subroutine `$CUSTOM_BUTTONS`.

5 Special Features

This section describes some special important considerations to keep in mind when using LaTeX2HTML.

5.1 Variation on HTML Versions

The Hypertext Mark-up Language (`HTML`) 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 as follows:

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 (Unicode). This is a 16-bit character set and can thus display a much larger set of characters. There are also provisions for bidirectional languages (e.g. in Arabic the text is written from right to left, but numerals from left to right), and provisions in `HTML` to determine the character set and the language used. Not all of the symbols are available in TeX, LaTeX2HTML, or any browser yet available. However the version 2.1 emulation of LaTeX2HTML is at least a start.

Version 2.2 This version supports all the features of version 2.1, plus the `HTML3` Table Model. This feature is already available in many `HTML` browsers, including *Netscape Navigator* V1.2.

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 In addition to all of the features of version 3.0, this version adds support for math mark-up. Currently the only browser which can display this mark-up is *Arena*.

5.2 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 `\today` is presented in a format 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 replace the navigation buttons which come with LaTeX2HTML (by default in English) with your own. As long as the new buttons have the same file-names as the old ones, there should not be a problem.

5.3 Image Conversion

LaTeX2HTML converts equations, special accents, external PostScript files, and LaTeX environments it cannot directly translate into inlined images. This section describes how it is possible to control the final appearance of such images. For purposes of discussion ...
"small images" refers to equations, special accents and any other image generating LaTeX commands; while ...
"figures" applies to image-generating LaTeX environments (e.g. minipage, figure, table, 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 variables are initially set to have value 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.
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 section 5.6 for more information.)
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 counterclockwise 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 centered vertically. The center option specifies that it should also be centered horizontally. This option is valid only if the HTML version is 3.0 or higher, or if the $NETSCAPE_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.

5.3.1 An Embedded Image Example

The effect of the LaTeX commands below can be seen in the thumbnail sketch of Figure 2.

```
\begin{figure}
    \htmlimage{thumbnail=0.5}
    \centering\includegraphics[width=5in]{figure.ps}
    \latex{\addtocounter{footnote}{-1}}
    \caption{A sample figure showing part of a page generated by
       \latextohtml{} containing a customized navigation panel
       (from the \protect\htmladdnormallink
{CSEP project\latex{\protect\footnotemark}}%
{http://csep1.phy.ornl.gov/csep.html}).}
       \label{fig:example}
\end{figure}
\latex{\footnotetext{http://csep1.phy.ornl.gov/csep.html}}
```

The \htmlimage command is also often useful to cancel-out the effect of the configuration variable $FIGURE_SCALE_FACTOR. For example to avoid resizing a color screen snap despite the value of $FIGURE_SCALE_FACTOR it is possible to use \htmlimage{scale=0}.

5.4 Figures, Tables and Arbitrary Environments

These are here to show how the translator handles figures, tables and other environments. Compare the paper with the online version.

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emur		33.33
armadillo	frozen	8.99

Table 2: A sample table taken from [1]

Here are some some automatically numbered right-justified equations ...

$$\Phi_{l+1,m,n} = \left(\Phi + h\frac{\partial\Phi}{\partial x} + \frac{1}{2}h^2\frac{\partial^2\Phi}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3\Phi}{\partial x^3} + \ldots\right)_{l,m,n} \tag{1}$$

... with some gratuitously áccënted text in-between.

$$\begin{aligned} \frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2} \\ + \frac{\Phi_{l,m,n+1} - 2\Phi_{l,m,n} + \Phi_{l,m,n-1}}{h^2} = -I_{l,m,n}(v) \end{aligned} \tag{2}$$

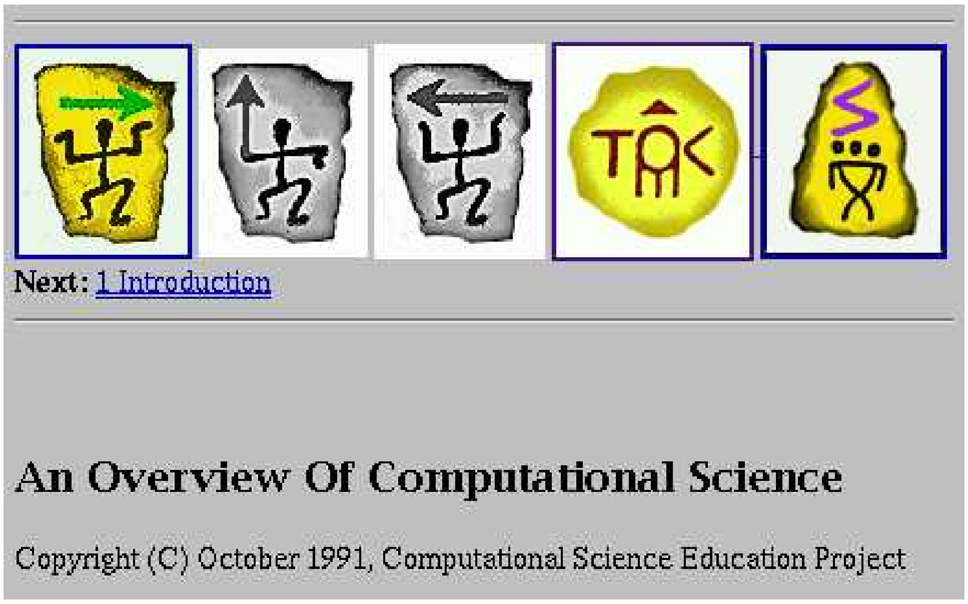

Figure 2: A sample figure showing part of a page generated by LaTeX2HTML containing a customized navigation panel (from the CSEP project[22]).

[22] http://csep1.phy.ornl.gov/csep.html

5.5 Image Sharing and Recycling

It is not hard too see how reasonably sized papers, especially scientific articles, can require the use of many hundreds of external images. For this reason, image sharing and recycling is of critical importance. In this context, "sharing" refers to the use of one image in more than one place in an article. "Recycling" refers to the use of an image left over from a previous run of LaTeX2HTML. Without them, every instance of an image would have to be regenerated every time even the slightest change in the document were made.

All types of images can be shared. These include "small images" and figures with or without thumbnails and image-maps. Furthermore, most images can also be reused. The only exception are those 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.

If you have a document with a great many displayed equations, you might try using the `heqn` style package. Inclusion of the `heqn.sty` file has absolutely no effect on the printed version of the article, but it does change the way in which LaTeX2HTML translates displayed equations and equation arrays. It causes the equation numbers of the `equation` environment to be moved outside of the images themselves, so that they become order-independent and hence recyclable. Images that result from the `eqnarray` environment are also recyclable, so long as their equation numbers remain unchanged from the previous run. An optional `\nonumber` command is recognized in each line of the equation array. A side-effect of this approach is that equation numbers will appear on the left side of the page. The `heqn` package requires the `html` package.

5.6 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 a `HTML` *map-file*, which associates 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 3.0 also allows it to reside on the client side for faster response. Both configurations are supported by LaTeX2HTML 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 a document has two figures designated to become active image-maps. The first figure includes a statement like:

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

The second figure has a line like:

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

A typical user-map file, named `report.map`, might contain the following information[23]:

```
#
#  Define the location(s) of the labels.pl file(s):
#
+report/ <URL>
#
#  Define map #1:
#
BlockDiagram.map:
label1  rect    288,145 397,189
label2  rect 307,225 377,252
label2  default
#
#  Define map #2
#
FlowChart.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` .

5.7 Other Style-Files

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

[23] 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.

`\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}
```

This will produce:

Item 1: This will have a white ball.

Item 2: This will also have a white ball.

Item 3: This will have a red ball.

One can also obtain LaTeX 2_{ε} style-files `floatfig.sty` and `wrapfig.sty`, which provide support for the floatingfigure and wrapfigure environments, respectively. These environments allow text to wrap around a figure in the printed version, but are treated exactly as an ordinary figures in the electronic version. They are described in Goossens, et. al. [2].

5.7.1 Indicating Differences between Document Versions

LaTeX2HTML supports the LaTeX 2_{ε} `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. This is a very primitive form of version control and there is much scope for improvement.

Within the LaTeX version of this manual two thicknesses of change-bar have been used. Thinner bars indicate changes introduced with version V96.1, while thicker bars indicate later additions. Within the HTML version the change-bars clearly indicate the different revisions with explicit numbering.

5.7.2 Indexing

LaTeX2HTML automatically produces an Index consisting of the arguments to all `\index` commands encountered, if there are any. A hyperlink is created to that point in the text where the `\index` command occurred.

More sophisticated indexing is available by loading the makeidx package. Most of the features described in [1, Appendix A] become available. This includes:

styled entries, using '@' : Entries of the form `\index{<sort-key>@<styled-text>}` produce `<styled-text>` as the entry, but sorted according to `<sort-key>`.

hierarchical entries, using '!' : Entries of the form `\index{<item>@<sub-item>}` set the `<sub-item>` indented below the `<item>`. Unlimited levels of hierarchy are possible, even though LaTeX is limited to only 3 levels. The `<sort-key>@<styled-text>` can be used at each level.

explicit ranges, using '`|(`' and '`|)`' : This is perhaps more useful in the LaTeX version. In the `HTML` version these simply insert words "from" and "to", respectively, prior to the hyperlink to where the index-entry occurs.

`|see{`*`<index-entry>`*`}` **:** provides a textual reference to another indexed word or phrase, by inserting the word "see". This can be used in conjunction with `\htmlref` to create a hyperlink to the *`<index-entry>`*; viz.

```
\index{latexe@\LaTeXe |see{\htmlref{\LaTeX}{IIIlatex}}}
```

where a `\label` has been specified in some other index-entry, as follows:

```
\index{latex@\LaTeX\label{IIIlatex}}
```

`|emph` **:** is recognised but *ignored*; other `|`*`<command>`* commands are *not* processed by LaTeX2`HTML`, with the following exception...

blank lines and alphabetization: Having precisely a single space-character after the `|` (e.g. `\index{A| }`) places a blank line before the index entry and omits the hyperlink. This is used mainly for visual formatting; it allows a break before the entries starting with each letter, say. Using a printable-key, as in `\index{Q@Q, R| }`, is appropriate when there are no indexed words starting with 'Q', say.

quoted delimiters: The three special delimiters can be used within the printable portion, if preceded by the double-quote character: `"@`, `"|`, `"!` and also `""` for the quote character itself. Also `\"` produces an umlaut accent on the following character, when appropriate, else is ignored.

Furthermore, the printable part of an index entry can contain `HTML` anchors; that is, hyperlinks and/or `\label{...}`s. This allows index entries to contain cross-links to other entries, for example, as well as allowing index-entries to be the target of hyperlinks from elsewhere within the document.

The next subsection describes how this feature is used within this manual to create a Glossary, containing a short description of all file-names, configuration-variables and application software mentioned within the manual, integrated with the Index. All occurrences of the technical names can be easily found, starting from any other.

When a single item is indexed many times, it is sufficient to have a `\label` command appearing within the printable portion of the first instance of an `\index{...}` command for that item, within a single document segment.

If the index-entries are in different segments of a segmented document, it is sufficient to have the `\index{...@...\label{...}}` appearing within that segment, in which the item is indexed, whose indexing information is loaded earliest via a `\internal[index]{...}` command. When in doubt, include one `\index{...@...\label{...}}` per segment in which the item is indexed.

For cross-links to work effectively within segmented documents, the indexing command `\index{...@...\label{...}}` *must* occur earlier in the same segment than any use of `\index{...@...\htmlref{...}{...}}` intended to create a link to that label. If the `\label` occurs in a different segment, then a `\internal[index]{...}` command for that segment, may be needed at the beginning of the segment with the `\htmlref`. When this is done incorrectly, the resulting link will be to the segment where the indexed item occurred, rather than staying within the Index.

Since use of section-names, as the text for hyperlinks, can lead to a very long and cumbersome Index, especially when single items have been indexed many times, a further feature is provided to obtain a more compact Index.

Use of the command-line option `-short_index` causes a codified representation of the sectioning to be used, rather than the full section-name. The differences are as follows.

- For example, `2.1` means sub-node #1 of node #2, representing the entire document via a tree-like structure.
- The top-most node is simply denoted `^`.
- With a segmented document, each segment is codified separately using the <*prefix*> supplied for that segment. The Index includes a legend of these prefixes, each giving the title of the leading page from the segment, as a hyperlink to the place on that page where its child-links are displayed.
- Hyperlinks start on the same line as the index-key, rather than the next line, separated by '`|`'. This gives further compactification for easier browsing.
- If `-prefix` <*prefix*> has been specified, then the <*prefix*> is prepended to the codified form. This is most useful for segmented documents. Now the top-most node is indicated by the bare <*prefix*>.

These features can also be obtained by setting the variable `$SHORT_INDEX` to have value `1`, in a configuration or initialization file; provided, of course, that the document loads the makeidx package.

5.7.3 Integrated Glossary and Index

A large number of different pieces of software are required to make LaTeX2HTML work effectively, as well as many files containing data or code to work with parts of this software. For this reason, a Glossary is included with this manual. It contains the names of all files, configuration variables, application software and related technical terms, with a short description of what it is, or does, and perhaps a URL for further reference.

In the printed version each item in the Glossary is accompanied by the page-numbers on which the item is mentioned, somewhat like in the Index. For the HTML version, each glossary-item contains a hyperlink to an index-entry, which then has links to each occurrence. These extra index-entries do not appear in the printed version; indeed they also contain a hyperlink back to the corresponding glossary-entry.

This feature is currently available only when using the makeidx package and needs also the html and htmllist packages. It was developed for version 96.2 by Ross Moore `<ross@mpce.mq.edu.au>`, incorporating an extensive revision of `makeidx.perl`, as well as additions to LaTeX2HTML so that all aspects of indexing work correctly with segmented documents.

Since LaTeX provides no guidelines for how a Glossary should be constructed, the technique used here will be explained in detail, for both the printed and HTML versions.

- Firstly the `\makeglossary` command, which is similar to `\makeindex`, must appear in the document preamble, so that LaTeX will record uses of the `\glossary{...}` command within a file `manual.glo`.

  This command is redundant in the HTML version, so is given a trivial definition which is ignored by LaTeX.

- Next, the words, phrases or technical terms to be included in the Glossary are marked in the main text using the `\glossary` command, used indirectly via other macros. For example, file-names are inserted via `\fn{html.sty}`, `\fn{dvips}`, `\appl{dvips}` etc. which both insert the text and create the glossary-entry; *viz.*

```
\newcommand{\fn}[1]{\htmlref{\texttt{#1}}{GGG#1}\glossary{#1}}
\newcommand{\appl}[1]{\htmlref{\textsl{#1}}{GGG#1}%
  \Glossary{#1}{\textsl{#1}}}
```

- The expansions of `\glossary`, and the slightly more general `\Glossary`, are different for the printed and `HTML` versions. For the `HTML` version the following definitions occur within an htmlonly environment:

```
\def\glossary#1{\index{#1@\texttt{#1} \label{III#1}%
  \htmlref{(G)}{GGG#1}}}
\def\Glossary#1#2{\index{#1@{#2} \label{III#1}\htmlref{(G)}{GGG#1}}}
\def\makeglossary{}
```

  ...while in LaTeX we need only: `\newcommand\Glossary[2]{\glossary{#1@#2}}` .

  Notice how the feature of makeidx, allowing the printable portion to be separate from the sorting-key, is used to allow text-styles to be included within both index-entries and glossary-entries. Indeed the purpose of `\Glossary` is to allow deviations from a fixed style, e.g.

```
\newcommand{\MF}{\htmlref{\textsl{Metafont}}{GGGmetafont}%
  \Glossary{metafont}{\textsl{Metafont}}}%
```

  Also notice that in the `HTML` version an index-entry is created that includes, within its printable portion, both a `\label` and a hyperlink. The former, having name `III...`, will ultimately reside on the Index page, while the latter will point to an anchor named `GGG...` on the Glossary page. These names must be distinct from any other names used with `\label`s elsewhere in the document, hence the use of prefixes `III` and `GGG`. A short string '(G)' is used for the text of the hyperlink in the Index.

- The text descriptions of the glossary-items are stored in a file called `l2hfiles.dat`, with one description per line. For the `HTML` version this file is actually read as input:

```
\begin{htmlonly}
\section{Glossary of variables and file-names\label{Glossary}}
\begin{htmllist}\htmlitemmark{OrangeBall}
\input l2hfiles.dat
\end{htmllist}
\end{htmlonly}
```

  For this reason alone it is desirable to have `l2hfiles.dat` sorted alphabetically.

- The mechanism used for the LaTeX version also requires the file to be sorted strictly alphabetically, according to the sort-keys associated to each glossary entry.
  (This requirement could be relaxed, but only with a loss in efficiency; see below.)

  LaTeX constructs its Glossary by running the `makeindex` utility on the file `manual.glo`, using the following command:

```
makeindex -o manual.gls -s l2hglo.ist manual.glo
```

Its output, which includes page numbering for an index, is stored in `manual.gls` and subsequently read by LaTeX using:

```
\begin{latexonly}
\InputIfFileExists{manual.gls}{\clearpage\typeout{^^Jcreating Glossary...}}
{\typeout{^^JNo Glossary, since  manual.gls  could not be found.^^J}}
\end{latexonly}
```

The configuration file `l2hglo.ist` is included along with this manual. It contains a portion that inserts tricky TeX code at the beginning of `manual.gls`. This code extracts from `l2hfiles.dat` that line corresponding to each glossary entry, then typesets it itemized within an environment called theglossary.

```
\newenvironment{theglossary}{\begin{list}{}{%
  \setlength{\labelwidth}{20pt}%
  \setlength{\leftmargin}{\labelwidth}%
  \setlength\itemindent{-\labelwidth}%
  \setlength\itemsep{0pt}\setlength\parsep{0pt}%
  \rmfamily}}{\end{list}}
```

Currently searching within `l2hfiles.dat` is only done sequentially, stopping at the end of the file. If an entry is not found then it is skipped and a message printed to the log; the next entry will search from the top of the file. If all entries are included and maintained in strict order, there will be no skipping and each line of `l2hfiles.dat` is read exactly once.

- Within `l2hfiles.dat` the data lines look like:

```
\item[\gn{french.perl}] adds \Perl\ code to be compatible with the ...
\item[\gn{\textsl  {ftp}}] ‘File Transfer Protocols’, network ...
\item[\gn{german.perl}] adds \Perl\ code to be compatible with the ...
...
```

For the LaTeX version the `\item[\gn{...}]` is only used for pattern-matching, to find the correct data entry. All typesetting is controlled from within `manual.gls`.

However the `HTML` version requires the following definition:

```
\newcommand{\gn}[1]{\texttt{#1}\label{GGG#1}\htmlref{\^}{III#1}}%
```

which establishes the hyperlink to the Index, marked by ‘^’, and provides the `\label` to create the target in the Glossary for any `\glossary{...}` command having the corresponding argument.

6 Known Problems

Here are some of the problems of the current version:

Correctness and Efficiency: The translator cannot be guaranteed to perform as expected. Several aspects of the implementation need optimization and improvement. Apart from possible bugs the translator may place heavy demands on your resources.

Unrecognized Commands and Environments: Unrecognized commands are ignored and any arguments are left in the text. Unrecognized environments are passed to LaTeX and the result is included in the document as one or more inlined images.

Cross-references: References in environments that are passed to LaTeX for processing (e.g. a `\cite`, or a `\ref` command), are not processed correctly. `\label` commands are handled correctly.

Order-Sensitive Commands: Commands which affect global parameters during the translation, and are sensitive to the order in which they are processed may not be handled correctly. In particular, counter manipulation (e.g. `\newcounter`, `\setcounter`, `\stepcounter`, etc.) commands may cause problems.

Index: The translator generates its own index by saving the arguments of the `\index` command. The contents of the `\theindex` environment are ignored.

New Definitions: New definitions (`\newcommand`, `\newenvironment`, `\newtheorem` and `\def`), will not work as expected if they are defined more than once. Only the last definition will be used throughout the document.

Scope of declarations and environments: If the scope of a declaration or environment crosses section boundaries, then the output may not be as expected, because each section is processed independently.

Math-mode font-size changes: Math-mode font changes made outside the math-mode are not honoured. Thus the two equations in `$a_b$` and `{\LARGE $a_b$}` would come out looking the same. The trick is to write `$a_b` and `$\mbox{\LARGE $a_b$}$`.

6.1 Troubleshooting

Here are some curable symptoms:

Cannot run any of the *Perl* programs: If your *Perl* installation is such that *Perl* programs are not allowed to run as shell scripts you may be unable to run `latex2html`, `texexpand` `pstogif` and `install-test`. In this case change the first line in each of these programs from

```
#!/usr/local/bin/perl
```

to

```
: # *-*-perl-*-*
    eval 'exec perl -S  $0 "$@"'
    if $running_under_some_shell;
```

The `install-test` script gives uninformative error messages: If, for any reason, you have trouble running `install-test`, do not despair. Most of what it does is to do with checking your installation rather than actually installing anything. To do a **manual installation** just change the variable `$LATEX2HTMLDIR` in the beginning of the file `latex2html` to point to the directory where the LaTeX2HTML files can be found.

Also, make sure that the files `pstogif`, `texexpand` and `latex2html` are executable, and if necessary use the Unix `chmod` command to make them executable.

It just stops. Check the style files that you are using. It is likely that you are using a style file which contains raw TeX commands. In such a case start LaTeX2HTML with the option `-dont_include` <*style-file name*>. Alternatively, add the name of the style to the variable `$DONT_INCLUDE` in your `$HOME/.latex2html-init` file. If you don't have such a file then create one and add the lines:

```
$DONT_INCLUDE = "$DONT_INCLUDE" . ":<style file name>";
1;  # This must be the last line
```

Another reason why LaTeX2HTML might stop is that the LaTeX source file itself contains raw TeX commands. In this case you may put such commands inside a latexonly (see Section 4.3) environment.

***Perl* cannot parse the `latex2html` script:** Update your *Perl* to patch level 36. You can check which version of *Perl* you are using by invoking *Perl* with the `-v` option. Earlier versions of *Perl* than that shown above have caused problems due to tighter control over syntax.

It crashes (dumps core) as soon as it starts : Update your *Perl* 4 to patch level 36 or later (*Perl* 5).

You can check which version of *Perl* you are using by invoking *Perl* with the `-v` option.

While you wait for your technical support people to upgrade *Perl* you could try invoking *Perl* from within LaTeX2HTML with the `-d` (debug) option. Then, when LaTeX2HTML starts, it will immediately fall into the *Perl* debugger. To continue just press `c<CR>`.

`dvips` complains about incorrect arguments: Please use a version which supports the command-line options `-M -S`, `-o` and `-i`. "Recent" versions, at least after 5.516, do support them.

It gives an "`Out of memory`" message and dies: If you are using version LaTeX2HTML 0.7 or later try splitting your source file into more than one file, using the LaTeX commands `\input` or `\include`. Also, use the `-no_images` option.

As a last resort you may consider increasing the virtual memory (swap space) of your machine. As an indication of what you might be able to do on your machine, a very long book (about 1000 printed pages) required about 24MB of RAM and over 150MB of swap space to convert on a local Sun Sparc ELC running SunOS 4.1.3.

It gives "dbm" related error messages: LaTeX2HTML 0.7 and later requires the Unix DataBase Management system (*DBM* or *NDBM*) in order to run. This is usually part of each Unix operating system but if you don't have it then you may need to get it. Use *Archie*[24] to find one.

[24] http://www.pvv.unit.no/archie/

The `\verb"ABC"` command doesn't work: This is a nasty bug. Please use any character other than quotes; e.g. `verb+ABC+`.

Cannot get the "tilde" (~) to show: The trick here is to use the command `\~{}`. Alternatively it is possible to use something like:

```
\htmladdnormallink{mylink}%
  {\begin{rawhtml}http://host/~me/path/file.html\end{rawhtml}}
```

or `\htmladdnormallink{mylink}{http://host/\%7Eme/path/file.html}`.

Warning: Some browsers may not be able to interpret the `%7E` as a "tilde" character.

Macro definitions don't work correctly: As mentioned in other places, not all plain TeX `\def`initions can be converted. But you may also have problems even when using LaTeX definitions (with `\newcommand` and `\newenvironment`) if such definitions make use of sectioning or `\verbatim` commands. These are handled in a special way by LaTeX2HTML and cannot be used in macro definitions.

In general the macro handling mechanism is inefficient and very fragile. Avoid using macros if possible.

`\input` commands: There is a bug in the expansion of `\input` commands which causes a problem when more than one `\input` command appears on the same line. There is no quick fix other than suggesting that you put each `\input` command on a line by itself, in the LaTeX source files.

`\input` commands in verbatim environments: These cause problems; there's no fix yet.

Optional arguments in description environments: If you have optional arguments for the `\item` command in a description environment containing nested "`]`" characters then these may not show up correctly. To avoid the problem enclose them in `{}`s; e.g. `\item[{[nested [angle [brackets] are ok]]}]`

LaTeX2HTML behaves differently even when you run it on the same file: If you notice any strange side-effects from previous runs of LaTeX2HTML, try using the option `-no_reuse` and choose `(d)` when prompted. This will clear any intermediate files generated during previous runs. Note that this option will disable the image-reuse mechanism.

Cannot convert PostScript images which are included in the LaTeX file: It is likely that the macros you are using for including PostScript files (e.g. `\epsffile`) are not understood by LaTeX2HTML. To avoid this problem enclose them in an environment which will be passed to LaTeX anyway; e.g.

```
\begin{figure}
\epsffile{<PostScript file name>}
\end{figure}
```

Another reason why this might happen is that your shell environment variable `TEXINPUTS` is undefined. This is not always fatal but if you have problems you can use full pathnames for included PostScript files (even when the PostScript files are in the same directory as the LaTeX source file). Alternatively try setting `TEXINPUTS` to '`.::`'. With some TeX and LaTeX installations setting `TEXINPUTS` to '`.::`' may cause problems in the normal operation of LaTeX. If you get errors such as LaTeX complaining that it can no longer find any style files then you must set `TEXINPUTS` to `"<path to your LaTeX installation>:."` if you want to use both LaTeX and LaTeX2HTML.

Some of the inlined images are in the wrong places: This happens when any one of the inlined images is more than a page (paper page) long. This is sometimes the case with very large tables or large PostScript images. In this case you can try specifying a larger paper size (e.g. '`a3`', '`a2`' or even '`a0`') instead of the default ('`a4`') using the LaTeX2HTML variable `$PAPERSIZE` in the file `latex2html.config`.

Another reason why this may happen is that by default the `dvips` program reverses the PostScript pages it generates. If your `dvips` program behaves in this way try changing the line: `$DVIPS = "dvips";` to: `$DVIPS = "dvips -r0";` in the file `latex2html.config`.

Unacceptable quality of converted images: Try changing the size of the image (see Section 5.3).

The bibliographic references are missing: Run `latex` and then `bibtex` on the original source file in order to generate a `.bbl` file. LaTeX2HTML may need the `.bbl` file in order to generate the references.

The labels of figures, tables or equations are wrong: This can happen if you have used any figures, tables, equations or any counters inside conditional text; i.e. in a latexonly or a htmlonly environment.

Problems after changing the configuration files: Please make sure that the last line in the configuration files (i.e. `.latex2html-init` and `latex2html.config`) is:

```
1; # This is the last line
```

This is a *Perl* quirk.

Problems when producing the `.dvi` version: If you are using any of the new LaTeX commands which are defined in the `html.sty` file make sure that `html.sty` is included; e.g. as one of the optional arguments to the `\documentclass` command.

Of course you also have to make sure that LaTeX knows where the `html.sty` file is, either by putting it in the same place as the other style-files on your system, or by changing your `TEXINPUTS` shell environment variable[25].

Some of the fonts are translated incorrectly: There is a fault in way the LaTeX scoping rules have been interpreted in LaTeX2HTML. Consider this:

```
\ttfamily fixed-width font.
\begin{something}
nothing here
\end{something}
default font.
```

[25] If don't know how to do either of these things, copy (or link) `html.sty` to the directory of your LaTeX document.

When processed by LaTeX, the effect of the `\tt` command is delimited by the beginning of the environment "something" so that "default font" will appear in the default font. But LaTeX2HTML will not recognise "something" as a delimiter and "default font" will appear in the wrong font.

To avoid this problem until it is fixed you may delimit the scope of some commands explicitly using `{}`'s; i.e.

```
\texttt{fixed-width font}.
\begin{something}
nothing here
\end{something}
default font.
```

Using *Ghostscript 3.x* it no longer generates inlined images for equations: If you have a version of LaTeX2HTML later than 0.6.1, go to the LaTeX2HTML directory and run `install-test` again. This should fix it.

With earlier versions of LaTeX2HTML you can fix it by replacing the file `pstoppm.ps` in the LaTeX2HTML directory with a newer one that accompanies *Ghostscript 3.x*. Alternatively you can avoid using `pstoppm.ps` by changing the way `GS` is invoked in the file `pstogif`, using something like:
`open (GS, "|$GS -q -sDEVICE=ppmraw  -sOutputFile=$base.ppm $base.ps");`

Cannot get it to generate inlined images: Try a small test file; e.g.

```
% image-test.tex
\documentstyle{article}
\begin{document}
Some text followed by \fbox{some more text in a box}.
\end{document}
```

You should see something like:

```
This is LaTeX2HTML Version  (Wed Dec 1 1993) by Nikos Drakos,
Computer Based Learning Unit, University of Leeds.

OPENING /usr/cblelca/nikos/scripts/tex2html/tests/image-test.tex

Reading ....
Translating ...0/1.....1/1......
Generating images using latex ...
This is TeX, C Version 3.14t3
12222_images.tex
LaTeX Version 2.09 <7 Dec 1989>

Generating postscript images using dvips ...
This is dvips 5.521 Copyright 1986, 1993 Radical Eye Software

' TeX output 1993.12.03:1050' -> 12222_image
(-> 12222_image001) <tex.pro>[1]
Initializing... done.
Ghostscript 2.6.1 (5/28/93)
Copyright (C) 1990-1993 Aladdin Enterprises, Menlo Park, CA.
```

```
  All rights reserved.
Ghostscript comes with NO WARRANTY: see the file COPYING for details.
GS>GS>Writing 12222_image001.ppm
GS>pnmcrop: cropping 119 rows off the top
pnmcrop: cropping 961 rows off the bottom
pnmcrop: cropping 208 cols off the left
pnmcrop: cropping 484 cols off the right

Doing section links .....
Done.
```

If there is a problem somewhere during the conversion from PostScript to GIF you can try to do it manually so that you can find out where the problem is. Here is one way to do it (Please use the `pstoppm3.ps` file instead of `pstoppm.ps` if your version of *Ghostscript* is later than 3.0):

```
cblelca% latex image-test.tex
This is TeX, C Version 3.14t3
(image-test.tex
LaTeX Version 2.09 <7 Dec 1989>
(/usr/TeX/tex.lib/inputs//paper.sty
Document Style ‘paper’ <28 Nov 89>.
(/usr/TeX/tex.lib/inputs//pap11.sty)
(/usr/TeX/tex.lib/inputs/doublespace.sty)
(/usr/TeX/tex.lib/inputs//smaller.sty))
(/usr/TeX/tex.lib/inputs/psfig.sty
psfig/tex 1.9
)
No file image-test.aux.
[1] (image-test.aux) )
Output written on image-test.dvi (1 page, 652 bytes).
Transcript written on image-test.log.
cblelca% dvips -o image-test.ps image-test.dvi

This is dvips 5.519 Copyright 1986, 1993 Radical Eye Software
' TeX output 1993.11.12:1412' -> image-test.ps
<tex.pro>. [1]
cblelca% gs -dNODISPLAY pstoppm.ps
Initializing... done.
Ghostscript 2.6.1 (5/28/93)
Copyright (C) 1990-1993 Aladdin Enterprises, Menlo Park, CA.
  All rights reserved.
Ghostscript comes with NO WARRANTY: see the file COPYING for details.
GS>(image-test) ppm1run
Writing image-test.ppm
GS>quit
cblelca% pnmcrop image-test.ppm >image-test.crop.ppm
pnmcrop: cropping 61 rows off the top
pnmcrop: cropping 110 rows off the bottom
pnmcrop: cropping 72 cols off the left
pnmcrop: cropping 72 cols off the right
cblelca% ppmtogif image-test.crop.ppm >image-test.gif
```

STILL cannot get it to generate inlined images for equations, etc. If you have no problems with the `image-test.tex` file but you still cannot convert the images in some of your files have a look in the directory of the generated `HTML` files for two files `images.tex` and `images.log`. Do you notice anything unusual in them? Copy `images.tex` in the directory of your original LaTeX file and run `latex` on `images.tex`. Can you see any errors in `images.log`? If yes can you fix `images.tex` to get rid of the errors? After fixing `images.tex` you can put it back in the directory of `HTML` files created by LaTeX`2HTML` and run LaTeX`2HTML` on the original document using the option `-images_only`.

If you get into a mess try running LaTeX`2HTML` with the options `-no_reuse` and `-no_images`; e.g.

```
cblipca% latex2html -no_reuse -no_images test.tex
This is LaTeX2HTML Version 95 (Tue Nov 29 1994) by Nikos Drakos,
Computer Based Learning Unit, University of Leeds.

OPENING /tmp_mnt/home/cblelca/nikos/tmp/test.tex
Cannot create directory /usr/cblelca/nikos/tmp/test: File exists
(r) Reuse the images in the old directory OR
(d) *** DELETE *** /usr/cblelca/nikos/tmp/test AND ITS CONTENTS OR
(q) Quit ?
:d

Reading ...
Processing macros ....+.
Reading test.aux ......................
Translating ...0/1........1/1.....
Writing image file ...

Doing section links .....

*********** WARNINGS ***********

If you are having problems displaying the correct images with Mosaic,
try selecting "Flush Image Cache" from "Options" in the menu-bar
and then reload the HTML file.

Done.
```

Then try to have a look in the file `images.tex` (as described earlier) and perhaps fix it. Once you are happy that `images.tex` is OK, run LaTeX`2HTML` again with the option `-images_only`. The options `-no_reuse`, `-no_images` and `-images_only` are available with LaTeX`2HTML` version 0.7 or later.

Some problems in displaying the correct inlined images, may be due to the image caching mechanisms of your browser. With some browsers a simple "**Reload Current Document**" will be enough to refresh the images but with others (e.g. *Mosaic*) you may need to request for the cache to be refreshed. With *Mosaic* try selecting "**Flush Image Cache**" from "**Options**" in the menu-bar and then reload the `HTML` file.

It cannot do slides, memos, etc. If you use SliTeX you can go a long way just by replacing the `{slides}` argument of the `\documentstyle` command with something like `{article}` just before using LaTeX`2HTML`. One problem may be that all your slides will end up in the same `HTML` file. If you use `lslide.sty` you may get much better results (use *Archie*[26] to find this or any other style files).

References

[1] Leslie Lamport, *LaTeX, A Document Preparation System. User's Guide & Reference Manual, 2nd Edition.* Addison–Wesley Publishing Company, Inc., 1994. Online information on TeX and LaTeX is available at `http://curia.ucc.ie/info/TeX/menu.html` and `http://es-sun2.fernuni-hagen.de/info2html?(latex.info)Top` .

[2] Michel Goossens, Frank Mittelbach, Alexander Samarin, *The LaTeX Companion.* Addison–Wesley, 1994 ISBN 0–201–54199–8

[26] http://www.pvv.unit.no/archie/

Glossary of variables and file-names

`.dvipsrc` initialisation file for `dvips` . . . 3, 5
`.latex2html-init` initialisation file for LaTeX2HTML . . . 5, 7, 11, 12, 37, 42
`<file>.ptr` initialises sectioning information for a specific segment within a segmented document . . . 18
`<file>.tex` LaTeX source for a document segment or whole document . . . 18
`<prefix><type>.pl` data file for *Perl*, containing information of the specific ***<type>***; needed especially with segmented documents . . . 18
`$AUTO_PREFIX` . . . 5, 18
`$CUSTOM_BUTTONS` . . . 23
`$DONT_INCLUDE` . . . 35
`$DVIPS_MODE` . . . 5
`$FIGURE_SCALE_FACTOR` . . . 9, 24, 25
`$HOME/.latex2html-init` . . . 5, 10, 11, 22, 35
`$ICONSERVER` . . . 6
`$LANGUAGE_TITLES` . . . 24
`$LATEX2HTMLDIR` . . . 4, 35, 43, 44, 46
`$LATEX2HTMLDIR/latex2html.config` . . . 11, 22
`$LATEX2HTMLDIR/styles/german.perl` . . . 10
`$MATH_SCALE_FACTOR` . . . 9, 24
`$NETSCAPE_HTML` . . . 25
`$PAPERSIZE` . . . 37
`$PK_GENERATION` . . . 5
`$SERVER` . . . 5
`$SHORT_INDEX` . . . 31
`$TEXINPUTS` . . . 5
`$WORDS_IN_NAVIGATION_PANEL_TITLES` . . . 22
`$WORDS_IN_PAGE` . . . 9

anonymous ftp pseudonym for use of the *ftp* program as the `anonymous` user, for file-transfers . . . 2
Archie publicly-accessible database of files and programs available via *anonymous ftp* or using the Web[27] . . . 2, 3, 35, 41
Arena Web-browser[28]; currently the only one which displays `HTML` 3.1 math markup 23

`bibtex` bibliography data-base program, used with LaTeX . . . 37

CERN originally named "Conseil Europeen pour la Recherche Nucleaire", now known as the "European Laboratory for Particle Physics"[29], located near Geneva; also home of the World-Wide Web Project[30] . . . 28
`changebar.sty` LaTeX 2_ε package to mark changes in documentation; this supercedes an earlier package . . . iii, 6, 29
`Changes` text file, recording the changes made to LaTeX2HTML since it was first released in July 1993 . . . 2, 7
`changes.tex` input file for the LaTeX2HTML manual; normally loaded only with the `HTML` version . . . 6

[27] http://www.pvv.unit.no/archie/
[28] http://www.w3.org/hypertext/WWW/Arena
[29] http://www.cern.ch/Welcome.html
[30] http://www.w3.org/hypertext/WWW/TheProject.html

`images.log` the log-file produced by running LaTeX on `images.tex` 40

`images.tex` file created by LaTeX2HTML which is passed to LaTeX to typeset complicated environments for conversion to images 40, 44

`install-test` tests the LaTeX2HTML installation 2, 4, 5, 34, 35, 38

`justify.perl` package defining HTML 3.2 markup for left/right/center justified paragraphs, interpreting the standard LaTeX environments flushleft/flushright/center . 44

`justify.sty` LaTeX package which does nothing other than force loading of `justify.perl` by LaTeX2HTML 6

`l2hfiles.dat` contains the descriptive data appearing in this glossary 6, 32, 33

`l2hglo.ist` configuration for `makeindex` to construct this glossary 6, 33, 45

`l2hidx.ist` configuration for `makeindex` to construct the index 6, 45

`l2hman.sty` style-file needed for typesetting the LaTeX2HTML manual using LaTeX . . 6

`labels.pl` data file containing symbolic labels for references to locations within a HTML document produced by LaTeX2HTML 16, 17, 28

`latex` command to run to LaTeX 3–7, 20, 37, 40

LaTeX the markup language, interpreted by LaTeX2HTML; images are processed by the LaTeX program i

LaTeX 2_ε the current version of LaTeX, first introduced in 1993 i

`latex2html` Unix command to execute the LaTeX2HTML script for generating HTML pages from a LaTeX document 2, 4, 5, 10, 20, 34, 35

LaTeX2HTML *Perl* script to convert LaTeX documents into web-pages, by Nikos Drakos with updates and extensions by others i

`latex2html-96.1.tar.gz` compressed archive of the latest (since Feb. 1996) LaTeX2HTML distribution 2

`latex2html.config` configuration file for LaTeX2HTML, normally to be found in the `$LATEX2HTMLDIR` directory 2, 4–6, 24, 37

`licence.tex` input file for the LaTeX2HTML manual; contains the licence statement and copyright notice 6

Linux Unix system for DOS-based PCs; LaTeX2HTML will work, provided some variables are set correctly 5

`lslide.sty` LaTeX package for producing colour-separations 41

`make` Unix utility used to track the time-dependency of multiple files in the development of a document or computing project 20, 45

`Makefile` contains the time-dependency data used by the `make` utility 20, 51

`makeidx.perl` adds *Perl* code to interpret index-entries otherwise intended for the `makeindex` program . . . . ii, 31

`makeindex` utility for sorting and filtering textual data; used primarily for making the index or glossary 6, 32, 44, 45

`makemap` *Perl* script to generate image-maps 2, 4, 5, 27, 46

`MakeTeXPK` script used by `dvips`, and other `dvi`-drivers, to control the automatic generation of missing fonts 5

`manhtml.tex` input file containing definitions needed for segments of the HTML version of the LaTeX2HTML manual 6

`manual.aux` auxiliary file created by running LaTeX on the LaTeX2HTML manual 6

`manual.glo` collects (unsorted) glossary entries for the LaTeX version of the LaTeX2HTML manual 31, 32, 45

`manual.gls` file containing the sorted entries for the glossary in the LaTeX2HTML manual; created by running `makeindex` on `manual.glo` using `l2hglo.ist` 6, 33

[31] http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html
[32] ftp://ftp.x.org/R5contrib/
[33] http://home.netscape.com
[34] ftp://ftp.x.org/R5contrib/
[35] ftp://ftp.uu.net/languages/perl/

Index