The Hyperlatex Markup Language

Otfried Schwarzkopf

Vakgroep Informatica, Universiteit Utrecht,
Postbus 80.089, 3508 TB Utrecht, the Netherlands
otfried@cs.ruu.nl

Januari 31, 1995

1 Introduction

Hyperlatex is a little package that allows you to use LaTeX to prepare documents in HTML (the *hypertext markup language* used by the *world wide web*), and, at the same time, to produce a fine printed document from your input. You can use all of LaTeX's power for the printed output, and you don't have to learn a new language for creating hypertext documents.

Using Hyperlatex is easy. You create a file *document.tex*, say, containing LaTeX commands — the same commands you are used to — plus a few additional directives controlling conversion to HTML. If you use the command

```
latex document.tex
```

then your file will be processed by LaTeX, resulting in a DVI-file, which you can print as usual.

On the other hand, you can run the command

```
hyperlatex document.tex
```

and your document will be converted to HTML format, presumably to a set of files called *document.html*, *document_1.html*, ... (These files are created in a separate directory which you can specify within the source file using the `\htmldirectory` command.) You can then use any HTML-viewer or WWW-browser, such as `Mosaic`, to view the document.

This document describes how to use the Hyperlatex package. It tells you the *mechanics* of setting up an input file for LaTeX and HTML, and discusses the subset of LaTeX commands which are understood and converted to HTML tags by the `hyperlatex` converter. This manual does not explain *what* to write in a WWW-document. There are style guides available, which you might want to consult. Writing an on-line document is not the same as writing a paper. I hope that Hyperlatex will help you to do both properly.

We assume that you are familiar with LaTeX, and that you have at least some familiarity with hypertext documents — that is, that you know how to use one of the WWW-browsers and understand what a *hyperlink* is.

If you want, you can have a look at the TeX source. You can use it as a template in writing your own documents, and illustrates some points discussed here.

While writing and testing Hyperlatex, I have converted several LaTeX documents into Hyperlatex format. It turns out that it takes only a few minutes for a document that does not use much mathematics or that defines lots of its own commands. One example I used was *A few rules from 'A Handbook for Scholars'* by Mark de Berg. A big document written with Hyperlatex is the *Ipe Manual*, which has about 50 pages in the printed version and 70 nodes as a HTML-document. Others at our department have used Hyperlatex, for instance to put the department's study guide (more than 200 nodes) on the world wide web.

If you have used Hyperlatex to make some document available on the world wide web, I would be happy to hear about it. I would certainly like to set up a list with demo documents.

A final footnote: The converter to HTML implemented in Hyperlatex is written in GNU Emacs Lisp. You can use it directly from Emacs. But even if you don't use Emacs, even if you don't like Emacs, or even if you subscribe to `alt.religion.emacs.haters`, you can happily use Hyperlatex. Hyperlatex can be invoked from the shell (as shown above), and you will never know that Emacs is responsible for the finely formatted document which you get.

The Hyperlatex code is based on the Emacs Lisp macros of the `latexinfo` package.

2 Using Hyperlatex

Using Hyperlatex is easy. You write your document in a LaTeX-file *document.tex*, using a certain subset of LaTeX commands, and some additional commands that control the conversion to HTML, and to make hyperlinks between parts of your document.

To make a printed document, you then run LaTeX on your file, like usual, and follow the standard procedures for printing the DVI-file.

To turn your document into HTML format, you can either run the `hyperlatex` shell script, which invokes Emacs and runs the conversion macros, or you can do conversion directly from Emacs. The `hyperlatex` script takes the following arguments:

```
hyperlatex files
```

You have to specify the full filenames, including the extension *.tex*.

To run conversion from within Emacs, put the following line in you *.emacs* file:

```
(autoload 'hyperlatex-format-buffer
                                "hyperlatex1")
```

Then you can call `hyperlatex-format-buffer` in the buffer containing the LaTeX input file. But note that the shell script version produces better error messages.

A typical HTML document consists of a set of files. In HTML-speak these files are also called 'documents'. In this manual we take the LaTeX point of view, and call 'document' what is enclosed in a `document` environment. We will use the term *node* for the individual files of the HTML document.

The node files are created in a directory which you have to specify in the preamble of your source file. You also have to specify the *base name* of the HTML-document:

```
\htmldirectory{directory}
\htmlname{basename}
```

The actual files created by `hyperlatex` are called *directory/basename.html*, *directory/basename_1.html*, *directory/basename_2.html*, and so on. The filename can be changed for individual nodes using the `\xname` command.

The entry point for your document will be the file *directory/basename.html*. This means that you can view your HTML-document using `Mosaic` as follows.

```
Mosaic directory/basename.html
```

3 Controlling the conversion to Html

Hyperlatex automatically partitions the document into several nodes. This is done based on the LaTeX sectioning. The section commands `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph` are assigned levels 1 to 6. (If you use the *article* document style, `\section` to `\subparagraph` are given levels 1 to 5, as there are no chapters).

The `\htmldepth` command in the preamble determines at what depth separate nodes are created. The default setting is 4, which means that (for *article* style) sections, subsections, and subsubsections are given their own nodes, while paragraphs and subparagraphs are put into the node of their parent subsection. You can change this by putting

```
\htmldepth{depth}
```

in the preamble. A value of 1 means that the full document will be stored in a single file.

A HTML file needs a *title*. This *must be set* in the preamble using the `\htmltitle` command. Use something short but helpful. The title you specify is used directly for the top node of your document. The other nodes get a title composed of this and the section heading.

It is common practice to put a short notice at the end of every HTML node, giving a reference to the author. This can be done by using the `\htmladdress` command in the preamble.

The individual nodes of a HTML document are linked together using *hyperlinks*. Hyperlatex automatically places buttons on every node that link it to the previous and next node of the same depth, if they exist, and a button to go to the parent node.

Furthermore, Hyperlatex automatically adds a menu to every node, containing pointers to all subsections of this section. (Here, 'section' is used as the generic term for chapters, sections, subsections, ...) This may not always be what you want. You might want to add nicer menus, with a short description of the subsections. In that case you can turn off the automatic menus by putting

```
\htmlautomenu{0}
```

in the preamble. On the other hand, you might also want to have more detailed menus, containing not only pointers to the direct subsections, but also to all subsubsections and so on. This can be achieved by putting

```
\htmlautomenu{depth}
```

in the preamble, where *depth* is the desired depth of recursion. The default behavior corresponds to a *depth* of 1.

A final note: All these commands must start at the beginning of a line, if you want Hyperlatex to see them.

4 Parsing by LaTeX and Hyperlatex

You are writing an input file that has to be read by LaTeX as well as the Hyperlatex converter. The parsing done by LaTeX is complex, and has many of us surprised in certain situations. It was hopeless to try to imitate this complex behavior using a modest collection of Emacs Lisp macros. Nevertheless, Hyperlatex should behave well on your LaTeX files. If your source is comprehensible to LaTeX (with the *hyperlatex.sty* package), then Hyperlatex should not have *syntactical* problems with it. There is, however, one difference in parsing arguments: In LaTeX, you can write

```
\emph example,
```

and what you will get is '*e*xample'. Hyperlatex will complain about this. To get the same effect, you will have to write

```
\emph{e}xample.
```

Hyperlatex has been designed to understand a certain subset of LaTeX. It will treat all other LaTeX instructions with an error message. This does not mean that you should not use any of these instructions for getting exactly the printed document that you want. By all means, do. However, I felt it was safer if Hyperlatex did not ignore any commands it doesn't know. So you will have to hide those commands from Hyperlatex using the escape mechanism.

Here is what your input file should roughly look like:

```
\documentclass{article}
\usepackage{hyperlatex}

\htmldirectory{HTML directory}
\htmlname{base filename of HTML nodes}
```

```
\htmltitle{title of HTML nodes}
\htmladdress{otfried@cs.ruu.nl}

.... more LaTeX declarations, if you want

\title{Title for LaTeX document}
\author{Author for LaTeX document}

\begin{document}

\maketitle
\section{Introduction}

\topnode{Welcome to this HTML Document}

This is the beginning of the section
titled 'Introduction' in the printed
manual, and at the same time the
beginning of the top node of the HTML
document....
```

If you are still using LaTeX2.09, replace the first two lines by

```
\documentstyle[hyperlatex]{article}
```

Note the use of the *hyperlatex* package. It contains the definitions for some LaTeX extensions useful in Hyperlatex, and also turns on the special input mode.

For the HTML document, Hyperlatex ignores everything before the line starting with `\topnode` (there may only be white space on this line before the command). The `\topnode` command specifies the heading for the *top node* of the HTML document. It does not produce any output in the printed manual.

5 A LaTeX subset — Getting started

Starting with this section, we take a stroll through the LaTeX-book [1], explaining all features that Hyperlatex understands, additional features of Hyperlatex, and some missing features. For the LaTeX output the general rule is that *no LaTeX command has been changed*. If a familiar LaTeX command is listed in this manual, it is understood both by LaTeX and the Hyperlatex converter, and its LaTeX meaning is the familiar one. If it is not listed here, you can still use it by escaping into TeX-only mode, but it will then have effect in the printed manual only.

5.1 Hyperlatex input mode

There are ten characters that LaTeX treats as special characters, which means that they do not simply typeset themselves:

```
# $ % & ~ _ ^ \ { }
```

Hyperlatex has only five special characters:

```
\ { } % ~
```

The remaining five characters are not special in Hyperlatex. They simply typeset themselves. To typeset one of the special characters, use

```
\=   \{   \}  \%  \sim
```

Note that `\{`, `\}`, and `\sim` exist in LaTeX, but only work in math mode. These, and the two shortcuts `\=` and `\+` are actually the only LaTeX commands whose definition have been changed. You can use `\back` as a synonym for `\=`, to typeset a backslash.

If you need the special meaning of one of LaTeX's special characters, you need to use an escape to LaTeX. The Hyperlatex input mode is turned on by `\begin{document}`. This means that you can still use the regular LaTeX special characters in the preamble. (For technical reasons the special input mode is turned on by `\topnode` if you are using LaTeX2.09.)

We said above that the remaining characters typeset themselves. This has to be taken with a grain of salt. LaTeX still obeys ligatures, which turns `ffi` into 'ffi', and some characters, like `>`, do not resemble themselves in some fonts (`>` looks like > in roman font). The only characters for which this is critical are `<`, `>`, and `|`. Better use them in a typewriter-font (this includes the `example` and `verbatim` environments and the `\code` and `\kbd` fonts). Note that `?‘` is a ligature even in `\typew` font and displays as `?‘`, but displays correct in the other (logical) fonts listed above.

Like LaTeX, the Hyperlatex converter understands that an empty line indicates a new paragraph. You can achieve the same effect using the command `\par`.

5.2 Dashes and Quotation marks

Hyperlatex translates a sequence of three dashes `---` into two dashes `--`. The quotation mark sequences `''` and `‘‘` are translated into simple quotation marks `"`.

5.3 Simple text generating commands

The following simple LaTeX macros are implemented in Hyperlatex:

- `\LaTeX` produces LaTeX.
- `\TeX` produces TeX.
- `\LaTeXe` produces LaTeX2$_\varepsilon$.
- `\copyright` produces ©.
- `\ldots` produces three dots . . .
- `\minus` produces a minus sign −.
- `\quad` and `\qquad` produce some empty space.
- `\ss` produces ß.
- `\today` produces 17th May 1995— although this might depend on when you use it . . .

5.4 Emphasizing Text

Hyperlatex understands the following physical font specifications of LaTeX2$_\varepsilon$:

- `\textbf` for **bold**
- `\textit` for *italic*
- `\textsc` for SMALL CAPS
- `\texttt` for `typewriter`
- `\underline` for underline

Note that these font changes are properly cumulative in LaTeX2$_\varepsilon$ and in the `netscape` browser, but are not in LaTeX2.09 and in older HTML browsers. The following commands are supported for backwards compatibility:

- \bf and \bold for **bold**
- \it and \italic for *italic*
- \scap for SMALL CAPS
- \typew for `typewriter`

So you can write

```
{\it italic text}
```

but also

```
\textit{italic text}
```

Note that if you use the old LaTeX versions \it and \bf, the command must come directly after an opening brace. You may **not** write

```
{roman text \it italic text}
```

The HTML guidelines encourage you to think in terms of *logical concepts* instead of physical fonts. So, do not write \textit{filename}, but write \file{filename}. This has the advantage that the reader of the document can still decide how she wants the logical concept 'filename' to be rendered (for instance in a light green cyrillic font, if she wants). Here are the logical fonts available in HTML:

- \cit for *citations*.
- \code for `code`.
- \dfn for *definitions*.
- \em and \emph for *emphasized text*.
- \file for *filenames*.
- \kbd for `keyboard input`.
- \samp for `sample input`.
- \strong for **strong emphasis**.
- \var for *variables*.

Finally, you can use 20\dmn{in} to typeset the dimension 20 in.

You can use \/ to separate slanted and non-slanted fonts (it will be ignored in the HTML-version).

5.5 Changing Type Size

LaTeX's declarations for changing the type size are all understood, and HTML-tags are generated for them. Note, however, that currently only the `netscape` browser interprets these size changes, other browsers will simply ignore them. The commands are \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge, and \Huge. As the \it command, these commands have to immediately follow an opening brace. So you can write

```
{\large larger text},
```

but you may **not** write

```
{normal text \large larger text}
```

In the HTML version, these font sizes are relative to the node's basefont size (\normalsize being the basefont size, \large being the basefont size plus one etc.) To set a node's basefont size, you can use the command

```
\html{basefont size=x}
```

where *x* is a number between 1 and 7.

5.6 Preventing line breaks

The ~ is a special character in Hyperlatex, and is replaced by a HTML tag for 'non-breakable space'. It seems, however, that the current `Mosaic` version does not honor this, and simply treats it as a space. Nevertheless, ~'s are useful for the printed document.

5.7 Footnotes

are not yet implemented.

5.8 Formulas

There is no *math mode* in HTML, and all commands related to it are rejected. It is probably useless to try to convert a paper with a lot of mathematics into HTML format anyway.

However, sometimes you want to include simple expressions like 'the segment from point p to point q' or 'Pythagoras' theorem states that $a^2 + b^2 = c^2$.' In such cases you would like to have the properly formatted version of the formula in the printed document, and some approximation in the HTML-version. This can be done with the new \math command:

```
\math{argument}
```

In LaTeX, this command typesets the *argument*, which is read in *math mode* with all special characters enabled. Hyperlatex simply typesets the *argument* without any special treatment (but embedded commands are expanded). Often the LaTeX math expression does not look good when put into the HTML-document untreated, contains unknown commands, or you simply want something different. You might, for instance, want to typeset the ith element (`the \math{i}th element`) of an array as a_i in LaTeX, but as `a[i]` in HTML. This can be done with the optional argument of \math:

```
\math[HTML-version]{LaTeX-version}
```

In this example: \math[\code{a[i]}]{a_{i}}.

As mentioned, there is no math mode in HTML and you have to do with an approximation of the formula. Nevertheless, if you want, you can still have them displayed in an italic font. To do so, place a \htmlmathitalics command in your preamble. It must start on the first character of a line.

5.9 Ignoring input

The percent character % introduces a comment in Hyperlatex. Alternatively, you can use the command \C. Everything after a % or \C up to the end of the line is ignored, as well as any white space on the beginning of the next line.

5.10 Document class and title page

This material appears before the `\topnode` command and is therefore ignored by the Hyperlatex converter. You can use everything you want there.

5.11 Sectioning

The sectioning commands `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph` are recognized by Hyperlatex and used to partition the document in nodes. The `\chapter` command is recognized if the document class is not `article`. You can also use the starred version and the optional argument for the sectioning commands. The star will be ignored, as Hyperlatex does not number sections, and the optional argument will be used for node titles and in menus generated by `htmlmenu`.

You can use `\protect`. It will be ignored in the HTML-version.

5.12 Displayed material

The `quote`, `quotation`, and `verse` environment are all implemented by the Hyperlatex converter — but they are all identical!

The `center` environment is realized using an HTML tag that is currently only understood by the `netscape` browser.

To make lists, you can use the `itemize`, `enumerate`, and `description` environments. You *cannot* specify an optional argument to `\item` in `itemize` or `enumerate`, and you *must* specify one for `description`.

All these environments can be nested.

The `\\` command is recognized, with and without `*`.

There is also a `menu` environment, which looks like an `itemize` environment, but is somewhat denser since the space between items has been reduced. It is only meant for single-line items.

6 Conditional Compilation: Escaping into one mode

In many situations you want to achieve slightly (or maybe even totally) different behavior of the LaTeX code and the HTML-output. Hyperlatex offers several different ways of letting your document depend on the mode.

6.1 LaTeX versus HTML mode

The easiest way to put a command or text in your document that is only included in one of the two output modes it by using a `\texonly` or `\htmlonly` command. They ignore their argument, if in the wrong mode, and otherwise simply expand it:

```
We are now in
       \texonly{\LaTeX}\htmlonly{HTML}-mode.
```

Another possibility is by prefixing a line with `\T` or `\H`. `\T` is equal to `\C` in HTML-mode, and a noop in LaTeX-mode, and for `\H` it is the other way round:

```
We are now in
\T \LaTeX-mode.
\H HTML-mode.
```

The last way of achieving this effect is useful when there are big chunks of text that you want to skip in one mode — a HTML-document might skip a section with a detailed mathematical analysis, a LaTeX-document will not contain a node with lots of hyperlinks to other documents. This can be done using the `iftex` and `ifhtml` environments:

```
We are now in
\begin{iftex}
  \LaTeX-mode.
\end{iftex}
\begin{ifhtml}
  HTML-mode.
\end{ifhtml}
```

6.2 Escaping to 'real' LaTeX

Even within the `iftex` environment the special input mode of Hyperlatex is still effective. Sometimes you will want to be able to use the full power of LaTeX with all its special characters. This can be done in a `tex` environment. It is equivalent to `iftex`, but also turns on the five special characters that make the difference between 'real' TeX and Hyperlatex.

Here is another neat construction that lets you go into 'real' TeX mode for a single line:

```
\T {\tex ... and now we are in real TeX
                                    mode ...  }
```

The `\T` command escapes from Hyperlatex, and the `\tex` command sets TeX's special characters.

6.3 Flags — more on conditional compilation

You can also have sections of your document that are included depending on a flag which you have set or cleared before. To set a flag, use

```
\set{flag}
```

To clear a flag, use

```
\clear{flag}
```

Both commands can be used both in the preamble and in the body of the document. If used in the preamble, they must start at the beginning of the line or else be prefixed with `\H` and whitespace, if the Hyperlatex converted has to see them.

Then you can include parts of your document based on some flag:

```
\begin{ifset}{flag}
  Flag flag is set!
\end{ifset}

\begin{ifclear}{flag}
  Flag flag is not set!
\end{ifset}
```

You can set and clear a flag more than once. It is not an error to test a flag which has not been defined with `\set` or `\clear`. It is considered cleared.

7 Carrying on

In this section we continue to Chapter 3 of the LaTeX-book, dealing with more advanced topics.

7.1 Accents

Hyperlatex recognizes the accent commands

```
\'   \`   \^   \~
```

However, not all possible accents are available in HTML. Hyperlatex will make a HTML-entity for the accents in ISO Latin 1, but will reject all other accent sequences. The command `\c` can be used to put a cedilla on a letter 'c' (either case), but on no other letter. The following is legal

```
Der K{\"o}nig sa\ss{} am wei{\ss}en Strand
von Cura\c{c}ao und nippte an einer
Pi\~{n}a Colada \ldots
```

and produces

> Der König saß am weißen Strand von Curaçao und nippte an einer Piña Colada . . .

Not legal are `Ji{\v r}\'{\i}`, or `Erd\H{o}s`. To get a 'í', you have to type `\'{\i}`, not `\'\i`.

7.2 Defining commands and environments

Hyperlatex understands the simplest type of command definitions, namely commands without parameters, and *only in the preamble*. The `\newcommand` command must start at the beginning of the line, or must be prefixed by a `\H` command and white space. The same holds for new environments. Here are some legal examples:

```
\newcommand{\Html}{\scap{html}}

\T\newcommand{\bad}{$\surd$}
\H\newcommand{\bad}{\htmlimage{%
                        badexample_bitmap.xbm}}

\newenvironment{badexample}{\begin{description}
                \item[\bad]}{\end{description}}

\newcommand{\ipe}{\italic{Ipe}}

\H \newenvironment{smallexample}{%
              \begin{example}}{ \end{example}}
\T \newenvironment{smallexample}{\begingroup
                    \small\begin{example}}{%
                      \end{example}\endgroup}
```

The `\bad` command and the `smallexample` environments are good examples for conditional compilation. The `smallexample` environment is equal to `example` in HTML, but is typeset in a smaller font in the LaTeX document.

It is possible to trick Hyperlatex into defining a new command with an argument, if the HTML-implementation of the new command simply typesets the argument:

```
\T \newcommand{\frameit}[1]{\fbox{#1}}
\H \newcommand{\frameit}{\italic}
```

The new command `\frameit` will typeset its argument in italics in HTML-mode, but will put a frame around it in LaTeX.

There is no `\renewcommand`. You cannot redefine any predefined commands.

7.3 Theorems and such

There is no `\newtheorem` command. But you can define an environment which does approximately the same:

```
% LaTeX definition
\newtheorem{guess}{Conjecture}

% HTML definition
\H \newenvironment{guess}{\begin{quotation}%
                          \bold{Conjecture.}
   \html{I}}{\html{/I}\end{quotation}}
```

(The `\html` command generates plain HTML-tags. The 'I' and '/I' tags used here turn on and off italics mode.)

7.4 Figures and other floating bodies

You can use `figure` and `table` environments and the `\caption` command. They will not float, but will simply appear at the position in the text. No special space is left around them, so put a `center` environment in a figure. The `table` environment is mainly used with the `tabular` environment below.

7.5 Lining it up in columns

There is a weak implementation of the `tabular` environment available in Hyperlatex. First of all, the &-character is not special in Hyperlatex, so instead you have to use the `\S` command to separate columns.

To produce the HTML-version of the table, Hyperlatex removes all the `\S` commands *with any following white space* and the `\\` or `\\*` commands. The result is not formatted any more, and simply included in the HTML-document as a 'preformatted' display. This means that if you format your source file properly, you will get a well-formatted table in the HTML-document — but it is fully your own responsibility.

You can also use the `\hline` command to include a horizontal rule. Here is an example:

```
\begin{table}[htp]
\caption{Keyboard shortcuts for \ipe{}}
\begin{center}
\begin{tabular}{|l|lll|}
\hline
~      \S Left Mouse \S Middle Mouse\S Right Mouse \\
\hline
Plain \S (start drawing)\S move   \S select       \\
Shift \S scale          \S pan    \S select more  \\
Ctrl  \S stretch        \S rotate \S select type  \\
Shift+Ctrl\S ~          \S ~   \S select more type\\
\hline
\end{tabular}
\end{center}
\end{table}
```

Note the use of the ~-character. Without it the `\hline` command would eat up space up to the next `\S` command, and the same holds for the two `\S` commands on the last line. The example is typeset as follows:

Table 1: *Keyboard shortcuts for Ipe*

	Left Mouse	Middle Mouse	Right Mouse
Plain	(start drawing)	move	select
Shift	scale	pan	select more
Ctrl	stretch	rotate	select type
Shift+Ctrl			select more type

7.6 Simulating typed text

The `verbatim` environment and the `\verb` command are implemented. The starred varieties are currently not implemented. (The implementation of the `verbatim` environment is not the standard LaTeX implementation, but the one from the *verbatim.sty* style by Rainer Schöpf). The command `\+`*verb*+ can be used as a shortcut for `\verb+`*verb*+.

Furthermore, there is another, new environment `example`. `example` is also useful for including program listings or code examples. Like `verbatim`, it is typeset in a typewriter font with a fixed character pitch, and obeys spaces and line breaks. But here ends the similarity, since `example` does *not* turn off the five special characters. Using this you can still use font changes within an `example` environment, and you can also place hyperlinks there. Here is an example:

```
To clear a flag, use
\begin{example}
  \+\clear{+\var{flag}\}
\end{example}
```

Note also that an `example` environment is indented automatically, while a `verbatim` environment is not. In the LaTeX document, you can set the amount of indentation by setting `\exampleindent`:

```
\setlength{\exampleindent}{4mm}
```

8 Moving information around

In this section we deal with questions related to cross referencing between parts of your document, and between your document and the outside world. Here lie some of the big differences between a printed paper and a HTML-document. Where you would have an expression such as 'More details can be found in the classical analysis by Harakiri [8]' in the printed paper, the HTML-document would include a hyperlink to Harakiri's work.

8.1 Cross-references

You can use the `\label{`*label*`}` command to attach a *label* to a position in your document. This label can be used to create a hyperlink to this position from any other point in the document. This is done using the `\link` command:

```
\link{anchor}{label}
```

This command typesets anchor, expanding any commands in there, and makes it an active hyperlink to the position marked with *label*:

```
This parameter can be set in the
\link{configuration panel}{sect:con-panel}
                          to influence ...
```

The `\link` command does not do anything exciting in the printed document. It simply typesets the text *anchor*. If you also want a reference in the LaTeX output, you will have to add a reference using `\ref` or `\pageref`. This reference has to be escaped from the Hyperlatex converter. Sometimes you will want to place the reference directly behind the *anchor* text. In that case you can use the optional argument to `\link`:

```
This parameter can be set in the
\link{configuration
  panel}[~(Section~\ref{sect:con-panel})]%
          {sect:con-panel} to influence ...
```

The optional argument is ignored in the HTML-output. In most cases, you will need a `\ref`erence to the label already given in the `\link` command. To save you same typing, the `\link` command therefore defines `\Ref` and `\Pageref` (with capitals) to be `\ref{`*label*`}` and `\pageref{`*label*`}`, where *label* is the label used in the `\link` command. These definitions are already active when the optional argument is expanded. This means that we can rewrite the example above as:

```
This parameter can be set in the
\link{configuration panel}[~(Section~\Ref)]%
            {sect:con-panel} to influence ...
```

Often this format is not useful, because you want to put it differently in the printed manual. Still, as long as the reference comes after the `\link` command, you can use `\Ref` and `\Pageref`.

```
After \link{setting the parameter}{%
      sect:con-panel} it is not difficult
      to show that the dependence of the ....
      .... is obvious\texonly{ (see also
      Section~\Ref)}.
```

Note that when you use LaTeX's `\ref` command, the label does not mark a *position* in the document, but a certain *object*, like a section, equation etc. It sometimes requires some care to make sure that both the hyperlink and the printed reference point to the right place, and sometimes you will have to place two labels. The HTML-label tends to be placed *before* the interesting object — a figure, say — , while the LaTeX-label tends to be put *after* the object (when the `\caption` command has set the counter for the label).

A special case occurs for section headings. Always place labels *after* the heading. In that way, the LaTeX reference will be correct, and the Hyperlatex converter makes sure that the link will actually lead to a point directly before the heading — so you can see the heading when you follow the link.

8.2 Links to external information

You can place a hyperlink to a given URL (Universal Resource Locator) using the `\xlink` command. Like the `\link` command, it takes an optional argument, which is typeset in the printed output only:

```
\xlink{anchor}{URL}
\xlink{anchor}[printed reference]{URL}
```

In the HTML-document, *anchor* will be an active hyperlink to the object *URL*. In the printed document, *anchor* will simply be typeset, followed by the optional argument, if present.

8.3 Links into your document

The Hyperlatex converter automatically partitions your document into HTML-nodes and generates HTML-tags for your `\label`'s. These automatically created names are simply numbers, and are not useful for external references into your document — after all, the exact numbers are going to change whenever you add or delete a section or label, or when you change the `\htmldepth`.

If you want to allow links from the outside world into your new document, you will have to do two things: First, you should give that HTML node a mnemonic name that is not going to change when the document is revised. Furthermore, you may want to place a mnemonic label inside the node.

The `\xname{`*name*`}` command is used to give the mnemonic name *name* to the *next* node created by Hyperlatex. This means that you ought to place it *in front of* a sectioning command. The `\xname` command has no function for the LaTeX-document. No warning is created if no new node is started in between two `\xname` commands.

If you need an HTML label within a node to be referenced from the outside, you can use the `\xlabel{`*label*`}` command. *label* has to be a legal HTML label.

Here is an example: The section 'Changes between Hyperlatex 1.0 and Hyperlatex 1.1' in this document starts as follows.

```
\xname{hyperlatex_changes}
\section{Changes from from Hyperlatex~1.0 to
                            Hyperlatex~1.1}
\label{sec:changes}
```

It can be referenced inside this document with `\link{Changes}{sec:changes}`, and both inside and outside this document with `\xlink{Changes}{hyperlatex_changes.html}`.

The entry about `\xname` and `\xlabel` in that section has been marked using `\xlabel{external_labels}`. You can therefore directly refer to that position from anywhere using

```
\xlink{xlabel is new}{%
   hyperlatex_changes.html#external_labels}
```

8.4 Bibliography and citation

Hyperlatex understands the `thebibliography` environment. Like LaTeX, it creates a section titled 'References'. The `\bibitem` command is equivalent to `\par`, and sets a label with the given *cite key* at the given position. This means that you can use the `\link` command to define a hyperlink to a bibliography entry. The command `\Cite` is defined analogously to `\Ref` and `\Pageref` by `\link`. If you define a bibliography like this

```
\begin{thebibliography}{99}
   \bibitem{latex-book}
   Leslie Lamport, \cit{\LaTeX: A Document
                        Preparation System,}
   Addison-Wesley, 1986.
\end{thebibliography}
```

then you can add a reference to the LaTeX-book as follows:

```
... we take a stroll through the
\link{\LaTeX-book}[~\Cite]{latex-book},
                                explaining ...
```

Hyperlatex also understands the `\bibliographystyle` command (which is ignored) and the `\bibliography` command. It reads the *.bbl* file, inserts its contents at the given position and proceeds as usual. Using this feature, you can include bibliographies created with BibTeX in your HTML-document! It would be possible to design a WWW-server that takes queries into a BibTeX database, runs BibTeX and Hyperlatex to format the output, and sends back a HTML-document.

8.5 Splitting your input

The `\input` command is implemented in Hyperlatex. The subfile is inserted into the main document, and typesetting proceeds as usual. You have to include the argument to `\input` in braces.

8.6 Making an index or glossary

The Hyperlatex converter understands the commands `\index` and `\cindex`, which are synonymous. It collects the entries specified with these commands, and you can include a sorted index using `\htmlprintindex`. This index takes the form of a menu with hyperlinks to the positions where the original `\index` commands where located. You can specify a different sort key for an index intry using the optional argument of `\cindex`:

```
\cindex[index]{\verb+\index+}
```

This entry will be sorted like '`index`', but typeset in the index as '`\verb+\index+`'.

The *hyperlatex.sty* style defines `\cindex` as follows:

- `\cindex{`*entry*`}` is expanded to `\index{`*entry*`}`, and
- `\cindex[`*sortkey*`]{`*entry*`}` ist expanded to `\index{`*sortkey*`@`*entry*`}`.

This realizes the same behavior as in the Hyperlatex converter if you use the index processor `makeindex`. If not, you will have to consult your *Local Guide* and redefine `\cindex` appropriately.

The index in this manual was created using `\cindex` commands in the source file, the index processor `makeindex` and the following code:

```
\H \section*{Index}
\H \htmlprintindex
\T \input{hyperlatex.ind}
```

9 Designing it yourself

In this section we discuss the commands used to make things that only occur in HTML-documents, not in printed papers. Practically all commands discussed here start with `\html`, indicating that the command has no effect whatsoever in LaTeX.

9.1 Making menus

The `\htmlmenu` command generates a menu for the subsections of the current section. It takes a single argument, the depth of the desired menu. If you use `\htmlmenu{2}` in a subsection, say, you will get a menu of all subsubsections and paragraphs of this subsection.

If you use this command in a section, no automatic menu for this section is created.

A typical application of this command is to put a 'master menu' in the top node, containing all sections of all levels of the document. This can be achieved by putting `\htmlmenu{6}` in the text for the top node.

9.2 Rulers and images

The command `\htmlrule` creates a horizontal rule spanning the full screen width at the current position in the HTML-document. It has an optional argument that you can use to add the additional tags `size`, `width`, `align`, and `noshade`. These additional tags are currently only understood by the `netscape` browser. Here is an example.

```
\htmlrule[width=70% align=center]
```

[width=70

The command `\htmlimage{`*URL*`}` makes an inline bitmap with the given *URL*. It takes an optional argument that can be used to specify the additional tags understood by some HTML browsers. One of the letters 't', 'c', 'b', 'l', or 'r' can be specified as a shortcut for the alignments 'top', 'center', 'bottom', 'left', or 'right'. So `\htmlimage[c]{image.xbm}` includes the image in *image.xbm*, vertically centered at the current text position. A more complicated example is

```
\htmlimage[align=left width=50 height=75
                  hspace=3]{image.jpg}
```

(Note that *jpeg* inlined images are currently only understood by the `netscape` browser.)

This is what I use for figures in the Ipe Manual that appear in both the printed document and the HTML-document:

```
\begin{figure}
  \caption{The Ipe window}
  \begin{center}
    \T {\tex\Ipe{window.ipe}}
    \H \htmlimage{window.gif}
  \end{center}
\end{figure}
```

(`\Ipe` is the command to include 'Ipe' figures. Since the figure contains math mode material, it has to be escaped using `\tex`.)

9.3 Adding raw HTML

Hyperlatex provides two commands to access the HTML-tag level.

`\html{`*tag*`}` creates the HTML tag `<tag>`, and `\htmlsym{`*entity*`}` creates the HTML entity description `&entity;`.

The `\htmlsym` command is useful if you need symbols from the ISO Latin 1 alphabet which are not predefined in Hyperlatex. You can, for instance, define the ligature `\AE` as in TEX using

```
\H \newcommand{\AE}{\htmlsym{AElig}}
```

9.4 Turning TEX into bitmaps

There can be many things in a LATEX-file that Hyperlatex doesn't understand: equations, fancy tables, picture environments — the list is endless. Especially equations appear quite often and are pretty hard to represent in HTML. Sometimes the only sensible way to incorporate them into a HTML-document is by turning them into a bitmap. Hyperlatex has an environment `gif` that does exactly this: In the HTML-version, it is turned into a reference to an inline bitmap (just like `\htmlimage`). In the LATEX-version, the `gif` environment is equivalent to a `tex` environment. Note that running the Hyperlatex converter doesn't create the bitmaps yet, you have to do that in an extra step as described below.

The `gif` environment has three optional and one required arguments:

```
\begin{gif}[tags][resolution][                    fon
  TEX material ...
\end{gif}
```

For the LATEX-document, this is equivalent to

```
\begin{tex}
  TEX material ...
\end{tex}
```

For the HTML-version, it is equivalent to

```
\htmlimage[tags]{name.gif}
```

The other two parameters, *resolution* and *font_resolution*, are used when creating the `gif`-file. They default to 100 and 300 dots per inch.

Here is an example:

```
\htmlonly{\par}
\begin{gif}{eqn1}
  \[
  \sum_{i=1}^{n} x_{i} = \int_{0}^{1} f
  \]
\end{gif}
```

produces the following output:

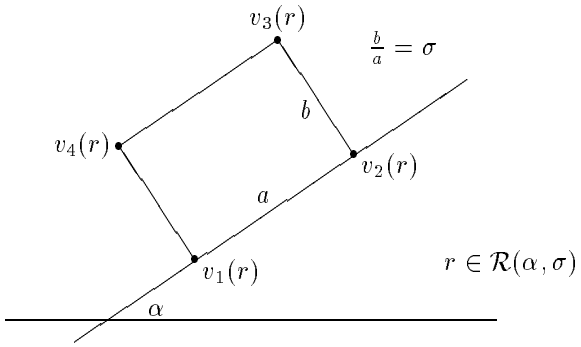
$$\sum_{i=1}^{n} x_i = \int_0^1 f$$

We could as well include a picture environment. The code

```
\begin{center}
 \begin{gif}[b][80]{boxes}
  \setlength{\unitlength}{0.1mm}
  \begin{picture}(700,500)
   \put(40,-30){\line(3,2){520}}
   \put(-50,0){\line(1,0){650}}
   \put(150,5){\makebox(0,0)[b]{$\alpha$}}
   \put(200,80){\circle*{10}}
   \put(210,80){\makebox(0,0)[lt]{$v_{1}(r)$}}
   \put(410,220){\circle*{10}}
   \put(420,220){\makebox(0,0)[lt]{$v_{2}(r)$}}
   \put(300,155){\makebox(0,0)[rb]{$a$}}
   \put(200,80){\line(-2,3){100}}
   \put(100,230){\circle*{10}}
```

```
    \put(100,230){\line(3,2){210}}
    \put(90,230){\makebox(0,0)[r]{$v_{4}(r)$}}
    \put(410,220){\line(-2,3){100}}
    \put(310,370){\circle*{10}}
    \put(355,290){\makebox(0,0)[rt]{$b$}}
    \put(310,390){\makebox(0,0)[b]{$v_{3}(r)$}}
    \put(430,360){\makebox(0,0)[l]{$\frac{b}{a} =
                                        \sigma$}}
    \put(530,75){\makebox(0,0)[l]{$r \in
                        {\cal R}(\alpha, \sigma)$}}
  \end{picture}
 \end{gif}
\end{center}
```

creates the following image.



It remains to describe how you actually generate those bitmaps from your Hyperlatex source. This is done by running LaTeX on the input file, setting a special flag that makes the resulting DVI-file contain an extra page for every `gif` environment. Furthermore, this LaTeX-run produces another file with extension *.makegif*, which contains commands to run `dvips` and `ps2gif` to to extract the interesting pages into Postscript files which are then converted to `gif` format. Obviously you need to have `dvips` and `ps2gif` installed if you want to use this feature. (A shellscript `ps2gif` is supplied with Hyperlatex. This shellscript uses `ghostscript` to convert the Postscript files to `ppm` format, and then runs `ppmtogif` to convert these into `gif`-files.)

Assuming that everything has been installed properly, using this is actually quite easy: To generate the `gif` bitmaps defined in your Hyperlatex source file *source.tex*, you run LaTeX as follows (of course you could make a shell script to save some typing).

```
latex '\def\makegifs{}\input{source.tex}'
```

This will create a DVI-file *source.dvi* and a file *source.makegif*. All `gif` images defined in *source.tex* are then created by calling

```
sh source.makegif
```

9.5 Customizing the navigation panels

Normally, Hyperlatex adds a 'navigation panel' at the beginning of every HTML node. This panel has links to the next and previous node on the same level, as well as to the parent node. The panel for the top node has a link to the first chapter or section.

In the long run, navigation panels should be fully customizable. However, since I'm still pondering how to do that properly, this isn't implemented yet. You can, however, turn the navigation panel off for selected nodes. This is done using the commands `\htmlpanel{0}` and `\htmlpanel{1}`. All nodes started while `\htmlpanel` is set to 0 are created without a navigation panel. Once the standard navigation panel has been suppressed, you can of course design and create your own navigation panel using `\link` commands.

10 Changes since Hyperlatex 1.0

Changes from 1.0 to 1.1

- The only change that introduces a real incompatibility concerns the percent sign %. It has its usual LaTeX-meaning of introducing a comment in Hyperlatex 1.1, but was not special in Hyperlatex 1.0.
- Fixed a bug that made Hyperlatex swallow certain ISO characters embedded in the text.
- Fixed HTML tags generated for labels such that they can be parsed by `lynx`.
- The commands `\+verb+` and `\=` are now shortcuts for `\verb+`*verb*`+` and `\back`.
- It is now possible to place labels that can be accessed from the outside of the document using `\xname` and `\xlabel`.
- The navigation panels can now be suppressed using `\htmlpanel`.
- If you are using LaTeX2$_\varepsilon$, the Hyperlatex input mode is now turned on at `\begin{document}`. For LaTeX2.09 it is still turned on by `\topnode`.
- The environment `gif` can now be used to turn DVI information into a bitmap that is included in the HTML-document.

Changes from 1.1 to 1.2

Hyperlatex 1.2 has a few new options that allow you to better use the extended HTML tags of the `netscape` browser.

- `\htmlrule` now has an optional argument.
- The optional argument for the `\htmlimage` command and the `gif` environment has been extended.
- The `center` environment now uses the *center* HTML tag understood by some browsers.
- The font changing commands have been changed to adhere to LaTeX2$_\varepsilon$. This hasn't been done before because it didn't make sense while font changes in HTML were not properly cumulative. The font size can be changed now as well, using the usual LaTeX commands.

Changes from 1.2 to 1.3

Hyperlatex 1.3 fixes a few bugs.

11 Acknowledgments

Thanks to everybody who reported bugs in Hyperlatex 1.0 or who suggested useful new features. This includes Arne Helme, Bob Kanefsky, Greg Franks, Jim Donnelly, Jon Brinkmann, Nick Galbreath, and Piet van Oostrum.

12 Copyright

Hyperlatex is 'free,' this means that everyone is free to use it and free to redistribute it on certain conditions. Hyperlatex is not in the public domain; it is copyrighted and there are restrictions on its distribution as follows:

13 Glossary

- **node**
  A HTML-document usually consists of several files, here called *nodes*. Other HTML documentation often calls nodes 'documents', and a full document is sometimes referred to as a 'work.'
- **preamble**
  The *preamble* of a LaTeX file is the part between the `\documentstyle` command and the `\begin{document}` command. LaTeX does not allow text in the preamble, you can only put definitions and declarations there. `hyperlatex` looks in the preamble for the commands
  - `\htmldirectory`
  - `\htmlname`
  - `\htmltitle`
  - `\htmldepth`
  - `\htmlmathitalics`
  - `\htmlautomenu`
  - `\htmladdress`
  - `\newcommand`
  - `\newenvironment`
  - `\set`
  - `\clear`

  Note that Hyperlatex will only see these commands if they start at the beginning of a line.
- **top node**
  The *top node* is the entry point of your HTML document. It is stored in a file named *basename.html*, while all other nodes are stored in numbered files (*basename_N.html*). The top node is an ancestor of all other nodes. It is considered to be at level zero, while all other nodes have a level corresponding to the sectioning command, and therefore at least 1.
  The top node often contains a menu of all sections of the document. This can be achieved using the command

  ```
  \htmlmenu{6}
  ```

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, 1986.

[1] at `http://www.cs.ruu.nl/people/otfried/txt/copying.txt`