The LaTeX2HTML Translator

Nikos Drakos
Computer Based Learning Unit
University of Leeds.

March 26, 1999

Documentation revised and updated for V97.1 and HTML 3.2;
and further revisions for V98.1 and later, and for HTML 4.0 by:

Ross Moore
Mathematics Department
Macquarie University, Sydney.

This document accompanies LaTeX2HTML version 99.1 (beta)[1].

This manual differs from earlier versions by updates to the section "Installation and Further Support", a few small changes in other sections, and a shortening of "Known Problems" by removing references to old problems that no longer occur. Not all of the newer features of LaTeX2HTML V99.1 are described yet. A fully updated manual is underway and will be released when completed.

History Nikos Drakos' original manuscript was updated for version V96.1 by Herb Swan and converted for LaTeX 2_ε by Michel Goossens. Extensive revisions were made by Ross Moore for V96.1 `rev-f`, incorporating also suggestions from Michel Goossens. Another major revision was required to adequately describe the new features made possible with HTML 3.2, and recent developments in image-generation and macro-handling. This work was done by Ross Moore, as were most of the revisions for V98.1, V98.2 and V99.1. Portability for non-Unix systems has been achieved due to work done mainly by Marek Rouchal, Uli Wortmann, Fabrice Popineau and Daniel Taupin.
Changes for the V98.1 revision are indicated with narrow change-bars. Wider change-bars indicate where the most recent changes and features added in V99.1 are described.

[1] **This is a preliminary document for LaTeX2HTML 99.1.**

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, including the power of a LaTeX-like macro language capable of producing correctly structured HTML tags.

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 etc. are converted to images which are placed automatically at the correct position 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 and install it, and how to use it effectively.

Credits, 1993–1994

Several people have contributed suggestions, ideas, solutions, support and encouragement. Some of these are Roderick Williams, Ana Maria Paiva, Jamil Sawar and Andrew Cole at the Computer Based Learning Unit[2].

CERN The idea of splitting LaTeX files into more than one component, connected via hyperlinks, was first implemented in *Perl* by Toni Lantunen at CERN. Thanks to Robert Cailliau of the World-Wide Web Project, also at CERN, for providing access to the source code and documentation (although no part of the original design or the actual code has been used).

Robert S. Thau 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.

Eric Carroll who first suggested providing a command like `\hyperref`.

Franz Vojik provided the basic mechanism for handling foreign accents.

Todd Little The '`-auto_navigation`' option was based on an idea by Todd.

Axel Belinfante provided the *Perl* code in the `makeidx.perl` file, as well as numerous suggestions and bug-reports.

Verena Umar (from the Computer Science Education Project[3]) has been a very patient tester of some early versions of LaTeX2HTML and many of the current features are a result of her suggestions.

Ian Foster and Bob Olson. Thanks to Ian Foster and Bob Olson at the Argonne National Labs, for setting up the LaTeX2HTML mailing list[4].

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

Later Developments, 1995–1996

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 for his many extensive revisions;

Mark Noworolski for coordinating V95.3;

Sidik Isani for his improvement in GIF quality;

Michel Goossens was the driving force behind the upgrade to LaTeX 2_ε compatibility, and other features developed at CERN;

Herb Swan 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 V96.1 `rev-f`;

Ross Moore who has revised and extended this manual, helped design and test the segmentation strategy, and later revisions of V96.1 . Ross organised the release of V96.1 `rev-g` and provided many of the improvements incorporated into V96.1 `rev-h`.

Martin Wilck for the initial work on implementation of frames. Also Martin did most of the work implementing the extensive citation and bibliographic features of the natbib package, written by Patrick Daly. He also provided the `makeseg` *Perl* script to create Makefiles for segmented documents.

Jens Lippmann for organising the releases V96.1 `rev-h` to V98.1. Jens made significant contributions to the internal workings of LaTeX2HTML, as well as cleaning up much of its source code.

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

Thanks also to Donald Arseneau for allowing his `url.sty` to be distributed with this manual. Similarly, thanks to Johannes Braams for `changebar.sty`. Both of these are useful utilities which enhance the appearance of the printed manual. In particular, changes introduced with V98.1 and its revisions are denoted by thin change-bars, while thicker bars denote changes introduced with V98.2 and later releases.

Developments: late 1996 to mid 1997

During the latter part of 1996 there was much work on improving the capabilities of LaTeX2HTML. Some of this was due to the World Wide Web Consortium[5]'s proposals for HTML 3.2, becoming a formal recommendation in November 1996, and their subsequent acceptance in January 1997. Existing LaTeX markup for effects such as centering, left- or right-justification of paragraphs, flow of text around images, table-layout with formal captions, etc. could now be given a safe translation into HTML 3.2, compliant with a standard that would guarantee that browsers would be available to view such effects.

At the same time developers were exploring ways to enhance the overall performance of LaTeX2HTML. As a result the current V97.1 release has significant improvements in the following areas:

image-generation is much faster, requires less memory and inline images are aligned more accurately;

image quality is greatly improved by the use of anti-aliasing effects for on-screen clarity, in particular with mathematics, text and line-drawings;

memory-requirements are much reduced, particularly with image-generation;

mathematics can now be handled using a separate parsing procedure; images of sub-parts of expressions can be created, rather than using a single image for the whole formula;

macro definitions having a more complicated structure than previously allowed, can now be successfully expanded;

counters and numbering are no longer entirely dependent on the `.aux` file generated by LaTeX;

decisions about which environments to include or exclude can now be made;

HTML effects for which there is no direct LaTeX counterpart can be requested in a variety of new ways;

HTML code produced by the translator is much neater and more easily readable, containing more comments and fewer redundant breaks and `<P>` tags.

error-detection of simple LaTeX errors, such as missing or unmatched braces, is now performed — a warning message shows a line or two of the source code where the error has apparently occurred;

For these developments, thanks goes especially to:

Jens Lippmann for creating and maintaining the CVS repository at `http://saftsack.fs.uni-bayreuth.de/~latex2ht/user/`. This has made it much easier for the contributions from different developers to be collected and maintained as a "development version" which is kept up-to-date and available at all times. Together with Marek Rouchal he produced an extensive rewrite of the `texexpand` utility.

[5] http://www.w3c.org/

Ross Moore for extensive work on almost all aspects of the LaTeX2HTML source and documentation, combining code for LaTeX, *Perl*, HTML and other utilities. Most of the coding for the new features based on HTML 3.2, many of the new packages, faster image-generation and the improved support for mathematics and other environments, is his work.

Marek Rouchal for extending the former `pstogif` utility, transforming it into `pstoimg` which now allows for alternative image formats, such as PNG. Also he produced the neat `configure-pstoimg` script, which eases LaTeX2HTML installation, and a rewrite of `texexpand`.

Marcus Hennecke who has always been there, up-to-date with developments in HTML and related matters concerning Web publishing, and tackling the issues involved with portability of LaTeX2HTML to Unix systems on various platforms.

Furthermore Marcus has produced LaTeX2HTML-NG, a version of LaTeX2HTML which handles expansion of macros in a more "TeX-like" fashion. This should lead to further improvements in speed and efficiency, while allowing complicated macro definitions to work as would be expected from their expansions under LaTeX. (This requires *Perl* 5, using some programming features not available with *Perl* 4.)

Fabrice Popineau has produced an adaptation for the Windows NT platform, of LaTeX2HTML V97.1.

Uli Wortmann showed how to configure *Ghostscript* to produce anti-aliasing effects within images.

Axel Ramge for various suggestions and examples of enhancements, and the code to avoid a problem with *Ghostscript*.

Thanks also to all those who have made bug-reports, supplied fixes or offered suggestions as to features that might allow LaTeX2HTML to be used more efficiently in particular circumstances. Most of these have been incorporated into this new version V97.1, though perhaps not in the form originally envisaged. Such feedback has contributed enormously to helping make LaTeX2HTML the easy to use, versatile program that it has now become.

Keep the ideas coming!

1st LaTeX2HTML Workshop
Darmstadt, 15 February 1997

Thanks again to Jens Lippmann and members of the LiPS Design Team for organising this meeting; also to the Fachbereich Informatik at Darmstadt for use of their facilities.
This was an opportunity for many of the current LaTeX2HTML developers to actually meet for the first time; rather than communication by exchange of electronic mail messages.

- Nikos Drakos talked about the early development of LaTeX2HTML, while...
- ...Ross Moore, Jens Lippmann and Marek Rouchal described recent improvements.
- Michel Goossens presented a list of difficulties encountered with earlier versions of LaTeX2HTML, and aspects requiring improvement. Almost all of these now have been addressed in the V97.1 release, so far as is possible within the bounds inherent in the HTML 3.2 standard.
- Kristoffer Rose showed how it is possible to create GIF89 animations from pictures generated by TeX or LaTeX, using the Xy-pic graphics package and extensions, developed by himself and Ross Moore.

Also present were representatives from the DANTE e.V. Praesidium and members of the LaTeX3 development team. In all it was a very pleasant and constructive meeting.

TUG'97 — Workshop on LaTeX2HTML
University of San Francisco, 28 July 1997

On the Sunday afternoon (2.00pm–5.00pm) immediately prior to the TUG meeting, there will be a workshop on LaTeX2HTML, conducted by Ross Moore[6].

Admission: $50, includes a printed copy of the latest LaTeX2HTML manual.

TeXNortheast TUG Conference, TeX/LaTeX Now
March 22–24, 1998, New York City

Includes a workshop/presentation by Ross Moore[7].

Euro-TeX'98, 10th European TeX Conference
St. Malo, France — 29–31 March, 1998

Several of the LaTeX2HTML developers will be present. All European (and other) LaTeX2HTML users are encouraged to attend.

[6] Mathematics Department, Macquarie University, Sydney, Australia
[7] Mathematics Department, Macquarie University, Sydney, Australia

Developments: late 1997 to early 1998

Much of the work contributed to LaTeX2HTML during this time was related to bug fixing and maintaining the 97.1 release, in order to reach a more stable and reliable version which produces HTML code conforming to the W3C standards/drafts. To keep in context with this view, support for HTML 4 has been incorporated into the translator.

There have been improvements to the way math code is handled, as well as font-changing and numbering commands. These now are expected to work much closer to the way that LaTeX handles them.

Furthermore, missing LaTeX style translations for basic LaTeX and AMS-TeX document classes were added to the distribution: `book.perl`, `report.perl`, `article.perl`, `letter.perl`, `amsbook.perl` and `amsart.perl`. New styles implementing LaTeX packages include `seminar.perl`, `inputenc.perl` and `chemsym.perl` naming but a few.

The aim is ultimately to support all LaTeX, AMS-TeX etc. packages in the standard LaTeX distribution, or for which there is published documentation. At the time of writing this aim has not quite been reached. To support internationalisation, *Perl* extensions were provided for HTML output conforming to ISO-Latin 1, 2, 3, 4, 5, 6, and Unicode[8] encodings.

All of the above work was done by Ross Moore.

Additional document formats are now supported, these are IndicTeX, FoilTeX, and CWEB documents. You may use any of these packages to translate such documents together with LaTeX2HTML, refer to the instructions in the various README files.

Thanks go to Ross Moore for IndicTeX/HTML[9], to Boris Veytsman for FoilTeX/HTML and to Jens Lippmann for the CWEB to HTML translator.

Numerous discussions and efforts have been undertaken to get LaTeX2HTML working independent from the underlying operating system. Yet all obstacles are not quite taken, but it is forseeable that we are OS independent very soon. This release has been reported to run on OS/2, DOS, and MacOS, besides Unix-like operating systems. A former version has also been ported to Amiga OS, but that results still need to be re-integrated into the source. Ports for Windows'95 and Windows NT exist, but are not yet integrated with the main distribution.

Thanks go to Marcus Hennecke, Axel Ramge, Marek Rouchal and Uli Wortmann for fruitful and refreshening discussions about that `Override.pm` loading scheme (which finally made its way after enough chickens and eggs chased one another to death ☺), and to Daniel Taupin for his successful efforts to get LaTeX2HTML running on DOS.

Thanks go also to Fabrice Popineau for his port to Windows NT [10], and Nikos Drakos for a Windows 95 port based on V96.1h [11] (which is mentioned here at last, but not least).

We want to take the opportunity to thank Scott Nelson and the people at Lawrence Livermore National Laboratory who help to keep up the LaTeX2HTML main archive and the mailing list, and to Achim Bohnet at the Max Planck Institut fuer extraterrestrische Physik, Garching for maintaining the list's online archive. Finally thanks and greetings to all people that contributed to this release and have not been mentioned here...

You all showed spirit and favour. Thank you for your efforts!

---

[8] http://www.unicode.org/

[9] http://www-texdev.mpce.mq.edu.au/l2h/indic/IndicHTML/

[10] CTAN: .../tex-archive/systems/win32/web2c/l2h-win32.tar.gz

[11] ftp://ftp.mpn.com/pub/nikos/latex2html96.1-h-win32.tar.gz

. . . and don't forget Jens and the LiPS team at Darmstadt!

Proposals for Future Development:

LaTeX2HTML-NG

Developed by Marcus Hennecke this is a version of LaTeX2HTML that addresses various issues, not currently handled in the best way by version v97.1. These include:

- validating the HTML output, so that only correctly nested tags, and their contents, can be produced by the translator;
- more TeX-like order of macro-expansion, so that macros and their expansions will produce exactly the results expected from the TeX implementation of LaTeX;
- faster processing, by streamlining some of the current *Perl* code, and allowing shorter strings to be handled at any given time;
- customisation issues, allowing easier portability to Unix-like environments on other platforms.

Many of these features have been the inspiration for new code written for LaTeX2HTML v98.1.

The current version of LaTeX2HTML-NG can be obtained from the developer's repository, see page 6, in the directory `http://saftsack.fs.uni-bayreuth.de/~latex2ht/ng-user`. Beware that the files there are *not* compatible with those of the same name that come with the current version of LaTeX2HTML.

Extended Capabilities in Web browsers

The following areas are the subject of active development within the Web community. Limited support is available within LaTeX2HTML for some of these features, using the `-html_version 4.0` command-line switch.

style-sheets: proposals for a flexible mechanism to allow cascading (CSS) and DSSSL, within HTML 4.0[12].

XML: eXtensible Markup Language[13].

MathML: Mathematical Markup Language[14].

CML: Chemical Markup Language[15].

Fonts: further support for non-standard font encodings.

Icons: Alternative sets of icons for navigation buttons and other purposes.

For some background on these technologies read Michel Goossens' survey article "Hyperactivity in the Web-world" in *CERN Computer Newsletter* No. 227[16], and browse Axel Ramge's site[17] for ideas on how they could be used with LaTeX2HTML.

---

[12] http://www.w3.org/pub/Markup/
[13] http://www.w3.org/pub/WWW/TR/WD-xml.html
[14] http://www.w3.org/pub/WWW/TR/WD-math-970515
[15] http://www.venus.co.uk/omf/cml
[16] http://wwwinfo.cern.ch/cnls/227/art_xml.html
[17] http://www.ramge.de/ax/latex2html/latex2html.html

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.

Year 2000 compliance: LaTeX2HTML contains *no* executable software, *per se*. It consists entirely of scripts to run other pieces of software: *Perl*, LaTeX, *Ghostscript*, `netpbm`, etc. and standard Unix utilities (e.g. `cp`, `rm`, `make`, `ln`, ...) as well as the operating system shell.

These other pieces of software are to be obtained and installed independent from the LaTeX2HTML scripts.

LaTeX2HTML makes no reference to dates, apart from reading the current date from the operating system, and converting the resulting string data into a standard form. This may result in '00' appearing in the year 2000. However this representation of the date is used *for display only*; it does *not* control any further processing.

Contents

[18] 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 manual describes the LaTeX2HTML translator which is used to create Web pages from document source written for the LaTeX typesetting system, or simply containing LaTeX commands.

To use LaTeX2HTML to translate a file `<`*file*`>.tex` containing LaTeX commands, simply type:

```
latex2html <file>.tex
```

This will create a new directory called `<`*file*`>` which will contain the generated HTML files, some log files and possibly some images.

Basically the translator reads the source document and creates a linked set of HTML pages, displaying the information it contains. The LaTeX commands and environments that are found are interpreted either as "markup" instructions, or as macros expanding into more text or markup commands. Where such markup corresponds to the intended use for markup tags in the HTML language, a direct translation is made. If there is no natural way to present the information using simple text embellished with HTML markup tags, then an image is generated, using LaTeX itself to interpret the portion of code.

Of course this is a drastically over-simplified description of what LaTeX2HTML actually does. Many questions spring readily to mind. The answers to these and the options available to handle particular situations are discussed elsewhere in this manual.

- *What does "natural way to present the information" really mean?*

Text and paragraphing clearly should appear as such, whether printed or on-screen. Different font sizes and styles such as "bold-face" or "italic" are generally rendered accordingly. However, whereas LaTeX has access to appropriate fonts for specialised purposes such as mathematical symbols, these cannot be guaranteed to be available with all Web-browsers. So for information requiring such things, LaTeX2HTML will generally resort to making an image, using LaTeX itself to typeset the material required for that image.

Section 1.1 contains a brief overview of how LaTeX's standard environments are handled within LaTeX2HTML. It also mentions some of the extra features that are available. In general LaTeX2HTML attempts to use textual constructions to represent the required information. Generation of an image is done only when there is no adequate textual construction with the required version of HTML, or when specifically requested to do so. Various extensions, to cope with the different HTML versions and extra features, are discussed in Section 3. That describes what to expect on the HTML pages, with little or no changes required to the LaTeX source.

Just as LaTeX has various packages which can be used to present specific types of information in appropriate ways, so is LaTeX2HTML capable of handling the commands from many of these packages. See Table 6 for a listing of those packages which currently have special support.

- *Some features of HTML have no direct counterpart in a LaTeX typeset document. Can such features be used with LaTeX2HTML?*

Any effect currently available with any version of the HTML standard can be specified for a document processed by LaTeX2HTML. New LaTeX commands are defined in the `html.sty` package; the features that these commands allow are the subject of Section 4 in this manual. Some of the new commands provide improved strategies for effects already existing in LaTeX;

e.g. cross-references and citations. To use these effectively requires only small changes to the LaTeX source.

Other commands define new environments which are completely ignored when processed by LaTeX. Indeed the full scope of HTML 3.2 is available, using LaTeX-like macros to help structure the source, reduce the tedium of repetitious use of tags, and ensure that all appropriate tags are correctly closed.

- *What determines the amount of information that goes onto a single HTML page? How are different pages linked?*

The HTML pages can contain whole chapters, sections, (sub)subsections or (sub)paragraphs. This is fully customisable using the command-line options discussed in detail in Section 5.2 of this manual.

- *Does the original document have to be a valid LaTeX document, typesetting without errors? If not, does it help if it is?*

In fact any document can be fed to the LaTeX2HTML processor, but it is designed specifically to recognise and sensibly translate the intentions expressed by LaTeX markup commands. Although sensible results can be obtained even when the LaTeX source is not valid, the most reliable translations are obtained when it is. Relevant issues are discussed in Section 5.1.

- *When developing a document which contains special HTML features, is it best to regularly test it in LaTeX or with LaTeX2HTML?*

The answer to such a question changes as the developer gains more experience with the available tools. Some aspects to be considered are discussed in Section 5.1 of this manual.

Information relevant to obtaining the latest version of LaTeX2HTML, installation within the local environment, and where to look for help when things do not go as expected, can be found in Section 2.1.

What follows next is a brief summary of the features supported within LaTeX2HTML.

1.1 List of Features

Following is a listing of the main features of the translator; more specific details on these is given elsewhere in this manual.

The LaTeX2HTML translator ...

- 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 1), tables (see Table 4), figures (see Figure 2), and *any arbitrary environment.* Either the complete environment or subparts thereof are passed to LaTeX for conversion to images, which are then either included in the document or are made available through hypertext links.

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

- figures or tables can be arbitrarily scaled and oriented, and shown either as inlined images or "thumbnail" sketches or their contents displayed within a table constructed using the `<TABLE>` tags of `HTML` 3.2.
- theorem-like environments are supported, along with automatic numbering and counter dependencies.
- can produce output suitable for browsers that support inlined images or character-based browsers (as specified by the user). In particular the TeX or LaTeX code for mathematical expressions and formulas will be displayed in character-based browsers, such as lynx.
- coloured text and/or background is fully supported, as is the ability to use an image to create a tiled backdrop.
- handles definitions of new commands, environments and counters even when these are defined in external files for input[8];
- handles footnotes[9], tables of contents, lists of figures and tables, bibliographies and can generate an index. By including hyperlinks between index entries, simple navigation aids can be built into the index, for easy browsing.
- automatically translates cross-references and citations 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, else an image can be created;
- 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 virtually all of the concepts and commands described in the LaTeX blue book , where there is a meaningful interpretation appropriate to an `HTML` document. Also many other LaTeX constructions are handled, including many described in the LaTeX *Companion*[2] and LaTeX *Graphics Companion*[3, XY-pic];
- can be configured to translate equations either as GIF images or as `HTML` 3.0 mark-up (as browsers become available which are suitable for the task), or by making images of subparts of equations, as required.
- links symbolic references across document segments which have been independently processed;
- will try to translate any document with embedded LaTeX commands, irrespective of whether it is complete or syntactically legal.

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

[9] Like this!

1.2 Exemplary Documents

Here is a selection of documents illustrating different contexts in which LaTeX2HTML has been used. This list is by no means exhaustive, but all links were valid as of June 1997.

An earlier listing of converted documents can be found at: `http://cbl.leeds.ac.uk/~nikos/tex2html/doc/latex2html/node6.html#sample`. However some of the links are no longer valid.

LaTeX2HTML documentation and usage

- The LaTeX2HTML Translator, User Manual
  `http://www-texdev.mpce.mq.edu.au/l2h/docs/manual/`
  `http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/`
- Mathematics with LaTeX2HTML
  `http://www-texdev.mpce.mq.edu.au/l2h/mathdocs/amsmath2/`
- LaTeX2HTML, Style Sheets, XML
  `http://www.ramge.de/ax/latex2html/latex2html.html`
- *All about* LaTeX2HTML, by Nikos Drakos[10]
- What is LaTeX2HTML? (with XY-pic diagrams)
  `http://www.maths.mq.edu.au/texdev/Xypic/L2Htalk/`
- German specials in TeX
  `http://www.maths.mq.edu.au/texdev/tests/harn/node1.html`
- Crayola Colours
  `http://www.maths.mq.edu.au/~ross/latex/crayola/crayola.html`

Software Manuals, Computing Resources

- XY-pic User's Guide, accessible from the XY-pic Home Page[11] or "down-under"[12]
- LiPS — A System for Distributed Processing on Workstations[13]
  Manual, Lecture Notes and various theses
- LINUX Documentation Project
  `http://linuxwww.db.erau.edu/ldp/linux.html/`
  Getting Started, The LINUX Kernel, Network Administrator's Guide
  Programmer's Guide, System Administrator's Guide
- ECLiPSE — The ECRC Constraint Logic Parallel System[14]
  User Manual, Extensions User Manual
- MPQC — Massively Parallel Quantum Chemistry Program[15]
  User Guide, Scientific Computing library[16]
- Xgraphics
  `http://www-theorie.physik.uni-wuerzburg.de/~lueders/Xgraphics/`
- *Glish* 2.6, User Manual
  `http://aips2.nrao.edu/aips++/docs/html/aips++.html`
- UTCC — University of Tennessee, Computing Center[17]

[10] http://cbl.leeds.ac.uk/~nikos/tex2html/doc/latex2html/latex2html.html
[11] http://www.brics.dk/~krisrose/Xy-pic.html
[12] http://www.maths.mq.edu.au/texdev/xyguide-html/
[13] http://cdc-server.cdc.informatik.tu-darmstadt.de/
[14] http://www.ecrc.de/eclipse/eclipse.html
[15] http://midway.ca.sandia.gov/~cljanss/mpqc.html
[16] http://midway.ca.sandia.gov/~cljanss/mpqc/prog/prog.html
[17] http://www.utcc.utk.edu/utcc/docs/

Facilities and Services, GNU Emacs, Help Sheet, Intro to Unix, `vi` editor
Software Support

Journals, Conference Proceedings, Newsletters

- "CERN Computer Newsletter"
  `http://consult.cern.ch/cnls/` issues 214–224.
- "National Symposium in Mathematics"
  `http://www.maths.mq.edu.au/texdev/MathSymp/`
- American Mathematical Society ($\mathcal{AMS}$) (`http://www.ams.org/journals/`)
  Articles in the following electronic journals use a variant of LaTeX2HTML:
  Bulletin of the $\mathcal{AMS}$, *Conformal Geometry and Dynamics*
  Electronic Research Announcements

  Other $\mathcal{AMS}$ journals are available only to subscribers:
  Journal of the $\mathcal{AMS}$, *Mathematics of Computation*, *Representation Theory*
  Proceedings of the $\mathcal{AMS}$, *Transactions of the $\mathcal{AMS}$*

Encyclopædic Reference Material

- *Eric's Treasure Troves of Science*:[18] Astronomy, Chemistry, Math
  Music, Physics, Rocket History, Scientific Books, Scientific Biographies
- Resources for Economists on the Internet
  `http://wueconb.wustl.edu/EconFAQ/`
- *Les mathèmes de Lacan* by Jacques Siboni
  `http:www.shef.ac.uk/~psysc/thesaur3/index.html`

Course Materials

- Mathematics Courses at Macquarie University[19]
  MATH130, MATH132, MATH233, MATH300 MATH337,
- Engineering Science 100[20], Simon Fraser University.

Other Interesting Sites

- rel@X (`http://www.ramge.de/ax/ax.html`)
- Uli Wortmann's thesis and Geological papers use anti-aliased images;
  Die untere und mittlere Kreide[21], *The Barium Problem*[22].
- Stanford Computer Graphics Laboratory, Publications[23]
  Web Visualization in Hyperbolic Space, Topology of the MBONE, Complex Models from Range Images, Optical Triangulation, Metamorphosis
- The RGO Worldwide Guide to Public Videoconference Centers
  `http://www.ast.cam.ac.uk/~ralf/vcguide/`

[18] http://www.astro.virginia.edu/~eww6n#TreasureTroves
[19] http://www.maths.mq.edu.au/~ross/index.html#courses
[20] http://fas.sfu.ca/ensc/people/Faculty/jones/personal/ensc100/
[21] http://bonk.ethz.ch/papers/diss/main-node78.html
[22] http://bonk.ethz.ch/papers/lecture-notes-97/ln97-node10.html
[23] http:/www-graphics.stanford.edu/papers/

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. They are located at

US United States: `http://ctan.tug.org/ctan/`[24],

UK United Kingdom: `http://www.tex.ac.uk/`[25]

DE Germany: `ftp://ftp.dante.de/`[26].

In the directory `tex-archive/support/latex2html/` should be the latest version, as a compressed archive.

There are also many mirrors. To find the nearest to you, get a listing via the command:

```
finger ctan@www.tug.org
```

The site at `http://saftsack.fs.uni-bayreuth.de/~latex2ht/` is a convenient alternative for European users. This is connected to the developer's repository, so should always have the most recent release.

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/`.

Two other *ftp*-sites are:

- `http://ctan.tug.org/ctan/tex-archive/support/latex2html`
- `http://ftp.rzg.mpg.de/pub/software/latex2html/sources/`

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) or more recent Web-searching tools such as *FTP search* in Norway.

> **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/`.

For users of Windows NT, there is a port of LaTeX2HTML obtainable from `ftp://ftp.ese-metz.fr/pub/TeX/win32`. Obtain the release from this site and follow the instructions in the accompanying file `README.win32`. Thanks to Fabrice Popineau for this work. In future it is planned to merge this code with the main distribution.

Finally there is the LaTeX2HTML developers' CVS repository, at `http://saftsack.fs.uni-bayreuth.de/~latex2ht/user/`.
The files to be found here are the most up-to-date with current developments, but they cannot be guaranteed to be fully reliable. New features may be still under development and not yet sufficiently tested for release. A daily updated compressed archive of the developers' work may be downloaded from `http://saftsack.fs.uni-bayreuth.de/~latex2ht/l2h-latest.tar.gz`.

[24] `http://ctan.tug.org/ctan/tex-archive/support/latex2html`
[25] http://www.tex.ac.uk/`tex-archive/support/latex2html`
[26] ftp://ftp.dante.de/`tex-archive/support/latex2html`

Warning: Use the files from this site at your own risk.

Having obtained a compressed `tar` version, save it into a file `latex2html-98.1.tar.gz` say, then extract its contents with

```
% gzip -d latex2html-98.1.tar.gz
% tar xvf latex2html-98.1.tar
```

You should then have the following:

- `README` file;
- `Changes` index with latest changes;
- (no longer supplied);
- `latex2html` *Perl* script;
- `texexpand` *Perl* script[27];
- `latex2html.config` configuration file;
- `install-test` *Perl* script, for installation and testing;
- `dot.latex2html-init` sample initialisation file;
- `texinputs/` subdirectory, containing various LaTeX style-files;
- `versions/` subdirectory, containing code for specific `HTML` versions;
- `makemap` *Perl* script;
- `example/` subdirectory, containing the segmentation example, described in detail in Section 4.10;
- `.dvipsrc` file;
- `pstoimg` *Perl* script for image conversion (replaces `pstogif`);
- `configure-pstoimg` *Perl* script for installation;
- `local.pm` *Perl* input file;
- `icons.gif/` subdirectory, containing icons in GIF format;
- `icons.png/` subdirectory, containing icons in PNG format;
- `makeseg` *Perl* script and examples to handle segmented documents via a generated Makefile, see `makeseg.tex`;
- `docs/foilhtml/` contains LaTeX package and *Perl* implementation by Boris Veytsman, to support FoilTeX to `HTML` translation;
- `IndicTeX-HTML/` package that contains *Perl* and LaTeX code for translating IndicTeX documents (see README file);
- `docs/` subdirectory, containing the files needed to create a version of this manual;
- `styles/` subdirectory, containing *Perl* code for handling some style-files;
- `tests/` contains some test documents for LaTeX2`HTML`.

[27]Initially written by Robert S. Thau, completely rewritten by Marek Rouchal and Jens Lippmann.

2.1.1 Requirements

The translator makes use of several utilities all of which are freely available on most platforms. You may use *Archie*[28], or other Web-searching tools such as *FTP search*[29], to find the source code of any utilities you might need.

For the best use of LaTeX2HTML you want to get the latest versions of all the utilities that it uses. (It will still work with earlier versions, but some special effects may not be possible. The specific requirements are discussed below.)

- *Perl* version 5.002, or later (check with `perl -v`);
  Perl should be compiled to use the `csh` or `tcsh` shell, though LaTeX2HTML can also work with the `bash` shell if *Perl* is recompiled to use it as "full csh". Don't worry about this; any missing programs should be reported by `install-test` upon installation.
- LaTeX, meaning LaTeX 2_ε dated `<1995/06/01>`, or later;
- `dvips` or `dvipsk`, at version 5.58 or later;
- *Ghostscript* at version 4.02 or later;
- `netpbm` library of graphics utilities, version 1-MAR-94
  (check with `pnmcrop` '`-version`').

More specific 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 **Note: LaTeX2HTML requires *Perl* 5 to operate**.
     Warning 1: You really *do* need *Perl* 5.
     Versions of LaTeX2HTML up to V96.1h work both with *Perl* 4 at patch level 36 and *Perl* 5 , though some of the packages may only work with *Perl* 5 .
     Warning 2: Various aspects of *Perl*, which are used by LaTeX2HTML, assume certain system commands to be provided by the operating system shell. If `csh` or `tcsh` is used to invoke LaTeX2HTML then everything should work properly.
     Perl 5 eliminates this requirement on the shell.
   - *DBM* or *NDBM*, the Unix DataBase Management system, or *GDBM*, the GNU database manager.
     Note: Some systems lack any DBM support. *Perl* 5 comes with its own database system SDBM, but it is sometimes not part of some Perl distributions.
     The installation script `install-test` will check that for you. If no database system is found, you will have to install Perl properly.

2. **LaTeX commands with equations, figures, tables, etc.**
   As above plus ...
   - `latex` (version 2e recommended but 2.09 will work — with reduced ability to support styles and packages);

[28] http://www.pvv.unit.no/archie/
[29] http://ftpsearch.ntnu.no/ftpsearch

- `dvips` (version 5.516 or later) or `dvipsk`
  Version 5.62 or higher enhances the performance of image creation with a *significant* speed-up. See latex2html.config for this after you are done with the installation. Do not use the 'dvips -E' feature unless you have 5.62, else you will get broken images.
- `gs` *Ghostscript* (version 4.03 or later); with the ppmraw device driver, or even better pnmraw. Upgrade to 5.10 or later if you want to go sure about seldom problems with 4.03 to avoid (yet unclarified).
- The `netpbm` library of graphics utilities; `netpbm` dated 1 March 1994 is required, else part of the image creation process will fail.
  Check with: `pnmcrop` '`-version`'.

  Several of the filters in those libraries are used during the PostScript to GIF conversion.
- If you want PNG images, you need `pnmtopng` (current version is 2.31). It is not part of `netpbm` and requires `libpng-0.89c.tar.gz` and `libz` (1.0.4) (or later versions). `pnmtopng` supports transparency and interlace mode.
  Hooray!!! *Netscape Navigator* v4.04 has been reported to grok PNG images! This means that the PNG option is no longer ahead of its time!

3. **Segmentation of large documents**
   If you wish to use this feature, you will have to upgrade your LaTeX to LaTeX 2_ε . Some other hyperlinking features also require LaTeX 2_ε .
4. **Transparent inlined images**[30]
   If you dislike the white background color of the generated inlined images then you should get either the `netpbm` library (instead of the older `pbmplus`) or install the `giftrans`[31] filter by Andreas Ley `ley@rz.uni-karlsruhe.de`. LaTeX2`HTML` now supports the shareware program `giftool` (by Home Pages, Inc., version 1.0), too. It can also create interlaced GIFs.

If *Ghostscript* or the `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 37) 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*. Any browser which claims to be compatible with `HTML` 3.2 should meet this requirement.

If only a character-based browser, such as *lynx*, 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 5.2.3).

2.1.2 Installing LaTeX2HTML

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

[30] http://melmac.corp.harris.com/transparent_images.html
[31] `ftp://ftp.uni-stuttgart.de/pub/comm/infosystems/www/tools/imaging/giftrans/giftrans-1.11.1.tar.gz`

1. **Specify where *Perl* is on your system**.
   In each of the files `latex2html`, `texexpand`, `pstoimg`, `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. Copy the files to the destination directory.
   Copy the contents of the `texinputs/` directory to a place where they will be found by LaTeX, or set up your `TEXINPUTS` variable to point to that directory.

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 required by `latex2html` are correct. It will not actually install the external utilities. `install-test` asks you whether to configure for GIF or `PNG` image generation. Finally it creates the file `local.pm` which houses pathnames for the external utilities determined earlier.

   You might need to make `install-test` executable before using it.
   Use `chmod +x install-test` to do this.

   You may also need to make the files `pstogif`, `texexpand`, `configure-pstoimg` and `latex2html` executable if `install-test` fails to do it for you.

4. If you like so, copy or move the `latex2html` executable script to some location outside the `$LATEX2HTMLDIR` directory.

5. You might want to edit `latex2html.config` to reflect your needs. Read the instructions about `$ICONSERVER` carefully to make sure your `HTML` documents will be displayed right via the Web server.
   While you're at it you may want to change some of the default options in this file.
   If you do a system installation for many users, only cope with general aspects; let the user override these with `$HOME/.latex2html-init`.

Note that you *must* run `install-test` now; formerly you could manage without. If you want to reconfigure LaTeX2`HTML` for GIF/PNG image generation, or because some of the external tools changed the location, simply rerun `configure-pstoimg`.

This is usually enough for the main installation, but you may also want to do some of the following, to ensure that advanced features of LaTeX2`HTML` work correctly on your system:

- **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.

  The environment variable `TEXINPUTS` is not to be confused with the LaTeX2`HTML` 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 `$PK_GENERATION` specifies which fonts are used in the generation of mathematical equations. A value of "`0`" causes the same fonts to be used as those for the default printer. Because they were designed for a printer of much greater resolution than the screen, equations will generally appear to be of a lower quality than is otherwise possible. To cause LaTeX2HTML to dynamically generate fonts that are designed specifically for the screen, you should specify a value of "`1`" for this variable. If you do, then check to see whether your version of `dvips` supports the command-line option '`-mode`'. If it does, then also set the installation variable `$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 you have PostScript fonts available for use with LaTeX and `dvips` then you can probably ignore the above complications and simply set `$PK_GENERATION` to "`0`" and `$DVIPS_MODE` to `""` (the empty string). You must also make sure that `gs` has the locations of the fonts recorded in its `gs_fonts.ps` file. This should already be the case where *GS-Preview* is installed as the viewer for `.dvi`-files, using the PostScript fonts.

  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. This can be especially useful for multiple-segment documents.

- 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. Alternatively, a local copy of the icons can be

included within the subdirectory containing your completed HTML documents. This is most easily done using the '-local_icons' command-line switch, or by setting $LOCAL_ICONS to "1" in latex2html.config or within an initialization file, as described above.

> **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 in V96.1 that were not present in earlier 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 Section 4.10). The whole process may take quite some time, as each segment needs to be processed at least twice, to collect the cross-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: credits.tex, features.tex, hypextra.tex, licence.tex, manhtml.tex, manual.tex, overview.tex, problems.tex, support.tex and userman.tex
  - sub-directory: psfiles/ containing PostScript graphics used in the printed version of this manual
  - images of small curved arrows: up.gif, dn.gif
  - 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 supported for the manual. Please refer to text file `Changes` instead which is part of the distribution.

- **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`

- LaTeX2HTML is actively supported by the international TeX Users Group (TUG). All users are encouraged to join TUG, to keep up-to-date with the latest development in TeX, LaTeX, LaTeX2HTML and related programs. Consult the TUG Web pages at `http://www.tug.org/`.

2.2 Getting Support and More Information

A LaTeX2HTML mailing list is managed by the international TeX User Group (TUG).

This mailing used was originally established at the Argonne National Labs, where it was based for several years. (Thanks to Ian Foster and Bob Olson, and others.) Since February 1999, it has been run by TUG, thanks to Art Ogawa and Ross Moore.

To join send a message to: `latex2html-request@tug.org`
with the contents: `subscribe` .

To be removed from the list send a message to: `latex2html-request@tug.org`
with the contents: `unsubscribe` .

The mailing list also has a searchable online archive[32]. It is recommended to start with this, to become familiar with the topics actually discussed, and to search for articles related to your own interests.

An older archive[33] may still be accessible, but is no longer actively maintained.

Enjoy!

[32] `http://www.rosat.mpe-garching.mpg.de/mailing-lists/LaTeX2HTML/`
[33] `http://cbl.leeds.ac.uk/nikos/tex2html/doc/mail/mail.html`

3 Environments and Special Features

This section describes major features available for processing documents using LaTeX2HTML. Firstly the means whereby LaTeX2HTML can be configured to produce output for the different versions of HTML is discussed in Section 3.1. Following this is a description, in Section 3.2, of how to use languages other than English. The options available with the creation and reuse of images, are presented in Section 3.4, for those situations where a textual representation is inadequate or undesirable.

There are several strategies available for the presentation of mathematics according to the desired version of HTML. These are discussed in some detail, in Section 3.3. Environments such as figure, table, tabular and minipage have special features which are discussed in Section 3.5. Other supported packages are listed in Table 6.

3.1 Variation with 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 advanced HTML version with widely-accepted usage.

Currently the most advanced is HTML 4.0. However this has only recently become an officially recommended version. Not all of its features are fully implemented in popular browsers, and the level of usage is unclear. Hence the default version for LaTeX2HTML, version 98.1 remains at HTML 3.2. Further work is required before LaTeX2HTML can fully exploit the features available using HTML 4.0.

This provides support for alignment of headings, images and text (including text-flow around images), tables with separate captions and alignment of rows and columns, variable sizes and colors for text and color or patterns for the background as well as images, server-side image-maps, interactive forms, and the minimal typographic elements (bold, italic and teletype) that were supported already in HTML version 2.0 . Furthermore, HTML version 3.2 adheres to the ISO–Latin–1 (ISO–8879) character set.

Note: Although many people still use old browsers that implement only features available with HTML 2.0, this is *not* a good reason to limit translation of documents to using only these effects. Most of the translation done by LaTeX2HTML will still give acceptable results on older browsers. The deficiencies due to lack of super/subscripts, tables and some alignment effects should eventually convince such users to overcome the inertia, and update their browsers to later versions that correctly support these effects.

Sometimes it is known that the audience, for which a specific document is intended, has limited browser capabilities. Or perhaps special extended capabilities are known to be available. The LaTeX2HTML translation may be customised to suit the available functionality.

Other HTML versions and extensions supported by LaTeX2HTML are described below. See the description of the '`-html_version`' command-line option switch, on page 65.

Version 2.0 This provides only the functionality of the HTML 2.0 standard. There is little provision for aligning headings, paragraphs or images nor for super/subscripts to be generated. Images are created for tables and other environments that use <TABLE> tags with HTML 3.2; e.g. eqnarray and equation with equation numbering.

i18n (internationalised fonts) This extension (formerly known as `HTML` version 2.1) provides extensions for internationalisation. 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 '`i18n`' extension to LATEX2HTML is in preparation for when such browsers do become available, and such characters will be required in Web-accessible documents.

tables (`HTML3` model) Although `HTML` 3.2 implements tables using `<TABLE>` tags, the capabilities available to specify details of the table-layout are not as extensive as were originally proposed in the `HTML3` Table Model. This extension (formerly referred to as `HTML` version 2.2) provides the full capabilities of this model.

Note that current browsers may not correctly interpret all the features of tables constructed using this extension. Tables will be constructed, perhaps with some cells mis-aligned or without the desired merging of adjacent cells, etc. This feature was already available in many `HTML` browsers, including *Netscape Navigator* V1.2, so should be still available with later versions of these browsers.

`HTML` **3.0** This version of `HTML` was never accepted to become a recognised standard; perhaps because some of its models were "too advanced" at the time (notably the `HTML-Math` and the Table Model). The proposed `HTML` 3.0 "standard" was withdrawn and re-drafted to create the `HTML` 3.2 standard which is in current use. Standard textual formatting features, including centering, flush-right, flush-left and underlining are among the features retained.

math (`HTML3` model) This extension (formerly referred to as `HTML` version 3.1) adds support for the `HTML-Math` model, originally part of the proposed `HTML` 3.0 standard, see above. The only available browser which can display this mark-up is *Arena*. Originally developed by the World Wide Web Consortium as a test-bed browser, it is no longer supported by them.

There has been a recent proposal for a Mathematical Markup Language (`MathML`) from the W3C Math Working Group. This would suggest that the `HTML-Math` model is unlikely ever to be adopted; better things being expected in the near future using `MathML`.

See also Section 3.3 for a discussion the the mechanisms available with LATEX2HTML for handling mathematical equations and expressions.

3.2 Internationalisation

A special variable `$LANGUAGE_TITLES` in the initialisation 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 the `\today` command is presented in a format customary in that language.

Languages currently supported are finnish, french, english, german and spanish. It is trivial to add support for another language by creating a file in the `styles/` subdirectory, or by adding to the file `latex2html.config`. As a guide, here is the entry for French titles:

```
sub french_titles {
    $toc_title = "Table des mati\\'eres";
    $lof_title = "Liste des figures";
    $lot_title = "Liste des tableaux";
    $idx_title = "Index";
    $ref_title = "R\\'ef\\'erences";
    $bib_title = "R\\'ef\\'erences";
    $abs_title = "R\\'esum\\'e";
    $app_title = "Annexe";
    $pre_title = "Pr\\'eface";
    $fig_name = "Figure";
    $tab_name = "Tableau";
    $part_name = "Partie";
    $prf_name = "Preuve";
    $child_name = "Sous-sections";
    $info_title = "\\'Apropos de ce document...";
    @Month = ('', 'janvier', "f\\'evrier", 'mars', 'avril', 'mai',
             'juin', 'juillet', "ao\\^ut", 'septembre', 'octobre',
             'novembre', "d\\'ecembre");
    $GENERIC_WORDS = "a|au|aux|mais|ou|et|donc|or|ni|car|l|la|le|les"
        . "|c|ce|ces|un|une|d|de|du|des";
}
```

Notice how the backslash needs to be doubled, when a macro is needed (for accented characters, say). Also, the `$GENERIC_WORDS` are a list of short words to be excluded when filenames are specially requested to be created from section-headings. 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.

3.2.1 Alternate Font Encodings

LaTeX2HTML can interpret input using 8-bit fonts, provided it is told which font-encoding is being used. This can be done by appending an "extension" option to the '`-html_version`' command-line switch; e.g.

```
latex2html -html_version 3.2,latin2 ....  myfile.doc
```

declares that any 8-bit characters in the LaTeX source within the file `myfile.doc` are to be interpreted according to the ISO–8859–2 (ISO-Latin2) font encoding, rather than the default of ISO–8859–1 (ISO-Latin1).

Furthermore, ISO–10646 (Unicode) entities can be embedded within the output produced by LaTeX2HTML. For this a further "extension" option is appended; viz.

```
latex2html -html_version 3.2,latin2,unicode ....  myfile.doc
```

declares that the input is ISO-Latin2, but that 8-bit characters be output as the corresponding Unicode number. For example, e.g. the Polish Ł would become `Ł`. Otherwise the browser might render the character as £which is the character in the corresponding place for ISO-Latin1. The input encodings that are recognised are listed in Table 1.

extension	notes	encoding
`unicode`	(partial)	ISO–10646 (Unicode)
`latin1`	(default)	ISO–8859–1 (ISO-Latin-1)
`latin2`		ISO–8859–2 (ISO-Latin-2)
`latin3`		ISO–8859–3 (ISO-Latin-3)
`latin4`		ISO–8859–4 (ISO-Latin-4)
`latin5`		ISO–8859–9 (ISO-Latin-5)
`latin6`		ISO–8859–10 (ISO-Latin-6)

Table 1: Supported Font-encodings

If multiple extension options are requested, then later ones override earlier ones. Only in rare circumstances should it be necessary to do this. For example, if the latter encoding does not define characters in certain places, but an earlier encoding does so, and these characters occur within the source. In this case the `unicode` extension ought to be loaded also, else browsers may get quite confused about what to render.

3.2.2 Multi-lingual documents, using Images

Some multi-lingual documents can be constructed, when all the languages can be presented using characters from a single font-encoding, as discussed in the Section 3.2.1.

Another way to present multiple languages within a Web document is to create images of individual letters, words, sentences, paragraphs or even larger portions of text, which cannot be displayed within the chosen font-encoding. This is a technique that is used with IndicTEX/HTML[34], for presenting traditional Indic language scripts within Web pages. For these the LATEX source that is to be presented as an image needs special treatment using a "pre-processor". For the special styles defined in IndicTEX/HTML[35], running the preprocessor is fully automated, so that it becomes just another step within the entire image-generation process.

The technique of using images, can be used with *any* font whose glyphs can be typeset using TEX or LATEX. Using TEX's `\font` command, a macro is defined to declare the special font required; e.g. for Cyrillic characters, using the Univ. of Washington font:

```
\font\wncyr = wncyr10
```

Now use this font switch immediately surrounded by braces:

```
published by {\wncyr Rus\-ski\char26\ \char23zyk}.
```

to get:

published by Русский Язык.

3.3 Mathematics

There are various different ways in which LATEX2HTML can handle mathematical expressions and formulas:

- give a textual representation ("simple" math);

[34] http://www-texdev.mpce.mq.edu.au/l2h/indic/IndicHTML/
[35] http://www-texdev.mpce.mq.edu.au/l2h/indic/IndicHTML/

- make an image of the complete formula or expression;
- combination of textual representation and images of sub-expressions;
- SGML-like representation built using abstract "entities";
  e.g. for the HTML-Math model, or for MathML.

Which is the most appropriate normally depends on the context, or importance of the mathematics within the entire document. What LaTeX2HTML will produce depends upon

1. the version of HTML requested;
2. whether or not the special 'math' extension has been loaded;
3. whether the '-no_math' command-line option has been specified, or (equivalently) the $NO_SIMPLE_MATH variable has been set in an initialisation file.

The strategies used to translate math expressions are summarised in Table 2 for HTML 3.0+ and Table 3 for HTML 2.0.

'math'	**switch**	**strategy adopted**
not loaded	—	textual representation where possible, else image of whole expressions
not loaded	'-no_math'	always generates an image of the whole expression/environment
loaded	—	uses entities and <MATH> tags; e.g. for HTML-Math (or MathML in future)
loaded	'-no_math'	textual representation where possible, with images of sub-expressions

Table 2: Mathematics translation strategies, for HTML versions 3.0 and 3.2,
using <SUP> and <SUB> tags and <TABLE>s

Using the '-no_math' switch is best for having a consistent style used for all mathematical expressions, whether inline or in displays. The images are of especially good quality when "anti-aliasing" is being used (see page 69), provided the browser is set to have a light background colour. (When set against a gray or dark background, these images can become rather faint and hard to read.)

The final strategy in Table 2, using '-no_math' is the preferred method for good quality mathematics with HTML version 3.2 . It combines the browser's built-in fonts with the best quality images, when needed. To obtain it use the command-line option switches:

```
-no_math -html_version 3.2,math
```

This is what was used when creating the HTML version of this manual. For a more detailed discussion of processing mathematics using this strategy see the online document by the present author, entitled "Mathematics with LaTeX2HTML"[36]. Examples below show how to generate an image of a whole environment, even with these options in force.

Since the HTML 2.0 standard does not include superscripts and subscripts, via the <SUP> and <SUB> tags, the options are more limited. In this case creating images of sub-expressions is

‘math’	switch	strategy adopted
not loaded	—	textual representation where possible, else image of whole expressions
not loaded	‘-no_math’	always generates an image of the whole expression or environment
loaded	—	entities and <MATH> tags for HTML-Math
loaded	‘-no_math’	always generates an image of the whole expression or environment

Table 3: Mathematics translation strategies, for HTML version 2.0

not so attractive, since virtually the whole expression would consist of images in all but the simplest of cases.

Here are some examples of mathematical expressions and environments processed by LaTeX2HTML using different strategies. They are automatically numbered ...

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

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

$$\frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2}$$

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

The latter example uses an eqnarray environment and the `\nonumber` command to suppress the equation number on the upper line.

In the on-screen version of these equations simple alphabetic characters that are not part of fractions appear in the (italiced) text-font selected using the browser's controls. This may appear slightly different from the same symbol being used within a fraction, or other mathematical construction requiring an image to be generated. This is most apparent with the letter ‘h’ in the first equation and the subscripts at the end of the second equation.

By inserting an `\htmlimage{}` command into a math, equation or displaymath environment, a single image will be created for the whole environment. For an eqnarray environment, this will lead to having a single separate image for each of the aligned portions. The argument to `\htmlimage` need not be empty, but may contain information which is used to affect characteristics of the resulting image. An example of how this is used is given below, and a fuller discussion of the allowable options is given in Section 3.4.

Scale-factors for Mathematics. When an image is to be made of a mathematical formula or expression, it is generally made at a larger size than is normally required on a printed page. This is to compensate for the reduced resolution of a computer screen compared with laser-print. The amount of this scaling is given by the value of a configuration variable `$MATH_SCALE_FACTOR`, by default set to 1.6 in `latex2html.config`. A further variable `$DISP_SCALE_FACTOR` is used with ‘displayed math’ equations and formulas. This value multiplies the `$MATH_SCALE_FACTOR` to give the actual scaling to be used. The main purpose of this extra scaling is to allow some clarity in super/subscripts etc.

[36] http://www-texdev.mpce.mq.edu.au/l2h/mathdocs/

Anti-aliased Images. Figure 1 shows the same equations as previously, this time as images of the complete contents of the equation environment, and complete aligned parts of rows in an eqnarray. These are images, as they would appear if the HTML page were to be printed from the browser. A scaling of 60% has been applied to counteract the combined effects of the $MATH_SCALE_FACTOR of 1.4 and $DISP_SCALE_FACTOR of 1.2, used for the HTML version of this manual. For a comparison, the second group of images use anti-aliasing effects, whereas the first image does not; a 600 dpi printing is probably necessary to appreciate the difference in quality. Compare these images with those in Section 3.4.3.

Note: To generate anti-aliased images using *Ghostscript* requires version 4.03 or later.

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

$$\frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2}$$
$$+\frac{\Phi_{l,m,n+1} - 2\Phi_{l,m,n} + \Phi_{l,m,n-1}}{h^2} = -I_{l,m,n}(v) \; . \tag{4}$$

Figure 1: Images of equation displays, at normal screen resolution

These images of the whole environment were created using the \htmlimage command, to suppress the extended parsing that usually occurs when the 'math' extension is loaded; viz.

```
\begin{equation}
\htmlimage{no_antialias}
\Phi_{l+1,m,n} = \Bigl(\Phi+h\frac{\partial\Phi}{\partial x} +
...
\end{equation}
%
\begin{eqnarray}
\htmlimage{}
\frac{\Phi_{l+1,m,n}-2\Phi_{l,m,n}+\Phi_{l-1,m,n}}{h^{2}} +
...
\end{eqnarray}
```

Further aspects of the options available when generating images are discussed in the next section, in particular with regard to the quality of printed images.

The \mbox command. Another way to force an image to be created of a mathematical expression, when global settings are not such as to do this anyway, is via the \mbox command having math delimiters within its argument.

Normally \mbox is used to set a piece of ordinary text within a mathematics environment. It is not usual to have math delimiters $...$ or \(...\) within the argument of an \mbox. Whereas earlier versions of LaTeX2HTML simply ignored the \mbox command (treating its argument as normal text), the presence of such delimiters now results in an image being generated of the *entire contents* of the \mbox. It is not necessary for there to be any actual mathematics inside the `\mbox`'s contents; e.g. `\mbox{...some text...${}$}` will cause an image to be created of the given text.

The `\parbox` command. The `\parbox[<align>]{<width>}{<text>}` command also generates an image of its contents, except when used within a tabular environment, or other similar table-making environment. Here the important aspect is the width specified for the given piece of text, and any special line-breaks or alignments that this may imply. Hence to get the best effect, LaTeX is used to typeset the complete `\parbox`, with its specified width, alignment and contents, resulting in an image.

The heqn package. If you need HTML 2.0 compatible Web pages, and have a document with a great many displayed equations, then you might try using the heqn 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.

The `\nonumber` command is recognised in each line of the equation array, to suppress the equation number. 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.

Using HTML Version 3.2 the heqn package is quite redundant, since equation numbers are placed in a separate `<TABLE>` cell to the mathematical expressions themselves. It is *not* required and should *not* be requested, since this will override some of the improved functionality already available.

3.4 Figures and 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 inline math expressions, special accents and any other LaTeX command which causes an image to be generated; while . . .
- "figures" applies to image-generating LaTeX environments (e.g. makeimage, figure, table (with HTML 2.0), and displayed math environments when required to generate images, 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.

A further variable `$DISP_SCALE_FACTOR` is used with 'displayed math' equations and formulas; this value multiplies the `$MATH_SCALE_FACTOR` to give the actual scaling used. With the improved clarity of anti-aliased images, a scaling of 1.6 may be a little excessive for inline images. Accordingly this manual actually uses values of 1.4 and 1.2 respectively, for `$MATH_SCALE_FACTOR` and `$DISP_SCALE_FACTOR`. These go well with the browser's text-font set at 14 pt. The next larger size of 17 pt is then used for the `<LARGE>` tags in displayed equations.

A further variable `$EXTRA_IMAGE_SCALE` allows images to be created at a larger size than intended for display. The browser itself scales them down to the intended size, but has the extra information available for a better quality print. This feature is also available with single images. It is discussed, with examples, in Section 3.4.3.

`\htmlimage{<`*`options`*`>}` For finer control, several parameters affecting the conversion of a single image can be controlled with the command `\htmlimage`, which is defined in `html.sty`. With version V97.1 use of this command has been extended to allow it to control whether an image is generated or not for some environments, as well as specifying effects to be used when creating this image.

If an `\htmlimage` command appears within any environment for which creating an image is a possible strategy (though not usual, due to loading of extensions, say), then an image will indeed be created. Any effects requested in the <*options*> argument will be used. Having empty <*options*> still causes the image to be generated.

This ability has been used within this manual, for example with the mathematics images in Figure 1.

The <*options*> argument is a string separated by commas. Allowable options are:

- `scale=<`*`scale-factor`*`>`
  allows control over the size of the final image.

- `external`
  will cause the image not to be inlined; instead it will be accessible via a hyperlink.

- `thumbnail=<`*`scale-factor`*`>`
  will cause a small inlined image to be placed in the caption. The size of the thumbnail depends on the <*scale-factor*>, as a factor of the 'natural size' of the image, ignoring any `$FIGURE_SCALE_FACTOR` or `$MATH_SCALE_FACTOR`, etc. which may be applicable to the full-sized version of the image. Use of the '`thumbnail=`' option implies the '`external`' option.

- `map=<`*`server-side image-map URL`*`>`
  specifies that the image is to be made into an active image-map. (See Section 4.9 for more information.)

- `usemap=<`*`client-side image-map URL`*`>` same as previous item, but with the image-map processed by the client. (See Section 4.9 for more information.)

- `flip=<`*`flip_option`*`>`
  specifies a change of orientation of the electronic image relative to the printed version. The <*flip_option*> is any single command recognised by the `pnmflip` graphics utility. 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.

- `align=<alignment>`
  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. The default alignment is '`bottom`'.

- `transparent`, `no_transparent` or `notransparent`
  specify that a transparent background should (not) be used with this image, regardless of the normal behaviour for similar images.

- `antialias`, `no_antialias` or `noantialias`
  specify that anti-aliasing should (not) be used with this image, regardless of the normal behaviour for similar images.

- `extrascale=<scale-factor>`
  is used mainly used with a `<scale-factor>` of 1.5 or 2, when it is important to get printed versions of the completed `HTML` pages. The image is created scaled by the amount specified, but it is embedded in the `HTML` page with attributes to the `<IMG>` of `HEIGHT=...` and `WIDTH=...`, indicating the *unscaled* size. A browser is supposed to display the image at the requested size by scaling the actual image to fit, effectively imposing its own anti-aliasing. Some examples of this effect are show later, in Section 3.4.3. This effect can be applied to all images in a document by setting the `$EXTRA_IMAGE_SCALE` variable. However it may be desirable to also turn off "anti-aliasing", as these effects serve similar purposes but need not work well together. Furthermore different browsers may give results of different quality. It may be necessary to experiment a little, in order to find the combination that works best at your site.

- `height=<dimen>` and `width=<dimen>`
  are used to specify exactly the size to be occupied by the image on the `HTML` page. The value(s) given this way overrides the natural size of the image and forces the browser to shrink or stretch the image to fit the specified size. The `<dimen>` can be given as either (i) a number (of points); or (ii) with any of the units of cm, mm, in, pt; or (iii) fraction of `\hsize` or `\textwidth`, to become a percentage of the browser window's width, or of `\vsize` or `\textheight` for a percentage height.

  Note: images whose sizes are modified in this way may not be acceptable for image-recycling, (see page 3.4.2). Instead they may need to be generated afresh on each run of LaTeX2`HTML` through the same source document.

In order to be effective the `\htmlimage` command and its options must be placed *inside the environment* on which it will operate. Environments for alignment and changing the font size do not generate images of their contents. Any `\htmlimage` command may affect the surrounding environment instead; e.g. within a table or figure environment, but does not apply to a minipage.

When the `\htmlimage` command occurs in an inappropriate place, the following message is printed among the warnings at the end of processing. The actual command is shown, with its argument; also the environment name and identifying number, if there is one.

```
The command "\htmlimage" is only effective inside an environment
which may generate an image (e.g. "{figure}", "{equation}")
 center92: \htmlimage{ ... }
```

3.4.1 An Embedded Image Example

The effect of the LaTeX commands below can be seen in the thumbnail sketch of Figure 2. A 5 pt border has also been added around the thumbnail, using `\htmlborder` command; this gives a pseudo-3D effect in some browsers.

```
\begin{figure}
    \htmlimage{thumbnail=0.5}
    \htmlborder{5}
    \centering \includegraphics[width=5in]{psfiles/figure.ps}
    \latex{\addtocounter{footnote}{-1}}
    \caption{A sample figure showing part of a page generated by
       \latextohtml{} containing a customised navigation panel
       (from the \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}}
```

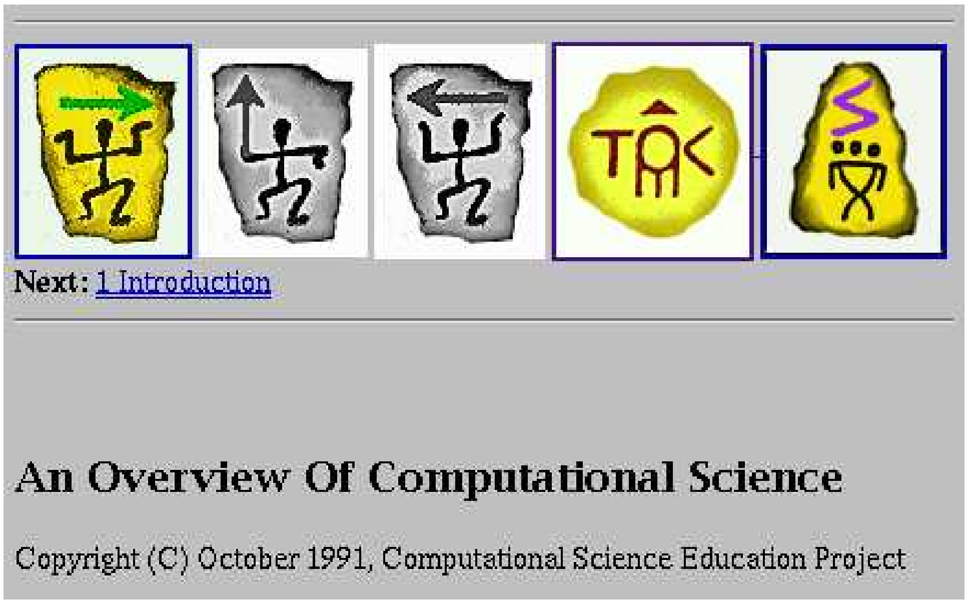

Figure 2: A sample figure showing part of a page generated by LaTeX2HTML containing a customised navigation panel (from the CSEP project[37]).

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}`.

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

3.4.2 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 LaTeX2`HTML`. Without this ability, every instance of an image would have to be regenerated each time even the slightest change were made to the document.

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, when '`-html_version 2.0`' has been specified; 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 not part of the image itself.Similarly when `HTML` 3.2 code is being produced, equation numbers are no longer part of the image. Instead they are placed in a separate cell of a `<TABLE>`. So most images of mathematical formulas can be reused also.

3.4.3 Quality of Printed Images

$$\Phi_{l+1,m,n} = \left(\Phi + h\frac{\partial\Phi}{\partial x} + \frac{1}{2}h^2\frac{\partial^2\Phi}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3\Phi}{\partial x^3} + \ldots\right)_{l,m,n} \tag{5}$$

$$\frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2} + \frac{\Phi_{l,m,n+1} - 2\Phi_{l,m,n} + \Phi_{l,m,n-1}}{h^2} = -I_{l,m,n}(v) \tag{6}$$

Figure 3: Displayed math environments with *extra-scale* of 1.5

Since it is often desirable to get a good quality print on paper directly from the browser, Figure 3 shows the same equations as on page 20. This time the '`extrascale=1.5`' option has been used. This value of 1.5 means that more than twice the number of pixels are available, for a cost of approximately 1.7 times the disk-space[38]. On-screen these images appear slightly blurred or indistinct. However there can be marked improvement in the print quality, when printed from some browsers; others may show no improvement at all. The "anti-aliasing" helps on-screen. In the printed version jagged edges are indeed softened, but leave an overall fuzziness.

Figure 4 shows the same equations yet again; this time with '`extrascale=2.0`'. Now there are 4 times the pixels at a cost of roughly 2.45 times the disk space. Compared with the previous images (having 1.5 times extra-scaling), there is little difference in the on-screen images. Printing at 300 dpi shows only a marginal improvement; but at 600 dpi the results

[38] This figure varies with the graphics format used, and the complexity of the actual image.

are most satisfying, especially when scaled to be comparable with normal 10 pt type, as here.

$$\Phi_{l+1,m,n} = \Big(\Phi + h\frac{\partial\Phi}{\partial x} + \frac{1}{2}h^2\frac{\partial^2\Phi}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3\Phi}{\partial x^3} + \dots\Big)_{l,m,n} \tag{7}$$

$$\frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2} + \frac{\Phi_{l,m,n+1} - 2\Phi_{l,m,n} + \Phi_{l,m,n-1}}{h^2} = -I_{l,m,n}(v)\ . \tag{8}$$

Figure 4: Displayed math environments with *extra-scale* of 2.0

3.5 Figures, Tables and Arbitrary Images

This section is to explain how the translator handles figures, tables and other environments. Compare the paper with the online version.

When the common version of `HTML` was only 2.0, then almost all complicated environments were represented using images. However with `HTML` 3.2, there is scope for sensible layout of tables, and proper facilities for associating a caption with a figure or table. To take advantage of this, the figure environment now has its contents placed within `<TABLE>` tags; any caption is placed as its `<CAPTION>`.

For consistency with former practice, the contents of the figure environment are usually represented by generating an image. This is frequently exactly what is required; but not always. On page 47 it is described how to use the makeimage environment, defined in the `html.sty` package, to determine just which parts (if any) of a figure environment's contents should be made into images, the remainder being treated as ordinary text, etc.

table and tabular environments. Similarly the makeimage environment can be used within a table, though usually this is used with a tabular or other table-making environment, such as tabbing or longtable or supertabular. Here is a simple example, from the LaTeX 'blue book' .

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

Table 4: A sample table taken from [1]

Table 5 is a screen-shot of how the resulting table appears on-screen, using a typical browser supporting `HTML` 3.2. Here it is scaled down by 70% to compensate for the 14 pt fonts being used when the screen-shot was taken.

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

Table 5: Alternate view of the table from [1]

minipage environments. The special feature of minipage environments is in the way `\footnote` and `\footnotemark` commands are handled. These are numbered separately from the rest of the footnotes throughout the document, and the notes themselves are collected together to be displayed at the end of the minipage's contents.

Variable	**Meaning**
none	none
Jacobi	m-step Jacobi iteration[a]
SSOR	m-step SSOR iteration[1]
IC	Incomplete Cholesky factorization[b]
ILU	Incomplete LU factorization[2]

[a] one footnote
[b] another footnote

The code used for this example was as follows[39]

```
\begin{minipage}{.9\textwidth}
\renewcommand{\thempfootnote}{\alph{mpfootnote}}
\begin{tabular}{|l|l|} \hline
\textbf{Variable} & \textbf{Meaning} \\ \hline
none      & none                  \\
Jacobi    & $m$-step Jacobi iteration\footnote[1]{one footnote} \\
SSOR      & $m$-step SSOR iteration\footnotemark[1] \\
IC        & Incomplete Cholesky factorization\footnote[2]{another footnote} \\
ILU       & Incomplete LU factorization\footnotemark[2] \\ \hline
\end{tabular}
\end{minipage}
```

Warning: With some figures, especially when containing graphics imported using `\includegraphics` or other special macros, the background color may come out as a shade of grey, rather than white or transparent. This is due to a setting designed to enhance anti-aliasing of text within images; e.g. for mathematics. To alleviate this possible problem, the '`-white`' command-line option can be used, to ensure a white background for images of figure environments. Alternatively, set the `$WHITE_BACKGROUND` variable (see section 5.2.3).

[39] Thanks to John Turner `turner@lanl.gov` for this example, which was used in developing code to handle minipage environments correctly.

3.6 Document Classes and Options

In general the standard LaTeX document-classes: `article`, `report`, `book`, `letter`, `slides` are translated by LaTeX2HTML in the same way. Currently the only real difference is with the display of section-numbering, when the '`-show_section_numbers`' switch is used, and when numbering of theorem-like environments is linked to section-numbering.

These differences are achieved using a mechanism that automatically loads a file: `article.perl`, `report.perl`, `book.perl`, `letter.perl`, `slides.perl` according to the requested document-class. These files contain *Perl* code and are located in the `styles/` directory. If a file of the same name exists in the working directory, this will be loaded instead.

Typically such files <*class*>`.perl` contain code to define subroutines or sets values for variables that will affect how certain translations are performed. There can be code that is executed only when specific class-options are specified along with the chosen document-class. For example, the `foils.perl` implementation of FoilTeX's foils class defines code create a new sub-section for each 'foil'. It also has code which allows LaTeX2HTML to ignore those of FoilTeX's special formatting commands that have no relevance when constructing an HTML page.

Any options given on the `\documentclass` or `\documentstyle` line may also cause a file containing *Perl* code to be loaded. Such a file is named <*option*>`.perl` for the appropriate <*option*>. When such a file exists, in the local directory or in the `styles/` directory, it typically contains *Perl* code to define subroutines or set values for variables that will affect how certain translations are performed. There can be code that is executed only for specific document-classes.

Since the files for class-options are loaded after those for the document-class, it is possible for the <*option*>`.perl` file to contain code that overrides settings made within the document-class file.

If a file named <*class*>_<*option*>`.perl` happens to exist for a given combination of document-class <*class*> and class-option <*option*>, then this will be loaded. When such a file exists, reading and executing its contents is done, rather than executing any <*class*>_<*option*> specific information that may be contained in <*class*>`.perl` or <*option*>`.perl` .

Currently there are no special option or class-option files provided with the LaTeX2HTML distribution. It is hoped that users will identify ways that specific features can be improved or adapted to specific classes of documents, and will write such files themselves, perhaps submitting them for general distribution.

Note: This mechanism for handling code specific to different document classes and class-options is more general than that employed by LaTeX 2_ε. New options can be defined for document-classes generally, or for specific classes, without the need to have corresponding `.sty` or `.clo` files. LaTeX simply notes the existence of unsupported options—processing is not interrupted.

3.7 Packages and Style-Files

Similar to the document-class mechanism described in Section3.6, LaTeX2HTML provides a mechanism whereby the code to translate specific packages and 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, either by

- a `\usepackage` command of LaTeX 2ε;
- an option to the `\documentstyle` command of LaTeX 2.09 ;
- an explicit `\input` or `\include` command;

the translator looks for a corresponding `.perl` file having the same file-name prefix; e.g. the file `$LATEX2HTMLDIR/styles/german.perl`. If such a `.perl` file is found, then its code will be incorporated with the main script, to be used as required.

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 to support the styles listed in Table 6. These provide good examples of how you can create further extensions to LaTeX2HTML.

Table 6: Supported LaTeX2HTML packages and style-files.

`.perl` file	**Description**
alltt	Supports the LaTeX 2ε's alltt package.
amsfonts	provides recognition of the special AMS font symbols.
amsmath	same as `amstex.perl`.
amssymb	same as `amsfonts.perl`.
amstex	Supports much of the AMS-LaTeX package (not yet complete).
babel	Interface to `german.perl` via the babel package.
changebar	Provides rudimentary change-bar support.
chemsym	defines the standard atomic symbols.
color	Causes colored text to be processed as ordinary text by LaTeX2HTML.
colordvi	supports the Crayola colors.
enumerate	supports structured labels for enumerate environments.
epsbox	Processes embedded figures not enclosed in a figure environment.
epsfig	Processes embedded figures not enclosed in a figure environment.
finnish	Support for the Finnish language.
floatfig	Processes floating figures.
floatflt	Processes floating figures and tables.
foils	Supports FoilTeX system.
frames	Provides separate frames for navigation and footnotes.
francais	Support for the French language, same as `french.perl`.
french	Support for the French language.
german	Support for the German language.
germanb	Support for the German language, same as `german.perl`.
graphics	Supports commands in the graphics package.
graphicx	Supports the alternate syntax of graphics commands.
harvard	Supports the harvard style of citation (same as fnnharvard.perl).
heqn	Alters the way displayed equations are processed.
hthtml	gives an alternative syntax for specifying hyperlinks, etc.
htmllist	Provides support for fancy lists.
justify	supports paragraph alignment—no longer needed.
latexsym	supports the LaTeX symbol font.
lgrind	macros for nice layout of computer program code.
longtable	supports use of long tables, as a single table.
makeidx	provides more sophisticated indexing.

Table 6: Supported LaTeX2HTML packages and style-files.

multicol	suppresses requests for multi-columns.
natbib	Supports many different styles for citations and bibliographies.
nharvard	Supports harvard-style citations, using natbib.
seminar	for creation of overhead-presentation slides.
spanish	Support for the Spanish language.
supertabular	supports use super-tables, as an ordinary table.
texdefs	Supports some raw TeX commands.
verbatim	Supports verbatim input of files.
verbatimfiles	Supports verbatim input of files, also with line-numbering.
wrapfig	Supports wrapped figures.
xspace	Supports use of the `xspace` package and `\xspace` command.
xy	Supports use of the XY-pic graphics package.

The problem however, is that writing such extensions requires an understanding of *Perl* programming and of the way the processing in LaTeX2HTML is organised. Interfaces that are more "user-friendly" are being investigated. Some of the techniques currently used are explained in Section 5.3.

3.7.1 Fancy List-Markers

An optional style-file `htmllist.sty` has been provided which produces fancier lists in the electronic version of the documentsuch as this. 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 filename prefix for the item-mark image can be given as an optional parameter; see example below. The images distributed with LaTeX2HTML for this purpose are listed with the description of the `\htmlitemmark` command, which provides an alternative means of choosing the item-mark, and allows the image to be changed for different items in the list.

The mark is determined by the `\htmlitemmark{<item-mark>}` command. This command accepts either a mnemonic name for the `<item-mark>`, from a list of icons established at installation, or the URL of a mark not in the installation list. The command `\htmlitemmark` must be used *inside* the htmllist environment in order to be effective, and 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}[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 2ε 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 *The LaTeX Companion*[2].

3.7.2 Support for FoilTeX

The FoilTeX system presents some additional problems for LaTeX2HTML:

- It has additional commands like `\foilhead` and `\rotatefoilhead`, that roughly correspond to sectioning commands,
- The images are produced at the sizes suitable for large screen presentation, but not for the HTML.

The package `foils.perl` deals with these problems. It treats foils as starred subsections and ignores FoilTeX-specific commands that have no meaning for HTML, like `\LogoOn`. The header `\documentclass[`+*options*`]{foils}` in the `images.tex` file is substituted by the header `\documentclass[$FOILOPTIONS]{$FOILCLASS}`, where the variables `$FOILOPTIONS` and `$FOILCLASS` can be set in the configuration file (by default they are `'10pt'` and `'article'` correspondingly). A further variable `$FOILHEADLEVEL` holds the level of sectioning at which a 'foil' is to correspond; the default level is 4 (sub-section).

The LaTeX style file `foilhtml.sty` in the `texinputs/` directory provides some additional features for FoilTeX. It implements structural markup commands like `\section`, `\tableofcontents` for foils. See the directory `docs/foilhtml/` for the details.

3.7.3 Indicating Differences between Document Versions

LaTeX2HTML supports the LaTeX 2ε `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. Thicker bars indicate changes introduced with version V97.1 , while thinner bars indicate earlier additions since V96.1 . Within the HTML version the change-bars clearly indicate the different revisions with explicit numbering.Within the HTML version, the graphic icons representing the changebars can be followed by some text indicating the new version. This is used repeatedly throughout the online version of this manual. It is achieved using the command `\cbversion{`*<version>*`}`, immediately following the `\begin{changebar}`. This sets a variable `$cb_version` to be used both at the beginning and end of the environment. The value of this variable is retained, to be used with other changebar environments, unless changed explicitly by another occurrence of `$cb_version`.

Warning: LaTeX2HTML will not correctly process changebar environments that contain sectioning commands, even when the (sub)sections or (sub)paragraphs are to occur on the same HTML page. If this is required, use a separate changebar environment within each (sub)section or (sub)paragraph.

3.8 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 handled correctly, by applying \emph to the text of the generated hyperlink.

|<*style*> : where <*style*> is the name of LaTeX style-changing command, without the initial '\'; e.g. 'emph', 'textbf', 'textit', etc. The corresponding LaTeX command is applied to the text of the generated hyperlink.

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 section 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, viewing the entire document as 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 initialisation file; provided, of course, that the document loads the makeidx package.

3.8.1 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 96.1f by Ross Moore, 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:

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

  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:

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

  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.

4 Hypertext Extensions to LaTeX

This section describes how you can define hypertext entries in your HTML documents from within your LaTeX source, as well as other effects available in HTML for which there need be no direct LaTeX analog for a printed document. These are implemented as new LaTeX commands which have special meaning during the translation by LaTeX2HTML into HTML, but are mostly ignored when processed by LaTeX.

The new commands described in the sections below are defined mainly in the html package, with LaTeX definitions in the file `html.sty`, which is part of the LaTeX2HTML distribution. It *must be included* in any LaTeX document using these features, by one of the following methods:

- including `html` as an optional argument to `\documentstyle` in LaTeX 2.09 ;
- including `html` in a LaTeX 2_ε `\usepackage` command.

It is *not sufficient* to load the style file via an `\input` or `\include` command, such as `\input html.sty`. This will load the required definitions for LaTeX, but will not load the `html.perl` package file for LaTeX2HTML.

Warning: Some of these features, but not all, are also available with LaTeX 2.09. Users of LaTeX2HTML are strongly advised to upgrade their LaTeX installations to LaTeX 2_ε.

Several new environments are defined, in particular for specifying large (or small) sections of the text which are appropriate to only one version of the document—either the HTML or the LaTeX typeset version. Their use is discussed in Sections 4.2 and 4.4.

`\begin{rawhtml}` for including raw HTML tags and SGML-like markup.

`\begin{htmlonly}` for material intended for the HTML pages only.

`\begin{latexonly}` for material intended for the LaTeX version only. Note that any macro-definitions or changes to counter-values are local to within this environment.

`%begin{latexonly}` for material intended for the LaTeX version only. Macro-definitions and changes to counter-values are retained outside of this (pseudo-)environment.

`\begin{imagesonly}` for material intended to be used in the `images.tex` file only.

`\begin{comment}` for user-comments only, currently ignored in both the HTML and LaTeX versions. (To put HTML comments into the HTML files, use the rawhtml environment.)

`\begin{makeimage}` creates an image of its contents, as typeset by LaTeX. This is also used to prevent an image being made of the complete contents of a figure environment, allowing more natural processing.

`\begin{htmllist}` defined in `htmllist.sty` and `htmllist.perl`, this produces coloured balls tagging the items in a descriptive list, as used throughout the HTML version of this manual.

Warning: When using these environments it is important that the closing delimiter, `\end{htmlonly}` say, occurs on a line by itself with no preceding spaces, `<tab>`s or any other characters. (Otherwise LaTeX will *not* recognise the intended end of the environment when processing for the `.dvi` version.) Similarly there should be nothing on the same line *after* the opening environment delimiter, `\begin{htmlonly}` say.

The following commands are defined for LaTeX in `html.sty`. Corresponding *Perl* implementations are either in `html.perl` or in the `latex2html` script itself.

`\latextohtml` expands to the name LaTeX2HTML, of this translator;

`\htmladdnormallink` creates a (perhaps named) textual hyperlink to a specified <*URL*>;

`\htmladdnormallinkfoot` same as `\htmladdnormallink`, but LaTeX also prints the <*URL*> in a footnote;

`\htmladdimg` places an image (perhaps aligned) on the HTML page; ignored by LaTeX.

`\hyperref` creates a textual hyperlink to where a `\label` command occurred within the same document. This is the recommended substitute for LaTeX's `\ref` command.

`\htmlref` creates a textual hyperlink to the place where a `\label` command occurred; no reference is printed in the LaTeX version.

`\hypercite` creates a textual hyperlink to the bibliography page where citation details are shown. This is the recommended substitute for LaTeX's `\cite` command.

`\htmlcite` creates a textual hyperlink to the bibliography page where citation details are shown; no citation marker is printed in the LaTeX version.

`\externalref` creates a textual hyperlink to where a `\label` command occurred within a different document that has also been processed by LaTeX2HTML; ignored in LaTeX.

`\externalcite` creates a textual hyperlink to where a reference occurs in a bibliography page from a different document that has also been processed by LaTeX2HTML; ignored in LaTeX.

`\externallabels` allows hypertext links to a different document; ignored in LaTeX.

The following commands, also defined for LaTeX in `html.sty`, are normally used only when creating segmented documents, see Section 4.10.

`\segment` directs that an `\input` file <*file*> should be regarded as a separate "segment" of a larger LaTeX2HTML document. In LaTeX the file is input as usual, after counter values have first been written to a file, named <*file*>`.ptr`.

`\startdocument` tells LaTeX2HTML where the end of the preamble occurs for a document segment; ignored in LaTeX. (A segment cannot have a `\begin{document}` command, unless it is shielded from LaTeX within an htmlonly environment.)

`\internal` reads internal information from another document, so that symbolic references can be treated as if part of the current document; ignored in LaTeX.

`\htmlhead` places a sectional heading on a HTML page; used mainly with the document segmentation feature. It is ignored in LaTeX.

`\htmlnohead` suppresses the section-heading for a document segment; ignored in LaTeX.

`\segmentcolor` read from the `.ptr` file, this sets the text color for a document segment; ignored in LaTeX.

`\segmentpagecolor` read from the `.ptr` file, this sets the background color for a document segment; ignored in LaTeX.

The following commands are shorthand forms for some of the "conditional" environments listed above.

`\html` for putting small pieces of text into the HTML version only;

`\latex` for putting small pieces of text into the LaTeX version only;

`\latexhtml` puts one piece of text into the LaTeX version, another into the HTML version.

The following commands implement effects on the HTML pages for which there is no direct LaTeX counterpart. Most of these commands are discussed in detail in Section 4.8.

`\HTMLcode` a general command for placing raw HTML tags, with attributes and contents; tags and attributes are ignored in LaTeX, but not the contents. (See Section 43.)

`\htmlrule` places a (perhaps styled) horizontal line on the HTML page; ignored in LaTeX.

`\strikeout` places text between `<STRIKE>`...`</STRIKE>` tags; ignored in LaTeX.

`\htmlimage` used for fine control over the size of individual images, and other graphics effects (e.g. making a 'thumbnail' version); ignored in LaTeX. (See page 22 for details.)

`\htmlborder` places a border around the contents of an environment, but placing the environment as a cell inside a `<TABLE>`; ignored in LaTeX.

`\tableofchildlinks` determines where the table of childlinks should be placed on the HTML page; ignored in LaTeX.

`\htmlinfo` determines where the "About this document..." information should be placed; ignored in LaTeX.

`\htmladdtonavigation` appends a button to the navigation panels; ignored in LaTeX.

`\bodytext` allows the contents of the `<BODY...>` tag to be set explicitly for the current and subsequent HTML pages; ignored in LaTeX.

`\htmlbody` allows an attribute to be added or changed within the `<BODY...>` tag of HTML; ignored in LaTeX.

`\htmlbase` Allows a URL to be specified within the `<BASE...>` tag for all the HTML pages produced; ignored in LaTeX.

`\htmltracing{<level>}` specifies that extra tracing messages be generated, according to the `<level>`; ignored in LaTeX. (See page 71 for levels of verbosity.)

`\htmltracenv{<level>}` same as `\htmltracing` except that this command is evaluated in sequence with environments; ignored in LaTeX. (See also page 71.)

`\HTMLset` programmer's device, allowing an arbitrary *Perl* variable to be set or changed dynamically during the LaTeX2HTML processing; ignored in LaTeX.

\HTMLsetenv Same as the preceding \HTMLset command, except that this one is processed in order, as if it were an environment; ignored in LaTeX.

Most of the new environments listed above can also be used with delimiter macros \<*env-name*>...\end<*env-name*>. This alternative style, which is common with AMS-TeX, is discouraged for general LaTeX usage (even by the AMS itself) in favour of the usual \begin{<*env-name*>}...\end{<*env-name*>} markup notation. (Safety features that are available with the usual \begin...\end mechanism may not always work in the best way with this alternative style of environment delimiter. These comments apply to both the LaTeX and LaTeX2HTML processing.)

\rawhtml...\endrawhtml old AMS-style variant of rawhtml environment.

\htmlonly...\endhtmlonly old AMS-style variant of htmlonly environment.

\latexonly...\endlatexonly old AMS-style variant of latexonly environment.

\imagesonly...\endimagesonly old AMS-style variant of imagesonly environment.

\comment...\endcomment old AMS-style variant of comment environment.

Warning: These 'pseudo'-environments are not as reliable as their LaTeX counterparts. In particular, the \begin<*env-name*> and \end<*env-name*> commands should appear on lines by themselves, preferably with no preceding spaces or <*tab*> characters. This requirement is analogous to the warning at the bottom of page 37 for conditional environments.

4.1 Hyper-links in LaTeX

Arbitrary hypertext references are created using the \htmladdnormallink and \htmladdimg commands. These have syntax:

```
\htmladdnormallink{<text>}{<URL>}
\htmladdnormallink[<name>]{<text>}{<URL>}

\htmladdimg{<URL>}
\htmladdimg[<align>]<URL>}

\htmladdnormallinkfoot{<text>}{<URL>}
\htmladdnormallinkfoot[<name>]{<text>}{<URL>}
```

\htmladdnormallink The \htmladdnormallink 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}
```

The optional argument to \htmladdnormallink allows a name to be specified for the place in the document where the hyperlink occurs. This is done via the NAME="<*name*>" attribute for the <A...> anchor tag in HTML. Such a name can be used as the target for a hyperlink using the \htmlref command, described in Section 4.5.

`\htmladdimg` 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. The optional argument to `\htmladdimg` allows an alignment for the image to be given: `center`, `right` or `left`. In the latter cases, the image is bound to the specified side of the browser's window. Subsequent text paragraphs 'flow around' the other side of the image.

In fact any valid set of "attributes" for the `<IMG>` tag in `HTML` can be specified as the optional `<`*align*`>` parameter. In particular the `WIDTH`, `HEIGHT` and `BORDER` attributes can be set, perhaps overriding the natural size of the image.

`\htmladdnormallinkfoot` The `\htmladdnormallinkfoot` command takes the same arguments, and when generating `HTML` has the same effect, as `\htmladdnormallink`. However when processed by LaTeX it places the URL as a footnote.

Warning: The tilde (~) character is commonly used within hyperlink URLs. It is a quirk of TeX and 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 and Comments

LaTeX2`HTML` provides the ability to include raw `HTML` tags and text within the `HTML` version of a document, without requiring corresponding material for the LaTeX typeset version. This ability can be used to

- include `HTML` markup for effects that have no corresponding concept within a LaTeX typeset document (see the following example)
- take advantage of new `HTML` facilities as soon as they become available, and there are browsers capable of displaying them.
- include arbitrary `SGML`-like markup, for use with special browsers that know how to sensibly handle the resulting files.

`\begin{rawhtml}` The simplest way to include raw `HTML` tags and/or text is by using the rawhtml environment. (An alternative way is to use the `\HTML` command, described in Section 4.3, which allows macros to be expanded to give the required tags, attributes and contents.)
Note the warning on page 37 concerning how the environment delimiters should be used in the LaTeX source code.

A particularly good use of the rawhtml environment 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 NAME="other_wp">

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

The result is shown in Figure 5.

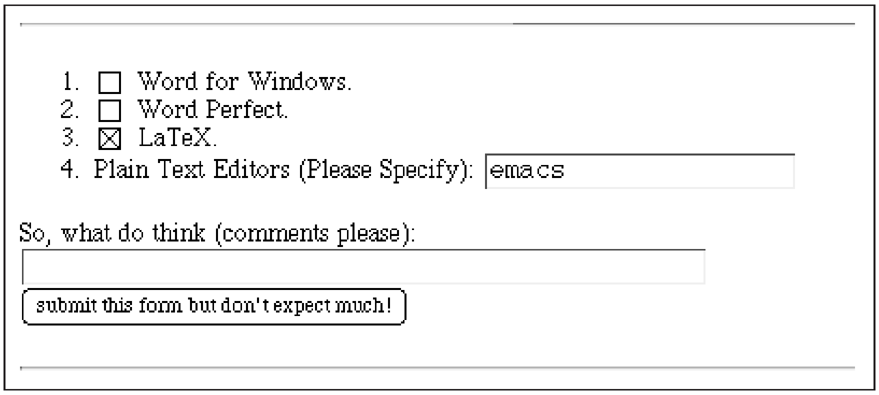

Figure 5: An electronic form. In the online version the form would be active.

`\beginrawhtml...\endrawhtml` This is an alternative way to specify a chunk of raw HTML code, using the old AMS-style of delimiting environments. Use of this style is discouraged; the rawhtml environment is preferred.

`\begin{comment}` This environment is simple for the convenience of "commenting-out" large sections of source code. The contents of this environment is completely ignored, both in the LaTeX and HTML versions. Such an environment is already used in AMS-LaTeX, and perhaps with other packages. It is defined here for its general utility.
To insert SGML-style comments into the HTML files, use the rawhtml environment as follows.

```
\begin{rawhtml}
<!--  this text is treated as a comment
      perhaps extending over several lines
-->
\end{rawhtml}
```

Note the warning on page 37 concerning how the environment delimiters should be used in the LaTeX source code.

`\comment...\endcomment` This is an alternative way to specify a chunk of material intended to be ignored in both the LaTeX and HTML versions, using the old AMS-style of delimiting environments. Use of this style (though convenient for typing) is discouraged, since it is not as reliable as using the comment environment.

4.3 Arbitrary Tags and Attributes

For version 97.1 of LaTeX2HTML there is a new command which provides an extremely flexible way to include HTML 3.2 tags, along with any values for the "attributes" of that tag, if desired.

```
\HTMLcode[<attribs>]{<tag>}
\HTMLcode[<attribs>]{<tag>}{<contents>}
```

When the *<tag>* also needs a closing tag (e.g <I>...</I>) the *<contents>* *must* be given, enclosed in braces. Both the opening and closing tags then will be placed correctly.

> **Warning:** In version 97.1 this command was actually called \HTML. However style files may well define \HTML to mean something else, like a styled version of the HTML acronym. So in version 98.1 the name has been changed to \HTMLcode.
>
> If no other definition of \HTML exists, then this command *will* be defined, to work the same as \HTMLcode.

An important aspect of this is that any of the *<tag>*, *<attribs>* and *<contents>* may be given wholly by expanding a LaTeX macro, or may contain arbitrary macros, perhaps including other \HTMLcode commands. The contents of Figure 6 was constructed using this feature; its LaTeX source follows.

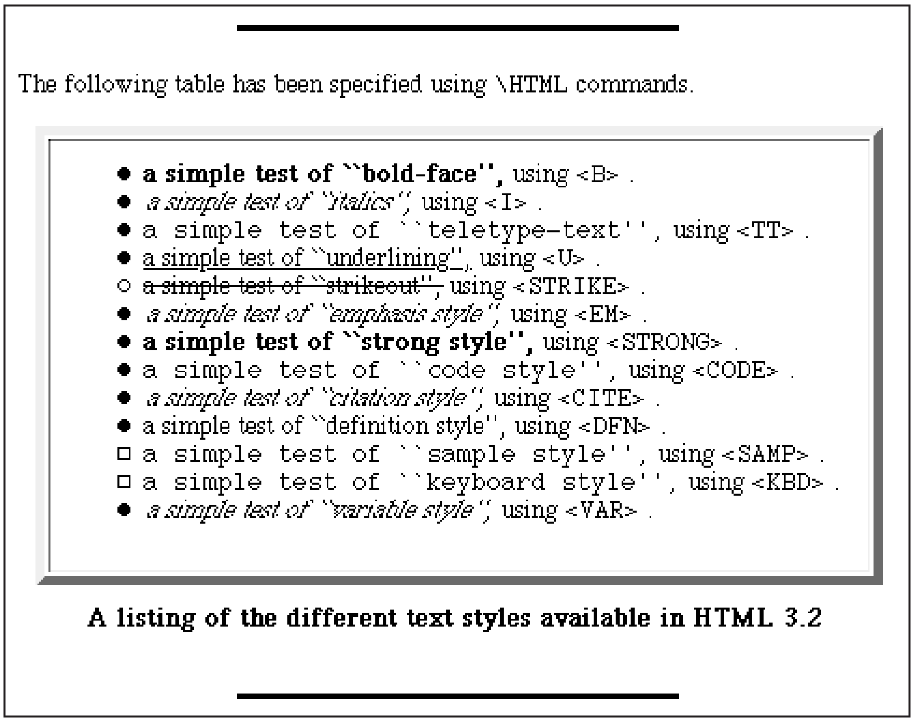

Figure 6: Example use of macros for raw HTML code.

```
\newcommand{\myalign}{center}
\newcommand{\mylist}{UL}
\newcommand{\myitem}[2]{\HTMLcode[disc]{LI}{\simpletest{#1}{#2}}}
```

```
\newcommand{\simpletest}[2]{%
 \HTMLcode{#1}{ a simple test of ``#2'',} using \HTMLcode{CODE}{<#1>} .}
\newcommand{\tableopts}{10,border=5}

\newcommand{\tablelist}[4][left]{\HTMLcode[#1]{DIV}{
\HTMLcode[\tableopts]{TABLE}{
\HTMLcode[bottom]{CAPTION}{
#3
}\HTMLcode{TR}{\HTMLcode{TD}{
\HTMLcode{#2}{
#4
}}}
}}\HTMLcode[all]{BR}}

\tablelist[\myalign]{\mylist}{%
\textbf{A listing of the different text styles available in HTML 3.2}}{%
\myitem{B}{bold-face}
\myitem{I}{italics}
\myitem{TT}{teletype-text}
\myitem{U}{underlining}
\HTMLcode[circle]{LI}{\simpletest{STRIKE}{strikeout}}
\myitem{EM}{emphasis style}
\myitem{STRONG}{strong style}
\myitem{CODE}{code style}
\myitem{CITE}{citation style}
\myitem{DFN}{definition style}
\HTMLcode[square]{LI}{\simpletest{SAMP}{sample style}}
\HTMLcode[square]{LI}{\simpletest{KBD}{keyboard style}}
\myitem{VAR}{variable style}}
```

The above code demonstrates many aspects of the way `\HTML` commands can be used.

nesting: `\HTML` commands can be nested to arbitrary depth.

macros: Macros can be used to specify all or part of each argument.

within macros: `\HTMLcode` commands work correctly within the expansions of other macros.

attribute values: Information within <*attribs*> can be specified in a very loose way, as a comma-separated list of key/value pairs or as single values.
Not even the commas are necessary: space(s), <*tab*>s or newlines are equally effective.
Indeed the horizontal rules preceding and following the table were specified by:

```
\HTMLcode[50\% 3 noshade center]{HR}
```

attribute names: Usually it is *not necessary* to know the names of the attributes to the tags that are to be used. It is sufficient just to give the values; these will be matched to the appropriate attribute, according to the type of data required. (If names are given, these are case-insensitive.)

newlines: Although LaTeX ignores linebreaks within the source code, this is not so with LaTeX2HTML. The strange spreading-out of the definition of the `\tablelist` command above was done with the purpose solely of making the code in the resulting HTML files more easily readable, to a human. (As most browsers ignore those newlines anyway, more compact code would have rendered the same on-screen.)

Some further aspects of the use of this `\HTML` command are not apparent from the above example.

invalid *<tag>* : If a *<tag>* is specified that is not part of the HTML 3.2 specifications, then it and its attributes are not placed into the HTML document created by LaTeX2HTML. Any *<contents>* *is* included as ordinary data; i.e. as text in paragraphs, etc.

required attributes: Some tags have attributes which are required to have values, if that tag is to be included in an HTML document. Using the `\HTML` command, if any such attribute is not given an appropriate value then the tag is ignored. Any *<contents>* are included in the document, as ordinary character data.

valid HTML : Currently there is *no* checking that the *<contents>* of a *<tag>* contains only data (perhaps including other tags) allowed by the DTD for HTML 3.2.

The requirement to produce valid HTML *currently rests with the user.*

This issue will be addressed in forthcoming revisions of LaTeX2HTML.

extra attributes and values: The list of attributes for a *<tag>* can include key-value pairs whose keys do not match any valid attribute for the *<tag>*. Such key-value pairs are simply ignored. Similarly extra data values are ignored, as are values that do not match the requirements for any valid attribute.

attributes with similar data-types: Several attributes to a *<tag>* may use values having the same or similar data-types. First any key-value pairs are processed. Remaining values are allocated to those attributes which do not already have a value. An ordering of the attributes is used, based on a perceived likelihood of each attribute being required to be changed from its default setting.

4.4 Conditional Text

`\begin{latexonly}` and `\begin{htmlonly}` 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, used on page 41 of this manual:

```
\begin{latexonly}
 \begin{figure}
     \begin{center}
     \fbox{\includegraphics[width=4in]{psfiles/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}
```

Note the warning at the bottom of page 37 concerning how the environment delimiters should be used in the LaTeX source code.

`\htmlonly...\endhtmlonly` This is an alternative way to specify a chunk of material intended for the `HTML` version only, using the old $\mathcal{AMS}$-style of delimiting environments. Use of this style is discouraged; the htmlonly environment is preferred.

`\latexonly...\endlatexonly` This is an alternative way to specify a chunk of material intended for the LaTeX typeset version only, using the old $\mathcal{AMS}$-style of delimiting environments. Use of this style is discouraged; the latexonly environment or the unscoped `%begin{latexonly}` construction are preferred.

Note the warning at the bottom of page 37 concerning how the environment delimiters should be used in the LaTeX source code.

`\latex`, `\html` and `\latexhtml` There are also shorthand notations to accomplish the same thing as in the latexonly environment and htmlonly environment, but with less typing.

- The `\latex{...}` command causes everything within the braces to be processed by LaTeX, but ignored by LaTeX2`HTML`.
- Conversely, the `\html{...}` command causes everything within the braces to be ignored by LaTeX and processed by LaTeX2`HTML`.
- 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 LaTeX2`HTML`.

Warning: Only small pieces of text work reliably in this way. With whole paragraphs or contained sub-environments, the "conditional" environments should be used instead.

`%begin{latexonly}` Another variant of the latexonly environment is available, in which everything between `%begin{latexonly}` and `%end{latexonly}` is ignored by LaTeX2`HTML`. 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, since LaTeX sees the initial `%`s simply as starting comments.

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.

Note the warning at the bottom of page 37 concerning how the environment delimiters should be used in the LaTeX source code.

Warning: Be careful when using LaTeX commands which alter the values of counters (e.g. numbered figures or equations) in conditional text, because this may cause the counter values in the electronic version to lose synchronisation with the values of the corresponding counters in the LaTeX version.

`\begin{imagesonly}` This environment is used to put LaTeX code into the `images.tex` file, to be used when generating images. Typically this is used to add commands to the preamble of `images.tex`, such as setting the text or background color. However code can be added at any other point as well; e.g. to change the background color of all images after a certain point in the document.

Note the warning at the bottom of page 37 concerning how the environment delimiters should be used in the LaTeX source code.

`\begin{makeimage}` This is a special environment which forces an image to be made of its contents. That is, one gets effectively a snapshot of a portion of a page that has been typeset using LaTeX. Within the normal LaTeX typeset version of the document, this environment is completely transparent, adding its contents to the page as usual.

One further important use of the makeimage environment is as follows. If a makeimage environment occurs as a sub-environment within a figure environment, then an image will *not* be made of the figure's contents. Instead, the contents are treated as normal text, each part being handled as if there were no figure at all, except that everything is placed within a single cell of a `<TABLE>`...`</TABLE>` construction in HTML 3.2. The contents of any `\caption` commands are placed between `<CAPTION>`...`</CAPTION>` tags for the `<TABLE>`.

Normally an image of the entire contents of the figure would be placed within the single cell of the `<TABLE>`. Now images are made of any subparts of those figure's contents that really need it, in particular the makeimage sub-environments. An empty makeimage sub-environment does not generate an image of itself, yet still it inhibits an image being made of the whole figure. These comments apply also to table environments.

4.5 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.

`\hyperref` 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`, taking 4 arguments, can be used in that 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 is 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 4.5 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.)

An extended syntax for `\hyperref` uses an optional argument, which determines what information is to be placed in the LaTeX version of the document. The value of this optional argument can also affect the number of required arguments. These forms are recognised:

```
\hyperref[ref]{<HTML-text>}{<LaTeX-text>}{<post-LaTeX>}{<label>}
\hyperref{<HTML-text>}{<LaTeX-text>}{<post-LaTeX>}{<label>}

\hyperref[pageref]{<HTML-text>}{<LaTeX-text>}{<post-LaTeX>}{<label>}
\hyperref[page]{<HTML-text>}{<LaTeX-text>}{<post-LaTeX>}{<label>}

\hyperref[noref]{<HTML-text>}{<LaTeX-text>}{<label>}
\hyperref[no]{<HTML-text>}{<LaTeX-text>}{<label>}
```

The first two are the defaults, where LaTeX uses `\ref{<label>}`. With the next two LaTeX uses `\pageref{<label>}`, while with the final two LaTeX completely ignores the *<label>*, setting just the *<LaTeX-text>*.

For creating hyperlinks to other documents using symbolic reference *<label>*s, see also the `\externalref` command, described on page 51.

The preceding paragraph is an example of the use of the `\hyperref[page]` option. Its source code is:

```
For creating hyperlinks to other documents
using symbolic reference \Meta{label}s,
see also the \Lc{externalref}
\hyperref[page]{command}{command, described on page~}{}{externref}.
```

which appears in the HTML version as:

> For creating hyperlinks to other documents, using symbolic reference *<label>*s, see also the `\externalref` command.

with the command being an active hyperlink. In fact both `\hyperref` and the `\htmlref` command, to be described next, permit textual hyperlinks based on symbolic *<label>*s from external files.

`\htmlref` 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}.
```

which produces:

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

In the HTML version it is shown as:

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

4.6 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{tugURL="\htmladdnormallink
{http://www.tug.org/}{http://www.tug.org}"}

@string{danteURL="\htmladdnormallink
{http://www.dante.de/}{http://www.dante.de}"}

@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 "
 # tugURL # " and " # danteURL }
```

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[40].

The natbib package, written for LaTeX by Patrick Daly, provides even more flexibility in the way a reference may be cited. All the features of this package are implemented for LaTeX2HTML via the `natbib.perl` file. (Indeed there is even a mode whereby natbib handles the Harvard style of citation. This requires loading also the nharvard package.)

Thanks... to Martin Wilck for the bulk of the work in producing this extension, and to Ross Moore for necessary adjustments to allow it to work correctly with the document segmentation strategy.

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

`\hypercite` Analogous to `\hyperref` is the `\hypercite` command, which allows a free-form textual hyperlink to the bibliography, whereas the LaTeX typeset version contains the usual citation code. The allowed syntax is as follows.

```
\hypercite[int]{<HTML-text>}{<LaTeX-text>}{<opt-LaTeX>}{<label>}
\hypercite[cite]{<HTML-text>}{<LaTeX-text>}{<opt-LaTeX>}{<label>}
\hypercite{<HTML-text>}{<LaTeX-text>}{<opt-LaTeX>}{<label>}

\hypercite[nocite]{<HTML-text>}{<LaTeX-text>}{<label>}
\hypercite[no]{<HTML-text>}{<LaTeX-text>}{<label>}
\hypercite[ext]{<HTML-text>}{<LaTeX-text>}{<label>}
```

The first three forms are equivalent; LaTeX uses `\cite[<opt-LaTeX>]<label>`, after placing the `<LaTeX-text>`. Note that `{<opt-LaTeX>}` *must* be specified, even if empty '`{}`'.

Similarly the latter three forms are equivalent, with LaTeX using `\nocite{<label>}`, to force the particular reference to appear on the bibliography page, even though no explicit marker is placed at this point. (Thus there is no need for an optional `<opt-LaTeX>` argument.) Within the HTML version a hyperlink is produced when the `<HTML-text>` is not empty. External label files are also searched, in order to match the symbolic `<label>`, see also `\externalcite` on page 51.

Earlier in this manual the following source code was used:

```
commands described in the \LaTeX{} \htmlcite{blue book}{lamp:latex},
...
as well as many other \LaTeX{} constructions, such as are described in
the \LaTeX{} \hypercite{\emph{Companion}}{\emph{Companion}}{}{goossens:latex}
and \LaTeX{} \hypercite{\emph{Graphics Companion} (e.g. \Xy-pic)}%
{\emph{Graphics Companion}}{\Xy-pic}{goossens:latexGraphics};
```

which produces:

> commands described in the LaTeX blue book ,
> ...
> as well as many other LaTeX constructions, such as are described in the LaTeX *Companion*[2] and LaTeX *Graphics Companion*[3, XY-pic];

whereas in the HTML version one sees:

> commands described in the LaTeX blue book,
> ...
> as well as many other LaTeX constructions, such as are described in the LaTeX *Companion* and LaTeX *Graphics Companion* (e.g. XY-pic);

`\htmlcite` Analogous to `\htmlref` is the `\htmlcite` command, which creates a textual hyperlink to a place on the document's bibliography page, but without displaying any reference marker in the LaTeX typeset version. (See above for an example.)

The `\externalcite` command, described on page 51, provides a similar facility when the bibliography page is "external"; that is, not part of the current document.

4.7 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.

`\externallabels` 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 4.5 is not defined within the document itself, LaTeX2HTML will look for that reference in one of the external files[41]. 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 in Section 4.10.

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.

`\externalref` This mechanism can be extended to external documents:

```
\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.

`\externalcite` Analogous to `\externalref`, the `\externalcite` command is used to create a citation link, where the bibliography page is not part of the current document. As with `\externalref` symbolic labels for the bibliography page must have been loaded using `\externallabels`.

A particularly important use for this is in allowing multiple documents to access information in a common bibliographic listing. For example: all of an author's publications; a comprehensive listing of publications in a particular field; the (perhaps yearly) output of publications from a particular organisation or institution.

Thanks... to Uffe Engberg[42] for suggesting this feature.

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

[42] http://www.brics.dk/~engberg

4.7.1 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 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[43].

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[44] 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).

4.8 Miscellaneous commands for `HTML` effects

The html package, through the LaTeX input file `html.sty`, and its *Perl* counterpart `html.perl`, implements several new commands that are intended entirely for effects within the produced `HTML` files. In LaTeX these commands, their arguments, and any optional arguments are completely ignored.

`\htmlrule` and `\htmlrule*` One such 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.

Much variation can be obtained in the horizontal rule that is produced, using extended forms of the `\htmlrule` command:

```
\htmlrule
\htmlrule*
\htmlrule[<attribs>]
\htmlrule*[<attribs>]
```

[43] 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).

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

Whereas a "break" tag `<BR>` normally precedes the `<HR>` generated by the `\htmlrule` command, this break is omitted when using the `\htmlrule*` variant.
Furthermore, the optional argument *<attribs>* can be used to specify attributes for *both* the `<HR>` and `<BR>` tags. More specifically, *<attribs>* should be a list of attribute-names and/or key-value pairs *<key>*=*<value>* separated by spaces or commas. This list is parsed to extract those attributes applicable to the `<HR>` tag, and those applicable to the `<BR>` (with the unstarred variant).

Using HTML 3.2, this allows variations to be specified for:

- the (vertical) thickness of the horizontal line in pixels: `SIZE=`*<num>*;
- the (horizontal) width of the line in pixels or points: `WIDTH=`*<width>*;
- alignment: `WIDTH="..."` taking `left`, `right` or `center`;
- removal of the shadowed effect `NOSHADE`;
- positioning of the rule with respect to text-flows: `CLEAR="..."` taking `left`, `all`, `right` or `none`.

Some examples of these effects appear on the HTML version of this page.

`\strikeout{`*<text>*`}` With this command the *<text>* is processed as normal in the HTML version, then placed between `<STRIKE>`...`</STRIKE>` tags. Thus a horizontal line should be drawn through the middle of the *<text>*. Currently the command and the *<text>* are ignored in the LaTeX version.

`\tableofchildlinks` As an extra aid to navigation within a long page, containing several (sub)subsections or deeper levels of sectioning, there is the `\tableofchildlinks` command. This does not generate anything new, for a table of the child links on or from a page is generated automatically by LaTeX2HTML.

However if this command, or its variant `\tableofchildlinks*`, occurs within the source code to appear on a particular HTML page, then the child-links table will be placed at that point where the command occurs. Normally a break tag `<BR>` is inserted to separate the table of child-links from the surrounding text. The `\tableofchildlinks*` omits this extra break when it would result in too much space above the table.

For example throughout this section of the HTML version of the manual, all subsections in which several explicit commands have been discussed have their child-links table placed at the top of the page, using `\tableofchildlinks*`. This helps to quickly find the description of how the commands are used.

`\htmlinfo` Normally an "About this document..." page is created at the end of the HTML document, containing technical information about how the document was created, by whom, or any other information contained in the `$INFO` variable. This information can be made to appear at any other place within the document by specifying `\htmlinfo` at the desired place in the source. For example, the information may be best suited for the title-page.

The variant `\htmlinfo*` places the information, but leaves out the standard "About this document..." header. Instead the `\htmlhead` command can be used to place an alternative heading, prior to the `\htmlinfo*` command. Neither this heading nor the `$INFO` contents appears in the LaTeX typeset version.

`\bodytext{<options>}` The text and background colors, and colors for the text of hypertext links can be set on an HTML page by giving appropriate attributes with the `<BODY...>` tag. This is particularly easy to do using the `\bodytext` command, which simply inserts the `<code>` as the desired list of attributes.

Warning: Any previous settings for the `<BODY...>` tag are discarded. Furthermore no checking is done to verify whether the given `<options>` indeed contains a list of attributes and values valid for the `<BODY...>` tag. When using `\bodytext` you are assumed to know precisely what you are doing!

Other packages contain commands which alter the contents of the `<BODY...>` tag; notably the `color.perl` implementation of LaTeX's color package, and the (prototype) frames package, by Martin Wilck and Ross Moore. In both these packages the requested information is checked for validity as an attribute within the `<BODY...>` tag.

`\htmlbody{<options>}` This is similar to the `\bodytext` command, except that it adds the value of an attribute, or allows an existing value to be changed. Thus it can be used to alter just a single one of the text and background colors, colors for the text of hypertext links or add a background pattern. The `<options>` are given as key-value pairs; some checking is done to ensure the validity of the attributes whose values are being set.

`\htmlbase{<URL>}` This specifies that the given `<URL>` be included in the `<HEAD>` section of each HTML page via a tag: `<BASE HREF="<URL>"`. Such a feature is particularly useful...

- when preparing a document whose final location may be different from where it was created; By making all internal references be relative (to the the provided `<URL>`), a whole directory tree containing the document and all its subparts can be moved to elsewhere. A single edit in each HTML file produces the complete document intact at the new location.
- by allowing just single page to be copied to another location, but act as if it were part of the original document (provided this is accessible across the Web). Relative URLs within the copied page are relative to the base `<URL>`, rather than relative to the new location.
- Other uses for this feature are likely to become apparent.

`\HTMLset{<which>}{<value>}` **and** `\HTMLsetenv{<which>}{<value>}` The `\HTMLset` command provides a mechanism whereby an arbitrary *Perl* variable can be assigned a value dynamically, during the LaTeX2HTML processing. A variable having name '`$<which>`' is assigned the specified `<value>`, overwriting any value that may exist already. The `\HTMLsetenv` is for the same purpose, but it is expanded in order as if it were an environment, rather than a command.

Warning: This is intended for *Perl* programmers only. Use this command at your own risk!

`\latextohtml` expands to the name LaTeX2HTML, of this translator. Commands for parts of names of important LaTeX packages are also included with LaTeX2HTML: e.g. TeX, LaTeX, AMS, XY. (This is to make it easy to refer to these products, in a consistent way within the HTML pages; you may still need LaTeX definitions for the typeset version.)

4.9 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 3.2 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` helps automate this process. This program (which is really a *Perl* script) takes one mandatory argument and an 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[45]:

```
#
#  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

[45]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.

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` .

4.10 Document Segmentation[46]

One of the greatest appeals of the World-Wide Web is its high connectivity through hyperlinks. As we have seen, the LaTeX author can provide these links either manually or symbolically. Manual links are more tedious because a URL must be provided by the author for every link, and updated every time the target documents change. Symbolic links are more convenient, because the translator keeps track of the URLs. Earlier releases of LaTeX2HTML required the entire document to be processed together if it was to be linked symbolically. However it was easy for large documents to overwhelm the memory capacities of moderate-sized computers. Furthermore, processing time could become prohibitively high, if even a small change required the entire document to be reprocessed.

For these reasons, program segmentation was developed. This feature enables the author to subdivide his document into multiple *segments*. Each segment can be processed independently by LaTeX2HTML. Hypertext links between segments can be made symbolically, with references shared through auxiliary files. If a single segment changes, only that segment needs to be reprocessed (unless a label is changed that another segment requires). Furthermore, the entire document can be processed without modification by LaTeX to obtain the printed version.
The top level segment that LaTeX reads is called the *parent* segment. The others are called *child* segments.
Document segmentation does require a little more work on the part of the author, who will now have to undertake some of the book-keeping formerly performed by LaTeX2HTML. The following four LaTeX extensions carry out segmentation:

`\segment{`*<file>*`}{`*<sec-type>*`}{`*<heading>*`}` This command indicates the start of a new program segment. The segment resides in *<file>*`.tex`, represents the start of a new LaTeX sectional unit of type *<sec-type>* (e.g., `\section`, `\chapter`, etc.) and has a heading of *<heading>*. (A variation `\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 initialise itself for the new document segment.

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

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 is to be read.
- `images` Allows images generated in other segments to be reused with the current segment.

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 3.8 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 generated automatically by a `\segment` command. It is not normally placed in the document at all; instead it facilitates information being passed from parent to child via the `<file>.ptr` file. 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. From version V97.1 , it is possible to use this command to insert extra section-headings, for use in the HTML version only.

`\htmlnohead` When placed at the top of the preamble of a document segment, the `\htmlnohead` command discards everything from the current page that has been placed already. Usually this will be just the section-head, from the `\htmlhead` command in the `.ptr` file. Numbering and color information is unaffected. This allows an alternative heading to be specified, or no heading at all in special circumstances; e.g. the page contains a single large table with a caption.

\segmentcolor{<*model*>}{<*color*>} This command is generated automatically by a \segment command. It is not normally placed in the document at all; instead it facilitates information being passed from parent to child via the <*file*>.ptr file. It specifies to LaTeX2HTML that text in the document should have the color <*color*>.

\segmentpagecolor{<*model*>}{<*color*>} This command is generated automatically by a \segment command. It is not normally placed in the document at all; instead it facilitates information being passed from parent to child via the <*file*>.ptr file. It specifies to LaTeX2HTML that the background of in the document should have the color <*color*> .

The use of the segmenting commands is best illustrated by the example below. You might want to check your segmented document for consistency using the -unsegment command line option.

4.10.1 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 recent distributions of LaTeX2HTML, having more prolific comments than are shown here.

The text of report.tex is as follows:

```
\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 as follows:

```
\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 as follows:

```
\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 LaTeX2HTML updates its own internal label file only if something in its current program segment causes the labels to change from the previous run. This ensures that LaTeX2HTML is not run unnecessarily. It is also usual for the information page to be suppressed by specifying '`-info 0`' for all but the top-level document.

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 images and `<`*prefix*`><`*type*`>.pl` files are created and stored within this directory.

Sometimes it is desirable to build one or more segments within separate sub-directories. This is especially so when a segment has a large number of images, 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}
```

5 Customising the Layout of HTML pages

LaTeX2HTML is a program for creating hyperlinked sets of HTML pages from a document marked-up using LaTeX commands. Previous sections have discussed the results of specific LaTeX commands. In this section we discuss instead the extensive range of command-line switches and options, and other aspects of *Perl* code, that affect the way the translation is performed.

To use LaTeX2HTML to translate a file `<file>.tex` containing LaTeX commands, type:

```
latex2html <file>.tex
```

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 browser, such as *NCSA Mosaic* or *Netscape Navigator*, on the main HTML file which is `<file>/<file>.html`. The file will contain navigation links to the other parts of the generated document. The `.tex` suffix is optional and will be supplied by the program if it is omitted by the user. Other suffixes are acceptable also, such as `.doc`.

It is possible to customise the output from LaTeX2HTML using a number of command-line options (see Section 5.2) with which you can specify:

- how to break up the document;
- where to put the generated files, and what are their names;
- the title for the whole document;
- the signature at the end of each page is;
- how many navigation panels to provide, what links to put in them;
- what other documents this one links to;
- extra information to include about the document;
- whether to retain the original LaTeX section-numbering scheme;
- and many other things that affect how the information is obtained, processed or displayed in the resulting `.html` files and images.

The LaTeX2HTML script includes a short manual which can be viewed with the command:

```
nroff -man latex2html
```

5.1 Developing Documents using LaTeX2HTML

Although any document containing LaTeX commands can be translated by the LaTeX2HTML translator, the best results are obtained when that document is itself a valid LaTeX document. Indeed it is generally a good idea to develop documents so that they produce good readable results in both the LaTeX typeset version as well as a set of HTML pages. This is not just a nicety; there are several good practical reasons for doing this.

LaTeX macros: The macro commands that LaTeX2HTML recognises are based upon corresponding commands for LaTeX. If one tries to use syntax that is incorrect for LaTeX then there is no reason why LaTeX2HTML should be able to "get it right", by somehow recognising the true intent.

error checking: Processing the document first using LaTeX is the easiest, and quickest, way to check for valid syntax. Whereas LaTeX stops at each error (when run in interactive mode), allowing a fix to be made "on the spot" or a "stop-fix-restart", LaTeX2HTML does not stop when it detects an error in LaTeX syntax. Useful messages are given concerning missing or unmatched braces, but other apparent anomalies generate only warning messages, which are saved to the end. (Some warnings are also shown immediately when the `$VERBOSITY` variable is set to at least 3.) In practice it can be much quicker to test for invalid syntax using LaTeX before attempting to use the LaTeX2HTML translator.

Furthermore, LaTeX warns of cross-reference labels that have not been defined. This is useful to help avoid having hyperlinks which point to nowhere.

The case of missing braces, or an unmatched opening brace, is an error that LaTeX2HTML actually handles better than LaTeX (or rather, the underlying TeX processor). Whereas TeX only detects an error when something else goes wrong later in the processing, LaTeX2HTML shows where the unmatched brace itself occurs.

auxiliary file: Some information that LaTeX2HTML might need is normally read from the `.aux` file for the document being processed; or perhaps from the `.aux` file of a different document, of which the current document is just a portion. Clearly a valid LaTeX run is required to produce the `.aux` file.

Of course if no information in the `.aux` file is actually used, then this LaTeX run can be neglected. Also, if the `.aux` file has already been created, and edits are made on the source which do not alter the information stored within the `.aux` file, then there is no need for a fresh LaTeX run (except for the purposes of error-checking).

bibliography: Suppose the document requires a bibliography, or list of references, which is to be prepared using BibTeX. LaTeX2HTML reads citation information from the `.aux` file, and can import the bibliography itself from the `.bbl` file. However these must first be created using LaTeX.

document segmentation: With the document segmentation technique, discussed fully in Section 4.10, it is vitally important that the full document processes correctly in LaTeX. The desired effect is that of a single large document, whereas the pieces will actually be processed separately. To achieve this, LaTeX writes vital information into special `.ptr` files. Like the `.aux` file, these files are read by LaTeX2HTML to get section-numbering and other information to be used while processing each segment.

print quality: When a document contains automatically-generate images, these images are usually bitmaps designed for viewing on-screen. In general the resolution of these is too poor to give a good result when printed on a high-resolution laser-printer. Thus if it is likely that the reader will want to obtain a printed version of your document, then it is best to include a hyperlink to the typeset `.dvi` version, or to a PostScript conversion of the `.dvi` file. (In either case, a link to a compressed version is even better.)

5.2 Command-Line Options

The command-line options described here can be used to change the default behaviour of LaTeX2HTML. Alternatively, often there is a corresponding environment variable whose value may be set or changed within a `.latex2html-init` initialisation file, in order to achieve the same result. There are so many options that they are listed here in groups, according to the nature of the effects they control. When a large number of such options are required for the processing of a document, it is usual to store the switches and their desired settings within a `Makefile`, for use with the Unix `make` utility. Now a simple command such as:

```
make mydocument
```

can initiate a call to `latex2html` that would otherwise take many lines of typing. Indeed it could instigate several such calls to LaTeX2HTML, or to other programs such as LaTeX and BibTeX, `dvips` and others. The document segmentation feature, discussed in Section 4.10, uses this technique extensively.

5.2.1 Options controlling Titles, File-Names and Sectioning

-t `<top-page-title>` (Same as setting: `$TITLE = "<top-page-title>";`)
Name the document using this title.

-short_extn (Same as setting: `$SHORTEXTN = 1;`)
Use a filename prefix of `.htm` for the produced `HTML` files. This is particularly useful for creating pages to be stored on CD-ROM or other media, to be used with operating systems that require a 3-character extension.

-long_titles `<num>` (Same as setting: `$LONG_TITLES = <num>;`)
Instead of the standard names: `node1.html`, `node2.html`,... the filenames for each `HTML` page are constructed from the first `<num>` words of the section heading for that page, separated by the '_' character. Commas and common short words (`a an to by of and for the`) are omitted from both title and word-count.

Warning: Use this switch with great caution. Currently there are no checks for uniqueness of names or overall length. Very long names can easily result from using this feature.

-custom_titles (Same as setting: `$CUSTOM_TITLES = 1;`)
Instead of the standard names: `node1.html`, `node2.html`, ... the filenames for each `HTML` page are constructed using a *Perl* subroutine named `custom_title_hook`. The user may define his/her own version of this subroutine, within a `.latex2html-init` file say, to override the default (which uses the standard names). This subroutine takes the section-heading as a parameter and must return the required name, or the empty string (default).

-dir `<output-directory>` (Same as setting: `$DESTDIR = "<output-directory>";`)
Redirect the output to the specified directory. The default behaviour is to create (or reuse) a directory having the same name as the prefix of the document being processed.

-no_subdir (Same as setting: `$NO_SUBDIR = 1;`)
Place the generated `HTML` files into the current directory. This overrides any `$DESTDIR` setting.

-prefix `<filename-prefix>` (Same as setting: `$PREFIX = "<filename-prefix>";`)
The `<filename-prefix>` will be prepended to all `.gif`, `.pl` and `.html` files produced, except for the top-level `.html` file; it may include a (relative) directory path. This will enable multiple products of LaTeX2HTML to peacefully coexist in the same directory. However, do *not* attempt to *simultaneously* run multiple instances of LaTeX2HTML using the same output directory, else various temporary files will overwrite each other.

-auto_prefix (Same as setting: $AUTO_PREFIX = 1;)
Constructs the prefix as '<*title*>–' to be prepended to all the files produced, where <*title*> is the name of the LaTeX file being processed. (Note the '–' in this prefix.) This overrides any $PREFIX setting.

-no_auto_link (Same as setting: $NO_AUTO_LINK = 1;)
If $NO_AUTO_LINK is empty and variables $LINKPOINT and $LINKNAME are defined appropriately (as is the default in the latex2html.config file), then a hard link to the main HTML page is produced, using the name supplied in $LINKNAME. Typically this is index.html; on many systems a file of this name will be used, if it exists, when a browser tries to view a URL which points to a directory. On other systems a different value for $LINKNAME may be appropriate. Typically $LINKPOINT has value $FILE.html, but this may also be changed to match whichever HTML page is to become the target of the automatic link.
Use of the '-no_auto_link' switch *cancels* this automatic linking facility, when not required for a particular document.

-split <*num*> (Same as setting: $MAX_SPLIT_DEPTH = <*num*>;) (default is 8)
Stop splitting sections into separate files at this depth. Specifying '-split 0' will put the entire document into a single HTML file. See below for the different levels of sectioning. Also see the next item for how to set a "relative" depth for splitting.

-split +<*num*> (Same as setting: $MAX_SPLIT_DEPTH = -<*num*>;) (default is 8)
The level at which to stop splitting sections is calculated "relative to" the shallowest level of sectioning that occurs within the document. For example, if the document contains \section commands, but no \part or \chapter commands, then '-split +1' will cause splitting at each \section but not at any deeper level; whereas '-split +2' or '-split +3' also split down to \subsection and \subsubsection commands respectively. Specifying '-split +0' puts the entire document into a single HTML file.

-link <*num*> (Same as setting: $MAX_LINK_DEPTH = <*num*>;) (default is 4)
For each node, create links to child nodes down to this much deeper than the node's sectioning-level. Specifying '-link 0' will show no links to child nodes from that page, '-link 1' will show only the immediate descendents, etc. A value at least as big as that of the '-split <*num*>' depth will produce a mini table-of-contents (when not empty) on each page, for the tree structure rooted at that node.

When the page has a sectioning-level less than the '-split' depth, so that the a mini table-of-contents has links to other HTML pages, this table is located at the bottom of the page, unless placed elsewhere using the \tableofchildlinks command.

On pages having a sectioning-level just less than the '-split' depth the mini table-of-contents contains links to subsections etc. occurring on the *same* HTML page. Now the table is located at the *top* of this page, unless placed elsewhere using the \tableofchildlinks command.

-toc_depth <*num*> (Same as setting: $TOC_DEPTH = <*num*>;) (default is 4)
Sectioning levels down to <*num*> are to be included within the Table-of-Contents tree.

-toc_stars (Same as setting: $TOC_STARS = 1;)
Sections created using the starred-form of sectioning commands are included within the Table-of-Contents. As with LaTeX, normally such sections are *not* listed.

-show_section_numbers (Same as setting: `$SHOW_SECTION_NUMBERS = 1;`)
Show section numbers. By default section numbers are *not* shown, so as to encourage the use of particular sections as stand-alone documents.

-unsegment (Same as setting: `$UNSEGMENT = 1;`)
Treat a segmented document (see the section about document segmentation) like it were not segmented. This will cause the translator to concatenate all segments and process them as a whole. You might find this useful to check a segmented document for consistency.

For all documents the sectioning levels referred to above are:

0	document
1	part
2	chapter
3	section
4	subsection
5	subsubsection
6	paragraph
7	subparagraph
8	subsubparagraph

These levels apply even when the document contains no sectioning for the shallower levels; e.g. no `\part` or `\chapter` commands is most common, especially when using LaTeX's article document-class.

5.2.2 Options controlling Extensions and Special Features

The switches described here govern the type of `HTML` code that can be generated, and how to choose between the available options when there are alternative strategies for implementing portions of LaTeX code.

-html_version `(2.0|3.0|3.2)[,(math|i18n|table)]*`
(Same as setting: `$HTML_VERSION = "...  ";`)
This specifies both the `HTML` version to generate, and any extra (non-standard) `HTML` features that may be required. The version number corresponds to a published DTD for an `HTML` standard (although 3.0 was never accepted and subsequently withdrawn). A corresponding *Perl* file in the `versions/` subdirectory is loaded; these files are named '`html`<*num*>`.pl`'.

Following the version number, a comma-separated list of extensions can be given. Each corresponds to a file '<*name*>`.pl`' also located in the `versions/` subdirectory. When such a file is loaded the resulting `HTML` code can no longer be expected to validate with the specified DTD. An exception is '`math`' when the '`-no_math`' switch is also used, which should still validate.

Currently, versions 2.0, 3.2 and 4.0 are available. (and also 2.1, 2.2, 3.0 and 3.1, for hoistorical reasons). The extensions '`i18n`', '`tables`', '`math`' correspond roughly to what used to be called versions '2.1', '2.2', '3.1' respectively, in releases of LaTeX2`HTML` up to 1996. Now these extensions can be loaded with any of '2.0', '3.2' or '4.0' as the specified standard.
The default version is usually set to be '3.2', within `latex2html.config`.

-no_tex_defs (Same as setting: `$TEXDEFS = 0;`) (default is 1)
When `$TEXDEFS` is set (default) the file `texdefs.perl` will be read. This provides code to allow common TEX commands like `\def`, `\newbox`, `\newdimen` and others, to be recognised, especially within the document preamble. In the case of `\def`, the definition may even be fully interpreted, but this requires the pattern-matching to be not too complicated.

If `$TEXDEFS` is '0' or empty, then `texdefs.perl` will not be loaded; the translator will make no attempt to interpret any raw TEX commands. This feature is intended to enable sophisticated authors the ability to insert arbitrary TEX commands in environments that are destined to be processed by LATEX anyway; e.g. figures, , pictures, etc. However this should rarely be needed, as now there is better support for these types of environment. There are now other methods to specify which chunks of code are to be passed to LATEX for explicit image-generation; see the discussion of the makeimage environment on page 47.

-external_file <*filename*> (Same as setting: `$EXTERNAL_FILE = "<filename>";`)
Specifies the prefix of the `.aux` file that this document should read. The `.aux` extension will be appended to this prefix to get the complete filename, with directory path if needed. This file could contain necessary information regarding citations, figure, table and section numbers from LATEX and perhaps other information also. Use of this switch is vital for document segments, processed separately and linked to appear as if generated from a single LATEX document.

-font_size <*size*> (Same as setting: `$FONT_SIZE = "<size>";`)
This option provides better control over the font size of environments made into images using LATEX. <*size*> must be one of the font sizes that LATEX recognizes; i.e. '10pt', '11pt', '12pt', etc. Default is '10pt', or whatever option may have been specified on the `\documentclass` or `\documentstyle` line. Whatever size is selected, it will be magnified by the installation variables `$MATH_SCALE_FACTOR`, `$FIGURE_SCALE_FACTOR` and `$DISP_SCALE_FACTOR` as appropriate.

Note: This switch provides no control over the size of text on the HTML pages. Such control is subject entirely to the user's choices of settings for the browser windows.

-scalable_fonts (Same as setting: `$SCALABLE_FONTS = 1;`)
This is used when scalable fonts, such as PostScript versions of the TEX fonts, are available for image-generation. It has the effect of setting `$PK_GENERATION` to '1', and `$DVIPS_MODE` to be empty, overriding any previous settings for these variables.

-no_math (Same as setting: `$NO_SIMPLE_MATH = 1;`)
Ordinarily simple mathematical expressions are set using the ordinary text font, but italiced. When part of the expression can not be represented this way, an image is made *of the whole formula*. This is called "simple math". When `$NO_SIMPLE_MATH` is set, then *all* mathematics is made into images, whether simple or not.

However, if the '`math`' extension is loaded, using the '`-html_version`' switch described earlier, then specifying '`-no_math`' produces a quite different effect. Now it is the special `<MATH>` tags and entities which are cancelled. In their place a sophisticated scheme for parsing mathematical expressions is used. Images are made of those sub-parts of a formula which cannot be adequately expressed using (italiced) text characters and `<SUB>` and `<SUP>` tags. See Section 3.3 for more details.

-local_icons (Same as setting: `$LOCAL_ICONS = 1;`)
A copy of each of the icons actually used within the document is placed in the directory along with the `HTML` files and generated images. This allows the whole document to be fully self-contained, within this directory; otherwise the icons must be retrieved from a (perhaps remote) server.

The icons are normally copied from a subdirectory of the `$LATEX2HTMLDIR`, set within `latex2html.config`. An alternative set of icons can be used by specifying a (relative) directory path in `$ALTERNATIVE_ICONS` to where the customised images can be found.

-init_file <*file*> Load the specified initialisation file. This *Perl* file will be loaded after loading `$HOME/.latex2html-init`, or `.latex2html-init` in the local directory, if either file exists. It is read at the time the switch is processed, so the contents of the file may change any of the values of any of the variables that were previously established, as well as any default options. More than one initialisation file can be read in this way.

-no_fork (Same as setting: `$NOFORK = 1;`)
When set this disables a feature in the early part of the processing whereby some memory-intensive operations are performed by 'forked' child processes. Some single-task operating systems, such as `DOS`, do not support this feature. Having `$NOFORK` set then ensures that unnecessary file-handles that are needed with the forked processes, are not consumed unnecessarily, perhaps resulting in a fatal *Perl* error.

-iso_language <*type*> This enables you to specify a different language type than `'EN'` to be used in the DTD entries of the `HTML` document, e.g. `'EN.US'`.

-short_index (Same as setting: `$SHORT_INDEX = 1;`)
Creates shorter Index listings, using codified links; this is fully compatible with the makeidx package.

-no_footnode (Same as setting: `$NO_FOOTNODE = 1;`)
Suppresses use of a separate file for footnotes; instead these are placed at the bottom of the `HTML` pages where the references occur.

When this option is used, it is frequently desirable to change the style of the marker used to indicate the presence of a footnote. This is done as in LaTeX, using code such as follows.

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

All the styles `\arabic`, `\alph`, `\roman`, `\Alph` and `\Roman` are available.

-numbered_footnotes (Same as setting: `$NUMBERED_FOOTNOTES = 1;`)
If this is set you will get every footnote applied with a subsequent number, to ease readability.

-address <*author-address*> (Same as setting: `$ADDRESS = "<author-address>";`)
Sign each page with this address. See `latex2html.config` for an example using *Perl* code to automatically include the date.

A user-defined *Perl* subroutine called `&custom_address` can be used instead, if defined; it takes the value of `$ADDRESS` as a parameter, which may be used or ignored as desired. At the time when this subroutine will be called, variables named `$depth`, `$title`, `$file` hold the sectioning-level, title and filename of the `HTML` page being produced; `$FILE` holds the name of the filename for the title-page of the whole document.

-info *<string>* (Same as setting: `$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 as the contents of the *"About this document"* page instead of the default information.

5.2.3 Switches controlling Image Generation

These switches affect whether images are created at all, whether old images are reused on subsequent runs or new ones created afresh, and whether anti-aliasing effects are used within the images themselves.

-ascii_mode (Same as setting: `$ASCII_MODE = $EXTERNAL_IMAGES = 1;`)

Use only `ASCII` characters and do not include any images in the final output. With '`-ascii_mode`' the output of the translator can be used on character-based browsers, such as *lynx*, which do not support inlined images (via the `<IMG>` tag).

-nolatex (Same as setting: `$NOLATEX = 1;`)

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 (Same as setting: `$EXTERNAL_IMAGES = 1;`)

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

-ps_images (Same as setting: `$PS_IMAGES = $EXTERNAL_IMAGES = 1;`)

Use links to external PostScript files rather than inlined images in the chosen graphics format.

-discard (Same as setting: `$DISCARD_PS = 1;`)

The temporary PostScript files are discarded immediately after they have been used to create the image in the desired graphics format.

-no_images (Same as setting: `$NO_IMAGES = 1;`)

Do not attempt to produce any inlined images. The missing images can be generated "off-line" by restarting LaTeX2`HTML` with the option '`-images_only`'.

-images_only (Same as setting: `$IMAGES_ONLY = 1;`)

Try to convert any inlined images that were left over from previous runs of LaTeX2`HTML`.

-reuse *<reuse_option>* (Same as setting: `$REUSE = <reuse_option>;`)

This switch specifies the extent to which image files are to be shared or recycled. There are 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 3.4.2 provides additional information about image-reuse.

-no_reuse (Same as setting: `$REUSE = 0;`)
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.)

-antialias (Same as setting: `$ANTI_ALIAS = 1;`) (Default is 0.)
Generated images of figure environments and external PostScript files should use anti-aliasing. By default anti-aliasing is not used with these images, since this may interfere with the contents of the images themselves.

-antialias_text (Same as setting: `$ANTI_ALIAS_TEXT = 1;`) (Default is 1.)
Generated images of typeset material such as text, mathematical formulas, tables and the content of makeimage environments, should use anti-aliasing effects. The default is normally to use anti-aliasing for text, since the resulting images are much clearer on-screen. However the default may have been changed locally.

-no_antialias (Same as setting: `$ANTI_ALIAS = 0;`) (Default is 0.)
Generated images of figure environments and external PostScript files should not use anti-aliasing with images, though the local default may have been changed to use it.

-no_antialias_text (Same as setting: `$ANTI_ALIAS_TEXT = 0;`) (Default is 1.)
Generated images of typeset material should *not* use anti-aliasing effects. Although on-screen images of text are definitely improved using anti-aliasing, printed images can be badly blurred, even at 300dpi. Higher resolution printers do a much better job with the resulting grey-scale images.

-white (Same as setting: `$WHITE_BACKGROUND = 1;`) (Default is 1.)
Ensures that images of figure environments have a white background. Otherwise transparency effects may not work correctly.

-no_white (Same as setting: `$WHITE_BACKGROUND = '';`) (Default is 1.)
Cancels the requirement that figure environments have a white background.

-ldump (Same as setting: `$LATEX_DUMP = 1;`) (Default is 0.)
Use this if you want to speed up image processing during the 2nd and subsequent runs of LaTeX2HTML on the same document. The translator now produces a LaTeX format-dump of the preamble to `images.tex` which is used on subsequent runs. This significantly reduces the startup time when LaTeX reads the `images.tex` file for image-generation.

This process actually consumes additional time on the first run, since LaTeX is called twice — once to create the format-dump, then again to load and use it. The pay-off comes with the faster loading on subsequent runs. Approximately 1 Meg of disk space is consumed by the dump file.

5.2.4 Switches controlling Navigation Panels

The following switches govern whether to include one or more navigation panels on each HTML page, also which buttons to include within such a panel.

-no_navigation (Same as setting: `$NO_NAVIGATION = 1;`)
Disable the mechanism for putting navigation links in each page.
This overrides any settings of the `$TOP_NAVIGATION`, `$BOTTOM_NAVIGATION` and `$AUTO_NAVIGATION` variables.

-top_navigation (Same as setting: `$TOP_NAVIGATION = 1;`)
Put navigation links at the top of each page.

-bottom_navigation (Same as setting: `$BOTTOM_NAVIGATION = 1;`)
Put navigation links at the bottom of each page as well as the top.

-auto_navigation (Same as setting: `$AUTO_NAVIGATION = 1;`)
Put navigation links at the top of each page. Also put one at the bottom of the page, if the page exceeds `$WORDS_IN_PAGE` number of words (default = 450).

-next_page_in_navigation (Same as setting: `$NEXT_PAGE_IN_NAVIGATION = 1;`)
Put a link to the *next logical page* in the navigation panel.

-previous_page_in_navigation
(Same as setting: `$PREVIOUS_PAGE_IN_NAVIGATION = 1;`)
Put a link to the *previous logical page* in the navigation panel.

-contents_in_navigation (Same as setting: `$CONTENTS_IN_NAVIGATION = 1;`)
Put a link to the *table-of-contents* in the navigation panel if there is one.

-index_in_navigation (Same as setting: `$INDEX_IN_NAVIGATION = 1;`)
Put a link to the *index-page* in the navigation panel if there is an index.

5.2.5 Switches for Linking to other documents

When processing a single stand-alone document, the switches described in this section should not be needed at all, since the automatically generated navigation panels, described in Section 5.2.4, should generate all the required navigation links. However if a document is to be regarded as part of a much larger document, then links from its first and final pages, to locations in other parts of the larger (virtual) document, need to be provided explicitly for some of the buttons in the navigation panel.

The following switches allow for such links to other documents, by providing the title and URL for navigation panel hyperlinks. In particular, the "Document Segmentation" feature of Section 4.10 necessarily makes great use of these switches. It is usual for the text and targets of these navigation hyperlinks to be recorded in a `Makefile`, to avoid tedious typing of long command-lines having many switches.

-up_url *<URL>* (Same as setting: `$EXTERNAL_UP_LINK = "<URL>";`)
Specifies a universal resource locator (URL) to associate with the *"UP"* button in the navigation panel(s).

-up_title *<string>* (Same as setting: `$EXTERNAL_UP_TITLE = "<string>";`)
Specifies a title associated with this URL.

-prev_url *<URL>* (Same as setting: `$EXTERNAL_PREV_LINK = "<URL>";`)
Specifies a URL to associate with the *"PREVIOUS"* button in the navigation panel(s).

-prev_title *<string>* (Same as setting: `$EXTERNAL_PREV_TITLE = "<string>";`)
Specifies a title associated with this URL.

-down_url <*URL*> (Same as setting: `$EXTERNAL_DOWN_LINK = "<URL>";`)
Specifies a URL for the *"NEXT"* button in the navigation panel(s).

-down_title <*string*> (Same as setting: `$EXTERNAL_DOWN_TITLE = "<string>";`)
Specifies a title associated with this URL.

-contents <*URL*> (Same as setting: `$EXTERNAL_CONTENTS = "<URL>";`)
Specifies a URL for the *"CONTENTS"* button, for document segments that would not otherwise have one.

-index <*URL*> (Same as setting: `$EXTERNAL_INDEX = "<URL>";`)
Specifies a URL for the *"INDEX"* button, for document segments that otherwise would not have an index.

-biblio <*URL*> (Same as setting: `$EXTERNAL_BIBLIO = "<URL>";`)
Specifies the URL for the bibliography page to be used, when not explicitly part of the document itself.

Warning: On some systems it is difficult to give text-strings <*string*> containing space characters, on the command-line or via a `Makefile`. One way to overcome this is to use the corresponding variable. Another way is to replace the spaces with underscores (_).

5.2.6 Switches for Help and Tracing

The first two of the following switches are self-explanatory. When problems arise in processing a document, the switches '`-debug`' and '`-verbosity`' will each cause LaTeX2HTML to generate more output to the screen. These extra messages should help to locate the cause of the problem.

-tmp <*path*> Define a temporary directory to use for image generation. If <*path*> is 0, the standard temporary directory `/tmp` is used.

-h(elp) Print out the list of all command-line options.

-v Print the current version of LaTeX2HTML.

-debug (Same as setting: `$DEBUG = 1;`)
Run in debug-mode, displaying messages and/or diagnostic information about files read, and utilities called by LaTeX2HTML. Shows any messages produced by these calls.

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).

-verbosity <*num*> (Same as setting: `$VERBOSITY = <num>;`)
Display messages revealing certain aspects of the processing performed by LaTeX2HTML on the provided input file(s). The <*num*> parameter can be an integer in the range 0 to 8. Each higher value adds to the messages produced.

0. No special tracing; as for versions of LaTeX2HTML prior to V97.1 .
1. (This is the default.) Show section-headings and the corresponding HTML file names, and indicators that major stages in the processing have been completed.
2. Print environment names and identifier numbers, and new theorem-types. Show warnings as they occur, and indicators for more stages of processing. Print names of files for storing auxiliary data arrays.

3. Print command names as they are encountered and processed; also any unknown commands encountered while pre-processing. Show names of new commands, environments, theorems, counters and counter-dependencies, for each document partition.
4. Indicate command-substitution the pre-process of math-environments. Print the contents of unknown environments for processing in LaTeX, both before and after reverting to LaTeX source. Show all operations affecting the values of counters. Also show links, labels and sectioning keys, at the stages of processing.
5. Detail the processing in the document preamble. Show substitutions of new environments. Show the contents of all recognised environments, both before and after processing. Show the cached/encoded information for the image keys, allowing two images to be tested for equality.
6. Show replacements of new commands, accents and wrapped commands.
7. Trace the processing of commands in math mode; both before and after.
8. Trace the processing of all commands, both before and after.

The command-line option sets an initial value only. During processing the value of `$VERBOSITY` can be set dynamically using the `\htmltracing{...}` command, whose argument is the desired value, or by using the more general `\HTMLset` command as follows: `\HTMLset{VERBOSITY}{<num>}`.

5.2.7 Other Configuration Variables, without switches

The configuration variables described here do not warrant having a command-line switch to assign values. Either they represent aspects of LaTeX2HTML that are specific to the local site, or they govern properties that should apply to *all* documents, rather than something that typically would change for the different documents within a particular sub-directory.

Normally these variables have their value set within the `latex2html.config` file. In the following listing the defaults are shown, as the lines of *Perl* code used to establish these values. If a different value is required, then these can be assigned from a local `.latex2html-init` initialisation file, without affecting the defaults for other users, or documents processed from other directories.

`$dd` holds the string to be used in file-names to delimit directories; it is set internally to '`/`', unless the variable has already been given a value within `latex2html.config`.

Note: This value *cannot* be set within a `.latex2html-init` initialisation file, since its value *needs to be known* in order to find such a file.

`$LATEX2HTMLDIR` Read by the `install-test` script from `latex2html.config`, its value is inserted into the `latex2html` *Perl* script as part of the installation process.

`$LATEX2HTMLSTYLES = "$LATEX2HTMLDIR/styles";` Read from the `latex2html.config` file by `install-test`, its value is checked to locate the `styles/` directory.

`$LATEX2HTMLVERSIONS = "$LATEX2HTMLDIR/versions";` The value of this variable should be set within `latex2html.config` to specify the directory path where the version and extension files can be found.

`$ALTERNATIVE_ICONS = '';` This may contain the (relative) directory path to a set of customised icons to be used in conjunction with the '`-local_icons`' switch.

`$TEXEXPAND = "$LATEX2HTMLDIR/texexpand";` Read by the `install-test` *Perl* script from `latex2html.config`, its value is used to locate the `texexpand` *Perl* script.

`$PSTOIMG = "$LATEX2HTMLDIR/pstoimg";` Read by the `install-test` *Perl* script from `latex2html.config`, its value is used to locate the `pstoimg` *Perl* script.

`$IMAGE_TYPE = '<image-type>';` Set in `latex2html.config`, the currently supported `<image-type>`s are: `gif` and `png`.

`$DVIPS = 'dvips';` Read from `latex2html.config` by `install-test`, its value is checked to locate the `dvips` program or script.

There could be several reasons to change the value here:

- add a switch `-P<printer>` to load a specific configuration-file;
  e.g. to use a specific set of PostScript fonts, for improved image-generation.
- to prepend a path to a different version of `dvips` than normally available as the system default (e.g. the printing requirements are different).
- to append debugging switches, in case of poor quality images;
  one can see which paths are being searched for fonts and other resources.
- to prepend commands for setting path variables that `dvips` may need in order to locate fonts or other resources.

If automatic generation of fonts is required, using *Metafont*, the following configuration variables are important.

`$PK_GENERATION = 1;` This variable must be set, to initiate font-generation; otherwise fonts will be scaled from existing resources on the local system.
In particular this variable must *not* be set, if one wishes to use PostScript fonts or other scalable font resources (see the '`-scalable_fonts`' switch).

`$DVIPS_MODE = 'toshiba';` The mode given here must be available in the `modes.mf` file, located with the *Metafont* resource files, perhaps in the `misc/` subdirectory.

`$METAFONT_DPI = 180;` The required resolution, in dots-per-inch, should be listed specifically within the `MakeTeXPK` script, called by `dvips` to invoke *Metafont* with the correct parameters for the required fonts.

`$LATEX = 'latex';` Read from `latex2html.config` by `install-test`, its value is checked to locate the `latex` program or script.

If LaTeX is having trouble finding style-files and/or packages, then the default command can be prepended with other commands to set environment variables intended to resolve these difficulties;
e.g. `$LATEX = 'setenv TEXINPUTS <path to search> ; latex'` .

There are several variables to help control exactly which files are read by LaTeX2HTML and by LaTeX when processing images:

`$TEXINPUTS` This is normally set from the environment variable of the same name. If difficulties occur so that styles and packages are not being found, then extra paths can be specified here, to resolve these difficulties.

`$DONT_INCLUDE` This provides a list of filenames and extensions to *not* include, even if requested to do so by an `\input` or `\include` command.
(Consult `latex2html.config` for the default list.)

`$DO_INCLUDE = '';` List of exceptions within the `$DONT_INCLUDE` list. These files *are* to be read if requested by an `\input` or `\include` command.

`$ICONSERVER = '<URL>';` This is used to specify a URL to find the standard icons, as used for the navigation buttons. Names for the specific images size, as well as size information, can be found in `latex2html.config`. The icons themselves can be replaced by customised versions, provided this information is correctly updated and the location of the customised images specified as the value of `$ICONSERVER`.
When the '`-local_icons`' switch is used, so that a copy of the icons is placed with the `HTML` files and other generated images, the value of `$ICONSERVER` is not needed within the `HTML` files themselves. However it *is* needed to find the original icons to be copied to the local directory.

`$NAV_BORDER = <num>;` The value given here results in a border, measured in points, around each icon. A value of '`0`' is common, to maintain strict alignment of inactive and active buttons in the control panels.

`$LINKNAME = '"index.$EXTN"';` This is used when the `$NO_AUTO_LINK` variable is empty, to allow a URL to the working directory to be sufficient to reach the main page of the completed document. It specifies the name of the `HTML` file which will be automatically linked to the directory name. The value of `$EXTN` is `.html` unless `$SHORTEXTN` is set, in which case it is `.htm` .

`$LINKPOINT = '"$FILE$EXTN"';` This specifies the name of the `HTML` file to be duplicated, or symbolically linked, with the name specified in `$LINKNAME`.
At the appropriate time the value of `$FILE` is the document name, which usually coincides with the name of the working directory.

`$CHARSET = 'iso_8859_1';` This specifies the character set used within the `HTML` pages produced by LaTeX2`HTML`. If no value is set in a configuration or initialisation file, the default value will be assumed. The lowercase form `$charset` is also recognised, but this is overridden by the uppercase form.

`$ACCENT_IMAGES = 'large';` Accented characters that are not part of the ISO-Latin fonts can be generated by making an image using LaTeX. This variable contains a (comma-separated) list of LaTeX commands for setting the style to be used when these images are made. If the value of this variable is empty then the accent is simply ignored, using an un-accented font character (not an image) instead.

Within the `color.perl` package, the following variables are used to identify the names of files containing specifications for named colors. Files having these names are provided, in the `$LATEX2HTMLSTYLES` directory, but they could be moved elsewhere, or replaced by alternative files having different names. In such a case the values of these variables should be altered accordingly.

`$RGBCOLORFILE = 'rgb.txt';`

`$CRAYOLAFILE = 'crayola.txt';`

The following variables may well be altered from the system defaults, but this is best done using a local `.latex2html-init` initialisation file, for overall consistency of style within documents located at the same site, or sites in close proximity.

$default_language = 'english'; This establishes which language code is to be placed within the <!DOCTYPE...> tag that may appear at the beginning of the HTML pages produced. Loading a package for an alternative language can be expected to change the value of this variable. See also the $TITLES_LANGUAGE variable, described next.

$TITLES_LANGUAGE = 'english'; This variable is used to specify the actual strings used for standard document sections, such as "Contents", "References", "Table of Contents", etc. Support for French and German titles is available in corresponding packages. Loading such a package will normally alter the value of this variable, as well as the $default_language variable described above.

$WORDS_IN_NAVIGATION_PANEL_TITLES = 4; Specifies how many words to use from section titles, within the textual hyperlinks which accompany the navigation buttons.

$WORDS_IN_PAGE = 450; Specifies the minimum page length required before a navigation panel is placed at the bottom of a page, when the $AUTO_NAVIGATION variable is set.

$CHILDLINE = "
<HR>\n"; This gives the HTML code to be placed between the childlinks table and the ordinary contents of the page on which it occurs.

$NETSCAPE_HTML = 0; When set, this variable specifies that HTML code may be present which does not conform to any official standard. This restricts the contents of any <!DOCTYPE...> tag which may be placed at the beginning of the HTML pages produced.

$BODYTEXT = ''; The value of this variable is used within the <BODY...> tag; e.g. to set text and/or background colors. It's value is overridden by the \bodytext command, and can be added-to or parts changed using the \htmlbody command or \color and \pagecolor from the color package.

$INTERLACE = 1; When set, interlaced images should be produced.
This requires graphics utilities to be available to perform the interlacing operation.

$TRANSPARENT_FIGURES = 1; When set, the background of images should be made transparent; otherwise it is white. This requires graphics utilities to be available which can specify the color to be made transparent.

$FIGURE_SCALE_FACTOR = 1.6; Scale factor applied to all images of figure and other environments, when being made into an image.

Note that this does not apply to recognised mathematics environments, which instead use the contents of $MATH_SCALE_FACTOR and $DISP_SCALE_FACTOR to specify scaling.

$MATH_SCALE_FACTOR = 1.6; Scale factor applied to all images of mathematics, both inline and displayed. A value of 1.4 is a good alternative, with anti-aliased images.

$DISP_SCALE_FACTOR = 1; Extra scale factor applied to images of displayed math environments. When set, this value multiplies $MATH_SCALE_FACTOR to give the total scaling. A value of '1.2' is a good choice to accompany $MATH_SCALE_FACTOR = 1.4;.

$EXTRA_IMAGE_SCALE This may hold an extra scale factor that can be applied to all generated images. When set, it specifies that a scaling of $EXTRA_IMAGE_SCALE be applied when images are created, but to have their height and width recorded as the un-scaled size. This is to coax browsers into scaling the (usually larger) images to fit the desired size; when printed a better quality can be obtained. Values of '1.5' and '2' give good print quality at 600dpi.

`$PAPERSIZE = 'a5';` Specifies the size of a page for typesetting figures or displayed math, when an image is to be generated.
This affects the lengths of lines of text within images. Since images of text or mathematics should use larger sizes than when printed, else clarity is lost at screen resolutions, then a smaller paper-size is generally advisable. This is especially so if both the `$MATH_SCALE_FACTOR` and `$DISP_SCALE_FACTOR` scaling factors are being used, else some images may become excessively large, including a lot of blank space.

`$LINE_WIDTH = 500;` Formerly specified the width of an image, when the contents were to be right- or center-justified. (No longer used.)

The following variables are used to access the utilities required during image-generation. File and program locations on the local system are established by the `configure-pstoimg` *Perl* script and stored within `$LATEX2HTMLDIR/local.pm` as *Perl* code, to be read by `pstoimg` when required.

After running the `configure-pstoimg` *Perl* script it should not be necessary to alter the values obtained. Those shown below are what happens on the author's system; they are for illustration only and do *not* represent default values.

- `$GS_LIB = '/usr/local/share/ghostscript/4.02';`
- `$PNMCAT = '/usr/local/bin/pnmcat';`
- `$PPMQUANT = '/usr/local/bin/ppmquant';`
- `$PNMFLIP = '/usr/local/bin/pnmflip';`
- `$PPMTOGIF = '/usr/local/bin/ppmtogif';`
- `$HOWTO_TRANSPARENT_GIF = 'netpbm';`
- `$GS_DEVICE = 'pnmraw';`
- `$GS = '/usr/local/bin/gs';`
- `$PNMFILE = '/usr/local/bin/pnmfile';`
- `$HOWTO_INTERLACE_GIF = 'netpbm';`
- `$PBMMAKE = '/usr/local/bin/pbmmake';`
- `$PNMCROP = '/usr/local/bin/pnmcrop';`
- `$TMP = '/usr/var/tmp';`

The following variables are no longer needed, having been replaced by the more specific information obtained using the *Perl* script `configure-pstoimg`.

`$USENETPBM = 1;`

`$PBMPLUSDIR = '/usr/local/bin';`

5.3 Extending the Translator

In Section 3.7 is was seen how the capabilities of LaTeX2HTML can be extended to cope with LaTeX commands defined in packages and style-files. This is in addition to defining simple macros and environments, using `\newcommand` and `\newenvironment`. The problem however, is that writing such extensions for packages requires an understanding of *Perl* programming and of the way the processing in LaTeX2HTML is organised.

The default behaviour for unrecognised commands is for their arguments to remain in the HTML text, whereas the commands themselves are passed on to LaTeX, in an attempt to generate an image. This is far from ideal, for it is quite likely to lead to an error in LaTeX due to not having appropriate arguments for the command.

Currently there are several related mechanisms whereby a user can ask for particular commands and their arguments to be either

- ignored (via: `&ignore_commands`);
- passed on to LaTeX for processing as an image
  (via: `&process_commands_in_tex`);
- passed to LaTeX for processing as an image, as if an environment
  (via: `&process_commands_inline_in_tex`);
- passed on to LaTeX, setting parameters to be used in subsequent images
  (via: `&process_commands_nowrap_in_tex`);
- processed as "order-sensitive", as if an environment rather than a command
  (via: `&process_commands_wrap_deferred`);

The string beginning `&....` is the name of the *Perl* subroutine that controls how the LaTeX commands are to be subsequently treated during processing by LaTeX2HTML. Towards the end of the `latex2html` script, one finds a list of LaTeX commands to be handled by each of these subroutines. These lists even include some common TeX commands.

Analogous lists occur in most of the package extension files. In many cases the commands are for fine-tuning the layout on a printed page. They should simply be ignored; but any parameters must not be allowed to cause unwanted characters to appear on the HTML pages. Customised extensions using these mechanisms may be included in the `$LATEX2HTMLDIR/latex2html.config` file, or in a personal `$HOME/.latex2html-init` initialisation file, as described next.

5.3.1 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.[47]:

```
<cmd_name1> # {} # [] # {} # [] ...
<cmd_name2> # <<pattern>> # [] ...
```

[47] 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.

`{}`'s mark compulsory arguments and `[]`'s optional ones, while `<<pattern>>` denotes matching everything up to the indicated string-pattern, given literally (e.g. `\\endarray`); spaces are ignored. Special characters such as `$`, `&`, `\` itself and perhaps some others, need to be "escaped" with a preceding `\`.

Some commands may have arguments which should be left as text even though the command should be ignored (e.g. `\hbox`, `\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 # [] # {}
linebreak # []
mbox
<add your commands here>
_IGNORED_CMDS_
```

5.3.2 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_
```

5.3.3 Handling "order-sensitive" Commands

Some commands need to be passed to LaTeX, but using the `&process_commands_in_tex` subroutine gives incorrect results. This may occur when the command is "order-sensitive", using information such as the value of a counter or a boolean expression (or perhaps requiring a box to have been constructed and saved). Try using the `&process_commands_inline_in_tex` subroutine instead. Commands declared this way are first "wrapped" within a dummy environment, which ensures that they are later processed in correct order with other environments and order-sensitive commands.

Other commands may need to be passed to LaTeX, not to create an image themselves, but to affect the way subsequent images are created. For example a color command such as `\color{red}` should set the text-colour to 'red' for all subsequent text and images. This must be sent to LaTeX so that it is processed at exactly the right time; i.e. before the first image required to be 'red' but following any images that are not intended to be affected by this colour-change. The subroutine `process_commands_nowrap_in_tex` is designed specifically to meet such requirements.

Commands can be order-sensitive without having to be passed to LaTeX. Indeed even if a *Perl* subroutine has been carefully written to process the command, it may still give wrong results if it is order-sensitive, depending on the values of counters, say. To handle such cases there is the `&process_commands_wrap_deferred` subroutine. This also "wraps" the command within a dummy environment, but when that environment is processed the contents are not sent to LaTeX, as in the previous case. All of the standard LaTeX commands to change, set or read the values of counters are handled in this way.

5.4 Customising 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 are presented in Section 5.2.4.

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

```
\htmladdtonavigation{\htmladdnormallink
  {\htmladdimg{http://server/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):

`$UP` Points up to the "parent" section;

`$NEXT` Points to the next section;

`$NEXT_GROUP` Points to the next "group" section;

`$PREVIOUS` Points to the previous 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):

`$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_TITLE` Points to the previous 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 "next group" and "previous group" are rarely used, since it is usually possible to determine which are the next/previous logical pages in a document. However these may be

needed occasionally with segmented documents, when the segments have been created with different values for the `$MAX_SPLIT_DEPTH` variable. This is quite distinct from the segmented document effect in which the first page of one segment may have its '*PREVIOUS*' button artificially linked to the first page of the previous segment, rather than the last page.
The number of words that appears in each textual link is controlled by the variable `$WORDS_IN_NAVIGATION_PANEL_TITLES` which may also be changed in the configuration files. Below is an example of a navigation panel for the bottom of `HTML` pages. (Note that the "`.`" is *Perl*'s string-concatenation operator and "`#`" signifies a comment.)

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

Note that extra buttons may be included by defining suitable code for the container `$CUSTOM_BUTTONS`. The use of explicit 'newline' (`\n`) characters is not necessary for the on-screen appearance of the navigation panel within a browser window. However it maintains an orderly organisation within the `.html` files themselves, which is helpful if any hand-editing is later required, or simply to read their contents. The corresponding subroutine for a navigation-panel at the top of a page need not use the rule `<HR>`, and would require a break (`<BR>`) or two at the end, to give some visual separation from the following material.

6 Known Problems

Here are some of the problems that were known to exist with previous versions of LaTeX2HTML. Most of those that were real errors are either fixed completely in more recent versions (v98.1 and later), or are much less likely to occur within correct LaTeX source. (Some are not really errors but indications of poor style in the user's choices among various ways to organise their source code.) Several are indeed limitations inherent in the way LaTeX2HTML currently performs its processing.

Unrecognised Commands and Environments: Unrecognised commands are ignored and any arguments are left in the text. Unrecognised environments are passed to LaTeX and the result is included in the document as one or more inlined images.
There are very few standard LaTeX commands that are not recognised. Many common TeX commands are recognised also, even though not explicitly mentioned in the LaTeX blue book[1]. Any aberrant commands should be reported to the LaTeX2HTML *mailing list, see Section 2.2.*

Index: The translator generates its own index by saving the arguments of the `\index` command. The contents of the theindex environment are ignored. When using the makeidx package, very sophisticated Indexes can be built automatically. The Index for this manual is a good example.

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. Stylistically it is bad to declare new environments or theorems outside of the document preamble, so this should not cause any real problems. Changes to commands using `\def` or `\renewcommand` should usually be made only locally, within special environments, to set a needed parameter; e.g. a basic length in a picture environment. But when such environments force an image to be generated, then LaTeX will make the correct redefinition.

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.
This is inherent to the way LaTeX2HTML *does its processing. It will not be different unless later versions change this strategy; (e.g. if* LaTeX2HTML-*NG ever becomes fully integrated.)*

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`, `pstoimg` 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 `pstoimg`, `texexpand` and `latex2html` are executable; if necessary use the Unix `chmod` command to make them executable.

It just stops. Check the style files that you are using. It is possible 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 environment (see Section 4.4).

The `$VERBOSITY` variable can be used to create tracing messages, which may help to locate which command or environment was being processed when everything stopped.

It appears to be doing nothing. Perhaps the processor has fallen into an unending loop. Usually there will be a bad definition, or other faulty source code, which has caused this. The `$VERBOSITY` variable (see page 71)can be set to generate tracing messages, which may help to locate which command or environment is being processed repeatedly. Try setting a value of '3'; e.g. using the commandline switch '`-verbosity 3`'. This will print command and environment names, as thaey are processed. It should soon become apparent where any such looping occurs.

It just fills the endlessly with dots. No 'perhaps' here; the processor has definitely fallen into an unending loop. See the previous item for how to detect which command or environment is causing the problem.

***Perl* cannot parse the `latex2html` script:** If *Perl* refuses to start LaTeX2HTML and issues errors, your *Perl* version is not up to date. Update your *Perl* to 5.003 or later. You can check which version of *Perl* you are using by invoking *Perl* with the '`-v`' option.

If *Perl* issues errors during runtime, this is most probably related to bugs within LaTeX2HTML or one of its modules. In this case you will need help from the developers or experienced users; this can be obtained via the discussion list.

It crashes (dumps core) as soon as it starts: Update your *Perl* to 5.003 or later.

It does not show any of your images: You can't run LaTeX2HTML in a subdirectory that contains a dot within the directory name, such as `latex2html-98.1`, or in name of any higher directory. This is because `dvips`'s `-o` option will change 98.1 into 98.001 and use that as the resulting output file, instead of image001 . The PostScript files will be placed higher up in the directory tree.

For instance, if `pwd` returns something like:

```
/usr/people/nelson/files/latex2html-98.1/work/tests
```

and you run LaTeX2HTML, then `dvips` will generate image output files here:

```
/usr/people/nelson/files
```

called `latex2html-98.001`, `latex2html-98.002`, ...instead of `image001`, `image002`, `image003`, ...in the subdirectory where your `.html` files were created. As a result the images will not show in your documents.

If you are getting 'File Not Found' errors, then reprocess your job using the '`-debug`' switch, to see what options are passed to `dvips`. If there is a '.' in some parts of any directory name, then look above that directory to see if files are being generated there.

One obvious fix is to rename the offending directory to remove the '.' from its name.

If that is not possible, then define an alternative location for image generation to take place; set `$TMP` to contain the name for this location. Typically `$TMP = '/usr/tmp';` . (This use of `$TMP` is a good thing to do anyway, especially if your Unix account is subject to quota limitations.)

It stops after having run LaTeX, displaying a message about `dvips`: See the previous item for the reason and 'fix'. This version of LaTeX2HTML detects the problem and will not allow images to be generated in the wrong place.

`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: Try splitting your source file into more than one file, using the LaTeX commands `\input` or `\include`. Also, try using the '`-no_images`' option.

Perhaps the processor has fallen into an infinite loop. Usually there will be a bad definition, or other faulty source code, which has caused this. See an earlier problem for how to set the `$VERBOSITY` variable to help locate the bad code leading to this memory exhaustion.

As a last resort you may consider increasing the virtual memory (swap space) of your machine.

`install-test` issues "dbm" related error messages:
LaTeX2HTML requires a DataBase Management system (*NDBM*, *GDBM*, or *SDBM*) in order to run. This is usually part of each Unix-like operating system and *SDBM* is part of *Perl* 5, but sometimes this is either missing on your operating system or left out in a binary *Perl* distribution. Use *Archie*[48] to find one or (better) update to a complete *Perl* 5 distribution.

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

`latex2html` **issues "dbm" related error messages:** If you get warnings like

```
ndbm store returned -1, errno 28, key "xyz" at latex2html line 123
```

this is related to an overflow of LaTeX2HTML internals. You will need help from the list, here.

If you get real error messages which cause LaTeX2HTML to abort, run `install-test` to check if your DataBase management works. You will probably need to re-install *Perl* 5 (see above topic).

This can happen when an image is being created from a large piece of LaTeX source code. The image-reuse mechanism uses the code itself to construct a database key. If too long, the key is invalid and may crash *DBM* or *NDBM*. (In fact this error should no longer occur in V97.1, so please advise the LaTeX2HTML developers if you get error messages of this kind.) The message should contain the name of environment which caused the problem, along with an identifying number; e.g. `eqnarray268`. To find which exact piece of code this represents, run LaTeX2HTML again, using the '`-debug`' switch. Then look at the files in the `TMP` subdirectory of the working directory named `TMP/part001`, `TMP/part002`, etc. Use the unix `grep` command: `grep 268 <dir>/TMP/part*` to find that number in these files. This should enable you to locate exactly where the problem occurs.

One solution may be to wrap the whole environment within `\begin{makeimage}` and `\end{makeimage}`. This will still cause the image to be created, but uses just the environment name and number as the database key.

The `\verb"ABC"` command doesn't work: This is an unfortunate bug which can be avoided by using any character other than quotes; e.g. `\verb+ABC+`.

Cannot get the "tilde" (˜) to show: The trick here is to use `\~{}`. Alternatively try using something like: `\htmladdnormallink{mylink}{http://host/\%7Eme/path/file.html}`.

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

Try using `\char126` . In any case tildes within `\htmladdnormallink` and similar commands should be handled correctly.

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.

`\input` **commands:** There is a bug in the expansion of `\input` commands which may cause 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 should no longer cause problems (actually since 97.1). `\input` commands are also handled correctly within comment environments declared using `\excludecomment` and when loading multiple `.aux` files, due to use of the `\include` command. Alternatively you might want to use either the verbatim or the verbatimfiles package.

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` may be undefined. This is not always fatal but if you have problems you can use full path-names 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: There are several known ways that this may occur.

- Perhaps 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. '`a4`', '`a3`' or even '`a0`') instead of the default ('`a5`') using the LaTeX2HTML variable `$PAPERSIZE` in the file `latex2html.config`.

  This error should no longer occur, with V97.1 *. Please report it on the mailing-list, if it does.*

- More likely is that some inappropriate LaTeX code has caused an error, resulting in an extra page (or pages) being generated. Examine the `images.log` file, to see if it reports any LaTeX errors.
- A much rarer reason 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";` within the file `latex2html.config`.
- Yet another reason for images appearing out of place, especially while developing a document, is that the browser's image cache is providing out-of-date versions rather than getting the latest version afresh. When this occurs there will often

be images stretched or shrunk to fit the wrong sized imaging area; this symptom is browser-dependent. Flushing the cache, then reloading the `HTML` document, should clear up the problem.

Unacceptable quality of converted images: Try changing the size of the image (see Section 3.4).

The bibliographic references are missing: Run `latex` and then `bibtex` on the original source file in order to generate a `.bbl` file. LaTeX2`HTML` 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 [49].

Some of the fonts are translated incorrectly: There is a fault in way the LaTeX scoping rules have been interpreted in LaTeX2`HTML`. Consider this:

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

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 LaTeX2`HTML` 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.
```

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

Nesting of font changing commands is now handled correctly. Such problems should no longer occur, though it always helps to use explicitly delimited 'pseudo'-environments; e.g. `{\bf ... }`, or LaTeX's commands requiring an explicit argument, such as `\textbf`.

Cannot get it to generate inlined images: Run LaTeX2HTML on your document, using the '`-debug`' switch. Look in the directory where the HTML files are generated for two files named `images.tex` and `images.log`. Examine their contents. Do you notice anything unusual in them?

Copy `images.tex` into the directory of your original LaTeX file and run `latex` on `images.tex`. Can you see any errors in `images.log`? If so, 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, then run LaTeX2HTML on the original document using the '`-images_only`' switch.

However if you make changes or additions to the original source then the same problems may occur again. Thus it is better to understand why the changes were required and alter your document's source code appropriately.

If you get into a mess delete all the image files and run LaTeX2HTML again. Often it is sufficient to just delete the file `images.pl`.

If you *still* get into a mess, try running LaTeX2HTML 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 LaTeX2HTML again with the option '`-images_only`'.

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 `\documentclass` command with something like `{article}` just before using LATEX2`HTML`. 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*[50] to find this or any other style files).

References

[1] Leslie Lamport, LATEX,*A Document Preparation System.* User's Guide & Reference Manual, 2nd edition. ISBN 0-201-52983-1, Paperback 256 pages, Addison–Wesley, 1994. Online information on TEX and LATEX is available at `http://www.tug.org/` and `http://www.dante.de/` .

[2] Michel Goossens, Frank Mittelbach, Alexander Samarin, *The* LATEX *Companion* ISBN 0-201-54199-8, Paperback 530 pages, Addison–Wesley, 1994.

[3] Michel Goossens, Sebastian Rahtz and Frank Mittelbach, *The* LATEX *Graphics Companion.* ISBN 0-201-85469-4, Softcover 608 pages, Addison–Wesley, 1997.

[4] Sebastian Rahtz and Michel Goossens, with E. Gurari, R. Moore & R. Sutor. *The* LATEX *Web Companion.* to appear in 1999, Addison–Wesley.

[5] Nikos Drakos, Text to Hypertext conversion with LATEX2`HTML`. *Baskerville*, December 1993, Vol. 3, No. 2, pp 12–15. `http://cbl.leeds.ac.uk/nikos/doc/www94/www94.html`

[6] Nikos Drakos, From Text to Hypertext: A Post-Hoc Rationalisation of LATEX2`HTML`. Published in "Proceedings of the 1st World Wide Web Conference", May 1994, CERN, Geneva, Switzerland. `http://cbl.leeds.ac.uk/nikos/doc/www94/www94.html`

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

Glossary of variables and file-names

`.dvipsrc` initialisation file for `dvips` 7, 11
`.latex2html-init` initialisation file for LaTeX2HTML 11, 62, 63, 67, 72, 74, 77, 78, 86, 91
`$ACCENT_IMAGES` holds the LaTeX style to be used when making images of accented characters which are not present in the character encoding being used 74
`$ADDRESS` contains information to be included as a signature at the bottom of each HTML page 67
`$ALTERNATIVE_ICONS` contains the location of special icons, other than the standard ones 67, 72
`$ANTI_ALIAS` when set, specifies that anti-aliasing be used with generated images of figures and included images 69
`$ANTI_ALIAS_TEXT` when set (the default), specifies that anti-aliasing be used with generated images of maths and typeset text 69
`$ASCII_MODE` when set, all icons and images are replaced with text strings, suitable for text-only browsers 68
`$AUTO_NAVIGATION` when set, a navigation panel is placed at the bottom of long pages, unless overridden by `$NO_NAVIGATION` being set also 70, 75
`$AUTO_PREFIX` if set, all files produced have a filename-prefix using the name of the LaTeX file being processed 11, 57, 64
`$BODYTEXT` string to be used within the `<BODY...>` tag on HTML pages 75
`$BOTTOM_NAVIGATION` when set, a navigation panel is placed at the bottom of every HTML page, unless overridden by `$NO_NAVIGATION` being set also 70
`$CHARSET` specifies the character set used within the HTML pages, currently this defaults to `"iso_8859_1"` (ISO–Latin–1) 74
`$CHILDLINE` has piece of HTML code to be inserted between the child-links table and page contents 75
`$CONTENTS` when navigation panels are created, this holds a hyperlink to the HTML file that has the Table-of-Contents 79
`$CONTENTS_IN_NAVIGATION` when set, an iconic link to the Table-of-Contents is placed in navigation panels 70
`$CRAYOLAFILE` when `color.perl` has been loaded, holds the name of the file which contains named color specifications for crayon colors 74
`$CUSTOM_BUTTONS` variable available to link to customised buttons in navigation panels 80
`$CUSTOM_TITLES` when set, a customised subroutine is used to determine the filename for each HTML page 63
`$DEBUG` when non-zero, LaTeX2HTML shows messages that would otherwise be ignored 71
`$DESTDIR` specifies the directory into which all the generated files will be placed . . 63
`$DISCARD_PS` when set, specifies that the temporary PostScript files created by `dvips` be discarded immediately after use 68
`$DISP_SCALE_FACTOR` extra magnification for images of displayed math environments; multiplies the `$MATH_SCALE_FACTOR` to give the total scaling 19–22, 66, 75, 76
`$DONT_INCLUDE` holds a list of filenames and filename extensions to *not* include, even if requested; filenames ending with `.sty` are automatically excluded . . 73, 74, 82, 89
`$DO_INCLUDE` holds a list of filenames and filename extensions to be included, even when the `$DONT_INCLUDE` variable would normally exclude them 74
`$DVIPS` contains the command to be used when the `dvips` program processes `images.dvi` to create PostScript files describing the required images 73
`$DVIPS_MODE` contains the *Metafont* mode to be used when the `dvips` program supports

`$depth` holds the value of the sectioning depth when the `<ADDRESS>` tag of an `HTML` page is being created 67

`$file` holds the name of the `HTML` page when its `<ADDRESS>` tag is being created . . 67

`$title` holds the title of the current section when the `<ADDRESS>` tag of an `HTML` is being created 67

`amsfonts.perl` contains code to recognise or implement features of the amsfonts package for LaTeX 29

`amstex.perl` contains code to recognise or implement features of the amstex package for LaTeX 29

anonymous ftp pseudonym for use of the *ftp* program as the `anonymous` user, for file-transfers 6

Archie publicly-accessible database of files and programs available via *anonymous ftp* or using the Web[51] 6, 8, 83, 88

Arena Web-browser[52]; currently the only one which displays `HTML` 3.1 math markup 15

`article.perl` implements section-numbering conventions, etc. for the *article* document-class 28

`bash` a unix shell, does not directly support use of LaTeX2`HTML` 8

`bibtex` bibliography data-base program, used with LaTeX 86

`book.perl` implements section-numbering conventions, etc. for the *book* document-class 28

CERN originally named “Conseil Europeen pour la Recherche Nucleaire”, now known as the “European Laboratory for Particle Physics”[53], located near Geneva; also where the World-Wide Web[54] was originally conceived 55

`changebar.sty` LaTeX 2_ε package to mark changes in documentation; this supercedes an earlier package . . . . iii, 12, 31

`Changes` text file, recording the changes made to LaTeX2`HTML` since it was first released in July 1993, until V96.1 and revisions during 1996 . . . . 7, 13

`chmod` Unix command for changing the access-privileges on files 82

`color.perl` contains *Perl* code to implement all the features of LaTeX's color package and loads a list of named colors 54, 74, 89, 93

`configure-pstoimg` *Perl* script to locate which Unix utilities are available on the local system, for use in generating images . . . . v, 7, 10, 76, 91, 96

`credits.tex` input file for the LaTeX2`HTML` manual; acknowledges the multitude of contributors to the development of LaTeX2`HTML` 12

`csh` a unix shell, provides support for LaTeX2`HTML` 8

DBM Unix Database Management utility 8, 84

`dn.gif` image of a small downward-turning arrow 12

`docs/` subdirectory of `$LATEX2HTMLDIR` containing the LaTeX2`HTML` documentation and a `Makefile` to control its processing 7, 12, 98

`docs/foilhtml/` subdirectory of `$LATEX2HTMLDIR` containing an implementation for the FoilTeX system 7, 31

`dot.latex2html-init` example initialisation file for LaTeX2HTML; usually to be found in the `$LATEX2HTMLDIR` 7, 11

[51] http://www.pvv.unit.no/archie/
[52] http://www.w3.org/hypertext/WWW/Arena
[53] http://www.cern.ch/Welcome.html
[54] http://www.w3.org/hypertext/WWW/TheProject.html

`Makefile` contains the time-dependency data used by the `make` utility . 59, 63, 70, 71, 94, 108

`makeidx.perl` adds *Perl* code to interpret index-entries otherwise intended for the `makeindex` program . . . . ii, 34

`makeindex` utility for sorting and filtering textual data; used primarily for making the index or glossary 12, 35, 96, 97

`makemap` *Perl* script to generate image-maps 7, 10, 11, 55, 98

`makeseg` *Perl* script to generate Makefiles for segmented documents, so that you can initiate the translation with a single `make` call. . . . . iii, 7

`MakeTeXPK` script used by `dvips`, and other `dvi`-drivers, controlling generation of missing fonts 11, 73

`manhtml.tex` input file containing definitions needed for segments of the `HTML` version of the LaTeX2`HTML` manual 12

`manual.aux` auxiliary file created by running LaTeX on the LaTeX2`HTML` manual . . . 12

`manual.glo` collects (unsorted) glossary entries for the LaTeX version of the LaTeX2`HTML` manual 34, 35, 97

`manual.gls` file containing the sorted entries for the glossary in the LaTeX2`HTML` manual; created by running `makeindex` on `manual.glo` using `l2hglo.ist` 12, 35

`manual.idx` collects (unsorted) index entries for the LaTeX version of the LaTeX2`HTML` manual 97

`manual.ind` file containing the sorted entries for the index in the LaTeX2`HTML` manual; created by running `makeindex` on `manual.idx` using `l2hidx.ist` 12

`manual.tex` main LaTeX source for the LaTeX2`HTML` manual 12

`Metafont` generates bitmapped fonts for TeX and LaTeX 73, 89, 91, 97, 99

`misc/` subdirectory containing miscellaneous files used by *Metafont*, often the location for `modes.mf` 73

`modes.mf` used by *Metafont* to customise font bitmaps for different printers . 11, 73, 97

`Mosaic` commonly-used Web-browser, see *NCSA Mosaic* below 88

`mybutton.gif` file-name used in example code for an image as a button to a hyperlink 79

`natbib.perl` implements the commands in the `natbib` package, allowing flexible styles for citations 49

`NCSA` "National Center for Supercomputing Applications" located at University of Illinois at Urbana–Champaign 55

`NCSA Mosaic` Web-browser, freely available[55] for most platforms 61

`NDBM` alternative Database Management utility for Unix 8, 83, 84

`netpbm` suite of Unix graphics utilities[56]; alternative to `pbmplus` . . . . x, 8, 9

`Netscape Navigator` popular Web-browser, commercially available[57] for most platforms and free to academic institutions 9, 15, 61

`Override.pm` *Perl* input file for operating system specific constructs like directory delimiter, getenv()/setenv(), and other system calls and quirks . . . . vii, 93

`overview.tex` input file for a section of the LaTeX2`HTML` manual; contains a brief summary of LaTeX2`HTML`'s capabilities 12

`pbmmake` Unix graphics utility which creates rectangles of a single color as a bitmapped graphic 92

---

[55] http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html

[56] ftp://ftp.x.org/contrib/utilities/

[57] http://home.netscape.com

`pbmplus` suite of Unix graphics utilities[58] . . . 9, 97

Perl the Unix program[59] which interprets LaTeX2HTML and orchestrates the whole conversion process . . . i

`PNG` Portable Network Graphics format, devised as an alternative to GIF with extra capabilities . . . v, 10, 98

`pnmcat` Unix graphics utility which concatenates several bitmapped graphics to produce a single bitmapped graphic . . . 92

`pnmcrop` Unix graphics utility which crops whole rows and/or columns of a single color, from specified edges of a bitmapped graphic . . . 8, 9, 92

`pnmfile` Unix graphics utility which reads a graphic in one of a large range of formats, creating a bitmap suitable for input to other utilities . . . 92

`pnmflip` Unix graphics utility which rotates and reflects through multiples of 90°. 22, 92

`pnmtopng` Unix graphics utility which converts to `PNG` format. . . . 9

`PostScript` graphics programming language used in printers; interpreted by *Ghostscript* . . . ii, 9, 11, 12, 21, 62, 66, 68, 69, 73, 83, 85, 89, 92, 93, 95, 98, 104

`ppmquant` Unix graphics utility used to limit the number of distinct colours in a single image . . . 92

`ppmtogif` Unix graphics utility used to convert a bitmapped graphic in `ppm` format into GIF format . . . 92

`problems.tex` input file for the LaTeX2HTML manual; contains a description of known problems with LaTeX2HTML, and how best to avoid or deal with them . . . 12

`psfiles/` sub-directory of `docs/` containing PostScript graphics used in the printed version of the LaTeX2HTML manual . . . 12

`pstogif` *Perl* script which controls the conversion of images from PostScript to GIF bitmaps; replaced by `pstoimg` for V97.1 . . . ii, v, 7, 10

`pstoimg` *Perl* script which controls the conversion of images from PostScript to a chosen graphics format . . . v, 7, 10, 73, 76, 81, 82, 90, 92, 93, 96, 98

`pstoppm.ps` contains PostScript code needed for image conversion with versions of *Ghostscript* earlier than 3.0 . . . ii

`README` found in `$LATEX2HTMLDIR`, this file contains simple information concerning the latest release . . . 7

`report.map` example data-file used as input for `makemap` to produce an image-map 55

`report.perl` implements section-numbering conventions, etc. for the *report* document-class . . . 28

`report.ps` PostScript version produced by LaTeX, for the `report.tex` example of a segmented document . . . 59

`report.tex` example source for a segmented document . . . 58–60, 98

`report/` subdirectory created when processing the example segmented document . 60

SDBM Unix Database Management utility distributed with *Perl* 5 . . . 83

`sec1.tex` one section of the example segmented document . . . 58–60

`sec2.html` one `HTML` page produced with the example segmented document . . . 59

`sec2.tex` another section of the example segmented document . . . 58, 59

`slides.perl` implements section-numbering conventions, etc. for the *slides* document-class . . . 28

`styles/` subdirectory of `$LATEX2HTMLDIR` containing `<`*`file`*`>.perl` files corresponding to LaTeX packages . . . 7, 16, 28, 72, 91

[58] ftp://ftp.x.org/contrib/utilities/
[59] ftp://ftp.uu.net/languages/perl/

Index