Ltoh: a customizable LaTeXto HTML converter

Version 97e, 31 Mar 1997

Russell W. Quong

(http://www.best.com/~quong/ltoh)

1 Introduction

Ltoh is a customizable LaTeX to HTML converter. It handles text, tables, and hypertext links. ltoh is a large Perl script, and hence is (almost completely) platform independent. ltoh is customizable in that you can specify how to translate a given LaTeX2ε macro into HTML, including your own personal macros. In fact, you must manually specify how to handle your own macros, otherwise ltoh will give a friendly warning.

See the ltoh web page (`http://www.best.com/~quong/ltoh`) for documentation, the latest release, and how to contact the author (see the bottom of the web page). Naturally, the HTML version of document was generated using ltoh, and in my opinion looks better than the LaTeX2ε dvi/PS output, mostly due to the extra colors. The first public release was version 97e on 3/31/97.

Ltoh has two main restrictions. First, ltoh does *not* handle math equations, which in general are difficult to display in HTML. [1] Second, ltoh requires La/Tex macro parameters to be delimited by braces; in practice, ltoh might be unsuitable for most existing TeXcode.

Surprisingly, I often preview my LaTeX2ε documents via ltoh instead of running `latex`, `dvips`, and `ghostview`.

2 The distribution

Ltoh is distributed as either a zip file (`ltoh.zip`) or a gzipped tar file (`ltoh.tgz`) (about 75K).

Both distributions contain the following files.

ltoh.pl	The perl script that does everything
ltoh.specs	The default specifications.
readme.html	Generated by ltoh
readme.dvi	LaTeX2ε output
readme.ps	Uses Times Roman
readme.txt	Text version (generated from netscape)
README	
rq-ltoh.specs	An example of my specifications
rq209.sty	Allow use of new LaTeX2ε font commands in old LaTeX2ε.

3 System Software Requirements

Ltoh version 97d requires the following system software.

1. A unix operating system. I have tested ltoh only under Linux, although there is nothing Linux specific in the code. A handful of lines are Unix dependent mostly in due to filename conventions. Because ltoh is written in Perl, almost all the code is platform-independent.
2. Perl version 5.00x, preferably 5.002 or higher. Run
   ```
   perl -v
   ```
   to see the version of Perl you have.
3. LaTeX2ε macro parameters (or arguments) must be brace delimited (surrounded by braces), because ltoh relies on the braces.
4. The *new latex* (LaTeX2ε) is strongly recommended over the *old latex* (LaTeX2.09),
   Additionally, the default ltoh specifications is based on standard new latex macros. Finally, to make full use of HTML tables, future versions of ltoh are likely to support multiple rows in the table packages only found in the new latex.

---

[1] Some have resorted to converting the latex equations into Postscript (PS), converting the PS to a bitmapped figure, and the displaying the figure in HTML. This is all too difficult for me.

3.1 If you must use LaTeX2.09 instead of LaTeX2ε

ltoh relies on unique matching braces to delimit arguments to the latex macros. In particular, the font family and size commands in old latex do not use braces to delimit arguments. Thus, ltoh does not (and probably never will) handle old latex 2.09 font specifications. Instead, you must use the LaTeX2ε convention, which delimits the text to be affected by a font change via braces as shown in the following comparison.

```
(Old latex) Normal but switch \bf to {bold \it then italics, back to}
bold \normalfont then normal.

(New latex) Normal but switch \textbf{to bold \textit{then italics, back
to} bold} then normal.
```

Produces:

> Normal but switch **to bold *then italics, back to* bold** then normal.

Using the old latex syntax, ltoh cannot determine when the bold and italic fonts stop being active.

If you have the new latex on your system, use it. If you must use an old latex file, convert it to look like new latex as much as possible.

1. Convert all font change macros to use the new latex syntax. Namely, convert `{\XYZ ...}` and `\XYZ ...\normalfont` to `\textXYZ...`.
2. Use the style file `rq209.sty` (which should be) included with the ltoh distribution, which defines the \text*XYZ* macros for use in the old latex.
   To use this file, put

   `\input{rq209.sty}`

   in your latex files. The file `rq209.sty` additionally defines the font size macros `\fsizeTiny`/.../`\fsize Huge` which take a single brace-delimited argument. For exapmle, use `\fsizesmall{some text}` instead of `{ \small some text }`. (This author wrote `rq209.sty` back in 1994 because the office computer ran the old latex but the home Linux machine ran the new latex.)
   Alternatively, write and use your own definitions of the \text*XYZ* font change macros.

(One final note.) The old latex convention is simply a poor technical chioce. The current philosophy for document specifications (and even programming languages) is that parameters/arguments/blocks are clearly delimited syntactically. The use of matching braces by latex2e conforms to the the SGML syntax, as does HTML which ubiquitously uses matching begin and end tags.

4 Running ltoh

To generate the HTML file `xyz.html` from the latex file `xyz.tex`, assuming ltoh is in your path, run:

```
  prompt> ltoh xyz.tex
or
  prompt> perl fullpath-of-ltoh.pl xyz.tex
```

(I have not tested ltoh on a Win32 machine, yet...) On a Win32 machine, which cannot automatically start Perl to execute the ltoh, you would probably run

```
  prompt> perl ltoh.pl xyz.tex
```

5 Specifications

There are five types of ltoh specifications. Please note the names.

Begin/end pair (`b/e`)	Specifies how to translate a latex `\begin{XYZ}` and matching `\end{XYZ}` command.
Command (`comm`)	Specifies how to translate a latex command that does not take any parameters, such as `\par`, `\item` or `\hrule`.
Simple-macro (`{}`)	Specifies a translation for a latex macro that takes a single brace-delimited argument *arg-1*, where the corresponding HTML consists simply of surrounding the argument with a preamble and postamble. The translation is simple as the argument stays put; ltoh merely puts stuff before and after it. That is, we expect `\simplemacro{...}` ⟶ `HTML-preable ... HTML-postable` For example, use a simple-macro specification to translate the latex macro `\textbf{ ...}` (switch to bold face) into the HTML `<strong> ...</strong>`.
Arg-macro (`{`N`}`)	Specifies a translation for a latex macro that takes N brace-delimited arguments; the corresponding HTML can make arbitrary use of the arguments. For example, my latex macro `\swallow{arg-1}` discards its single (possibly long) argument. In the corresponding HTML, we also "use" the argument, by discarding it.
Assignment (`:=`)	An assignment sets a ltoh variable which is then used later. As of version 97d, only a small number of built-in variables are supported. I hope to support setting and getting user-defined variables in the future.

The first four specications are known as translations specifications.

5.1 Translation specifications

The four types of translation specifications have the same form. Do not use leading whitespace. Here is the general form and an example of each type.

```
:type    :latex-macro-name:HTML-start-code:HTML-end-code:reserved/not-used

:b/e     :\begin{itemize}:<UL>:</>:
:comm    :\hrule:<hr>::
+comm    +\homepage+http://www.best.com/~quong++
:{}      :\textbf:<STRONG>:</>:
:{2} :\rqhttp#1#2:<a href="#2"> #1 </a>::
```

Each specification contains six parts.

1. The first character on the line is the *separator* char, which delimits the remaining parts of the specification. The separator char can be any character. I have used a colon (:) and a plus sign (+) in the above examples. Note that the `\homepage` macro expands to HTML containing a colon, so a colon cannot be the delimiter and I have used a plus. I do not recommend using a space/tab as the delimiter, as multiple spaces/tabs are easy to overlook.
2. The *type*. If the type is arg-macro, the number of parameters is specified in the braces. Arbitrary space may follow the type.
3. The *latex macro name* specifies the latex macro to be translated. A latex macro must start with a backslash (if you have redefined the LaTeX2ε catcodes, sorry). An optional Perl regular expression can following the macro name, to account for parameters for the macro. Usually the latex parameters are just discarded in the resulting HTML.
   As an example of an optional regular expression, the LaTeX2ε horizontal space command `\hspace` takes an optional `*` argument, and then a required horizontal length argument. In the generated HTML, we want to ignore the entire `\hspace` macro, and so I use the following ltoh spec.

   ```
   :comm :\hspace[*]?\{[^\}]+\}:::
   ```
4. The *HTML start code* is the HTML code to insert when the macro is first seen.

begin/end pair	The `\beginXXX` expands to *HTML start code*.
command	The LaTeX2ε macro expands to *HTML start code*.
Simple-macro	The opening brace expands to *HTML start code*; the closing brace expands to *HTML end code*.
arg-macro	The command and all its arguments expands to *HTML start code*.

In an arg-macro specification, using the LaTeX2ε convention, the first argument is represented by #1, the second argument by #2, and so on in the HTML start code. (To actually generate a `\#1`, use braces as in #{1}.) Thus, a macro that swaps the order of its parameters would be written as

```
:{2} :\swap_two:#2#1::
```

As another example, the LaTeX2ε `\makebox` command takes an optional alignment parameter (one of `[l]`, `[c]` or `[r]`) followed by text to be put into the box. I use the following ltoh spec to ignore the alignment parameter and to print the text out unadorned.

```
:{1} :\makebox[^{]*#1:#1::
```

5. The *HTML end code* specifier is the HTML code to insert at (i) the end of a begin-end pair or (ii) the closing brace of a simple-macro. Note that the *HTML end code* specifier does not make sense for a *command* or *arg-macro* specification and is ignored.
   As a convenience, using </> in the *HTML end code* expands to the end tag(s) in reverse order of the corresponding HTML begin code. For example, I want a LaTeX2ε `\section` to show up as a green ¡H2¿ header in HTML, so I specify

   ```
   :{} :\section:<hr><H2><FONT color=green>:</>:
   ```

   which is equivalent to

   ```
   :{} :\section:<hr><H2><FONT color=green>:</FONT></H2></HR>:
   ```

The following table summarizes the effects of the various specifications, and the parts of the spefications used.

Type	macro name	HTML start	HTML end		input	output
comm	\abc	XYZ	not-used		\abc	XZY
b/e	\begin{abc}	XYZ	ijk		\begin{abc}...\end{abc}	XZY ...ijk
{}	\abc	XYZ	ijk		\abc{...}	XYZ...ijk
{2}	\abc	X#2Y#1Z	not-used		\abc{===}{+++}	X+++Y===Z

As a final example, here's how generate links in HTML. I define a latex macro `\rqhttp` and a corresponding ltoh specficiation. Because the tilde is accessible only in math mode, I have had to define a latex macro (`\rqtilde`) for it, too.

```
(latex macro)
\def\rqtilde{\ensuremath{\tilde{\;}}\xspace}
\def\rqhttp#1#2{#1 (\texttt{#2})}
(ltoh  spec)
:comm :\rqtilde:~::
:2 :\rqhttp#1#2:<a href="#2"> #1 </a>::
```

In LaTeX2ε, I use the `\rqhttp` macro as follows.

See the \rqhttp{\ltoh webpage}{http://www.best.com/\rqtilde{}quong/ltoh}.

The resulting dvi output from latex and the HTML from ltoh look like

```
  (Latex) See the ltoh web page (http://www.best.com/~quong/ltoh).

  (HTML) See the <A HREF="http://www.best.com/~quong/ltoh"> ltoh web page </A>
```

Finally, good example of using ltoh specifiers is the default ltoh spec file `ltoh.specs` that comes with this release.

[Aside: Technically, the `simple-macro` specifier is not needed, as its functionality can be duplicated with an `arg-macro`. Namely,

```
:{}     :\macro:HTML-begin:HTML-end::
```

can be duplicated via

```
:{1}     :\macro:HTML-begin#1HTML-end:::.
```

Nonetheless, use of a simple-macro { } specifiction is preferred, because its processing is much simpler. With a simple-macro, ltoh does not have to extract and pass the parameter, and hence it is less likely to break than an arg-macro.]

5.2 Assignment

An assignment specification has two nearly identical forms. The double quotes are optional and can be used to imbed leading spaces into the string-value. The whitespace surrounding string-value is removed.

```
  variable-name := string-value
  variable-name := "string-value"

  title := The readme for ltoh
```

Here are the currently used built-in variables.

variable	Default	Description
`title`	`none`	Title of the resulting HTML file, via the `<TITLE>` tag. <u>**You must define this variable.**</u> Set `title` in the latex file itself. (It drives me nuts when web pages don't have titles)
`url`	`none`	URL of the home page of the author.
`author`	`none`	Author of the document.
`email`	`none`	Email address to which comments should be sent
`htmlfile_spec`	`$BASE.html`	Name of HTML file generated. The ltoh variable `$BASE` is the latex file name stripped of the directory and suffix components.

The `url`, `author`, and `email` variables are used to generate an address block at the bottom of the HTML page. (See the bottom of this document if you are reading it on the web).

6 Tables

ltoh handles the LaTeX2ε `tabular` and `tabularx` environments. Column alignments are read and passed onto the corresponding HTML. The known column alignments must be one of " l c r p X". If you define your own column alignment, it will not be understood.

ltoh handles the LaTeX2ε `multicolumn` macro reasonably well. The column alignment is read and passed onto the corresponding HTML.

I plan to support the `\multirow` macro soon.

ltoh ignores extraneous LaTeX2ε specs in the column alignment specifications, such as ,
but there is a small chance a complicated multiple column alignment spec will break this code.

The generated HTML table has a border if one or more dividing lines in the LaTeX2ε table column alignment (namely if a | specifier appears in the latex, the HTML table has a border). Nested tables are not supported, and I currently do not plan to support them, as this feature seems hard.

7 Where are specifications found?

As of version 97d, ltoh reads specifications from (i) various specification-files and (ii) from the LaTeX2ε file itself as it is being processed, as follows.

1. Reads the specification file `ltoh.specs` from the `install-dir`, which is the directory where you have placed both `ltoh.pl` and `ltoh.specs`. The `install-dir` can be anywhere.
   (In version 97d, you should do one of the following when running ltoh.
   (a) Run ltoh via:
   `prompt> perl install-dir/ltoh.pl file.tex`.
   (b) Set up an alias as follows:
   (csh) `alias ltoh perl install-dir/ltoh`
   or
   (bash) `alias ltoh=perl install-dir/ltoh`
   (c) Put `install-dir` in your path, and run:
   `prompt> ltoh.pl file.tex`

   (d) Set up a single symbolic link, say called `ltoh`, in your path which points directly to `install-dir/ltoh.pl`. You can then run
   `prompt> ltoh file.tex`

   This mess is relative symbolic links. Yes. Given an arbitrary invocation of ltoh involving symbolic links, I cannot currently determine where the `ltoh.pl` script actually resides (the `install-dir`). Once I implement this code, the setup won't be complicated.
2. Reads the specification file `.ltoh.specs` in your home directory.
3. Reads the specification file `.ltoh.specs` in the current directory, where the command is being run.
4. If any of these specification files exist, ltoh proceeds to the next step reading the LaTeX2ε input file
   However, if none of the preceding spec files were, ltoh tries to read
   `/usr/local/bin/ltoh.specs` and that fails, tries `/usr/bin/ltoh.specs`. If both of these still fail, ltoh **quits**.
5. Reads specifications from the latex file being processed, which appear as latex comments of the form

   `%-ltoh-` *`ltoh-specification`*

   .

   ltoh strips the leading `%-ltoh-` and processes the remainder of the line.
   If nothing else, set the `title` variable this way. For example, here's how this LaTeX2ε file starts.

```
\documentclass[]{article}
... various latex commands like \usepackage
%-ltoh-   title := Ltoh, a customizable LaTeX to HTML converter
%-ltoh-   :comm:\ltoh:<font color=green><tt>ltoh</tt></font>::
...
\begin{document}
... the body of the document
```

8 Limitations, bugs, missing features

8.1 Limitations

1. Specifications are applied only once to a latex macro. Namely if a macro *macroAA* generates a macro *macroBB* in latex, and *macroBB* is translated to *htmlBB*, then you must specify to ltoh that *macroAA* is translated directly to *htmlBB* in your specification for *macroAA*.

8.2 Bugs

It is not difficult to break ltoh, though there are often easy fixes by restructuring your LaTeX2ε.

8.3 Missing features

1. There are no command line flags. The following would be particularly useful.

-o *ofile*	generate HTML into file *ofile*
-I *specfile*	read specifications from *specfile*
-w N	set the warning level to N (for debugging)

2. There no means to manipulate counters, which would be useful for sectioning commands.
3. There should be away to set and get a value as part of a specification.
4. If the input uses LaTeX2ε sectioning macros, optionally generate a table of contents with links to anchors.
5. Allow for an abitrary prologue and epilogue in the HTML file.
6. In a table, we assume that the column alignment specs are on the same line as the owning `\begin{tabular}` or the owning `\multicolumn`. This assumption is very reasonable and circumventing this restriction is difficult.

9 Internal details

Ltoh first reads the entire LaTeX2ε file into memory, makes several passes over the data, and finally spits out the HTML file[2] The passes are roughly as follows.

1. Read spec files.
2. Handle comments and the verbatim environment.
3. Escape/protect characters.
4. Handle tables.
5. Handle begin/end pairs.
6. Handle LaTeX2ε commands without parameters.
7. Mark braces, linking braces to macros.
8. Handle simple macros.
9. Handle arg macros.
10. Generate HTML.

10 History

June 1996	Version 96a. Preliminary fully hard-coded (not customizable) version. Purely regular expression based. Unable to handle nested braces. Ugly, but it worked. Sort of.
July 1-15 1996	Version 96b. First working version. Able to handle commands with multiple arguments and nested arguments. Took me a lot longer than I had expected to get this working.
Jan 27-29 1997	Version 97a. Stops processing at `\end{document}`. Convert double back slashes `\\`to ` `, which should have done a long time ago. Fixed bug involving macros with only one parameter.
Feb 1997	Version 97b. Added HTML `<p>` tags whenever two or more consecutive blank lines are seen.
Mar 11-15 1997	Version 97c. Much improved handling of special characters such as {, }, <, > and @. In particular, bare braces which mean nothing in latex are stripped from the HTML. Improved paragraph detection handling. (OK, OK, "Improved ..." really means "fixed bugs in ..."). No longer generates HTML comments for latex comments, by default. Version 97c was meant to be first public release, but the tables this `readme.tex` document broke ltoh badly.
Mar 19-20 1997	Version 97d. Complete rewrite of the table handling code. Latex column alignment specifications are understood and passed onto to the HTML. Multiple columns specified via either `\multicolumn` or `\mc` (which is my personal abbreviation macro) are handled properly. We try to ignore extraneous LaTeX2ε specs in the column alignment specifications, such as `@` . , but there is there is a small chance multiple columns In particular, ltoh now handles this file (`readme.tex`) properly.
Mar 25-31 1997	Version 97e. Official release. Clean up source a bit for release. Minor improvements on tables (allow end of a row to be on a separate line), paragraphs, specification files and handling special characters (allow for multiple chars on one line).

11 License for use

You may use ltoh freely, under the following conditions, which are covered under a BSD-style license.

1. You must keep the ltoh notice at the end of all generated HTML. This notice indicates how the document was generated (namely, with ltoh) and has a link to the ltoh web page. You may reduce the size of the notice if you wish, but it must remain in your document and visible.
2. Ltoh comes with neither a warranty nor a gaurantee about its correctness, performance, or suitability for any task.
3. If you modify or redistribute ltoh, you must keep the current disclaimer and license with the ltoh source which must be visible upon startup, unless you
4. I, Russell Quong, retain the copyright for ltoh.

---

[2] Thus, it is conceivable that ltoh might run out of memory if processing a huge LaTeX2ε file on a machine with limited memory.

11.1 Official software license

Here's the official license as of 31 Mar 97.

```
# Copyright (c) 1996, 1997 Russell W Quong.
#
# In the following, the "author" refers to "Russell Quong."
#
# Permission to use, copy, modify, distribute, and sell this software and
# its documentation for any purpose is hereby granted without fee, provided
# that the following conditions are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#      This product includes software developed by Russell Quong.
# 3. All HTML generated by ltoh must retain a visible notice that it
#    was generated by ltoh and contain a link to the ltoh web page
#
# Any or all of these provisions can be waived if you have specific,
# prior permission from the author.
#
# THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND,
# EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY
# WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
#
# IN NO EVENT SHALL RUSSELL QUONG BE LIABLE FOR ANY SPECIAL,
# INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY
# DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
# WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY
# THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
# PERFORMANCE OF THIS SOFTWARE.
```

12 Motivation

(The motivation section belongs right after the introduction, but most people probably just want to get on with using ltoh. So this section has been relegated here. Ah well...)

Although other LaTeX2ε to HTML converters exist, I wrote my own because I wanted to generate HTML customized to my own liking. In particular, I use my own custom LaTeX2ε definitions (who doesn't?) and wanted to generate HTML appropriately. Additionally, when using other converters, I was unable get them to run properly, or I did not like the way the generated HTML looked.

Fundamentally, ltoh is a specialized macro processor that reads macro *specifications* and generates HTML accordingly. A specification indicates how to convert a specific LaTeX2ε definition into HTML.

My orginal goals in writing ltoh were

1. To become more proficient in Perl.
2. To write a converter that allowed customized HTML to be generated.
3. To handle arbitrary LaTeX2ε macros, including macros that take multiple arguments, and arguments that contained nested macros.
4. To generate HTML that resembled the original LaTeX2ε, as if someone had done the translation by hand. In particular, I wanted to avoid generating very long lines of HTML source, which are difficult to read when "viewing the source" in a browser.

13 Acknowledgements

Thanks to VA Research (`http://www.varesearch.com`) for letting the author work on Itoh.