

# LaTeX::ToUnicode documentation

Boris Veytsman\*

June 23, 2026

## Contents

---

\*borisv@lk.net

**NAME**

ltx2unitxt – convert LaTeX source fragment to plain (Unicode) text or simple html

**SYNOPSIS**

**ltx2unitxt** [*-c CONFIG*] [*-o OUTPUT*] [*--html*] [...] [*INFILE*]...

**DESCRIPTION**

Convert the LaTeX source in INFILE (or standard input) to plain text using Unicode code points for accents and other special characters; or, optionally, output HTML with simple translations for font changes and url commands.

Common accent sequences, special characters, and simple markup commands are translated, but there is no attempt at completeness. Math, tables, figures, sectioning, etc., are not handled in any way, and mostly left in their TeX form in the output. The translations assume standard LaTeX meanings for characters and control sequences; macros in the input are not considered.

The input can be a fragment of text, not a full document, as the purpose of this script was to handle bibliography entries and abstracts (for the ltx2crossrefxml script that is part of the crossrefware package). Patches to extend this script are welcome. It uses the LaTeX::ToUnicode Perl library for the conversion; see its documentation for details.

Conversion is currently done line by line, so TeX constructs that cross multiple lines are not handled properly. If it turns out to be useful, conversion could be done by paragraph instead.

The config file is read as a Perl source file. It can define a function ‘LaTeX\_ToUnicode\_convert\_hook()’ which will be called early; the value it returns (which must be a string) will then be subject to the standard conversion.

For an example of using this script and associated code, see the TUGboat processing at <https://github.com/TeXUsersGroup/tugboat/tree/trunk/capsules/crossref>.

**OPTIONS**

- c, --config=FILE**  
read (Perl) config FILE for a hook, as explained above
- e, --entities**  
output entities &#xNNNN; instead of literal characters
- g, --german**  
handle some features of the german package
- h, --html**  
output simplistic HTML instead of plain text
- o, --output=FILE**  
output to FILE instead of stdout
- v, --verbose**  
be verbose
- V, --version**  
output version information and exit
- , --help**  
display this help and exit

Options can be abbreviated unambiguously, and start with either `-` or `--`.

Dev sources, bug tracker: <https://github.com/borisveytsman/bibtexperllibs>

Releases: <https://ctan.org/pkg/bibtexperllibs>

ltx2unitxt (bibtexperllibs) 0.51 Copyright 2023 Karl Berry. This is free software: you can redistribute it and/or modify it under the same terms as Perl itself.

## 2 LaTeX::ToUnicode

Convert LaTeX commands to Unicode

### VERSION

version 1.93

### SYNOPSIS

```
use LaTeX::ToUnicode qw( convert debuglevel $endcw );

# simple examples:
convert( '{"a}' ) eq 'ä';      # true
convert( '{"a}', entities=>1 ) eq '&#00EF;'; # true
convert( '"a', german=>1 ) eq 'ä';      # true, 'german' package syntax
convert( '"a', ) eq '"a';      # false, not enabled by default

# more generally:
my $latexstr;
my $unistr = convert($latexstr); # get literal (binary) Unicode characters

my $entstr = convert($latexstr, entities=>1);      # get &#xUUUU;

my $htmstr = convert($latexstr, entities=>1, html=>1); # also html markup

my $unistr = convert($latexstr, hook=>\&my_hook); # user-defined hook

# if nonzero, dumps various info; perhaps other levels in the future.
LaTeX::ToUnicode::debuglevel($verbose);

# regexp for terminating TeX control words, e.g., in hooks.
my $endcw = $LaTeX::ToUnicode::endcw;
$string =~ s/\newline$endcw/ /g; # translate \newline to space
```

### DESCRIPTION

This module provides a method to convert LaTeX markups for accents etc. into their Unicode equivalents. It translates some commands for special characters or accents into their Unicode (or HTML) equivalents and removes formatting commands. It is not at all bulletproof or complete.

This module is intended to convert fragments of LaTeX source, such as bibliography entries and abstracts, into plain text (or, optionally, simplistic HTML). It is not a document conversion system. Math, tables, figures, sectioning, etc., are not handled in any way, and mostly left in their TeX form in the output. The translations assume standard LaTeX meanings for characters and control sequences; macros in the input are not considered.

The aim for all the output is utter simplicity and minimalism, not faithful translation. For example, although Unicode has a code point for a thin space, the LaTeX `\thinspace` (etc.) command is translated to the empty string; such spacing refinements desirable in the TeX output are, in our experience, generally not desired in the HTML output from this tool.

As another example, TeX % comments are not removed, even on lines by themselves, because they may be inside verbatim blocks, and we don't attempt to keep any such context. In practice, TeX comments are rare in the text fragments intended to be handled, so removing them in advance has not been a great burden.

As another example, LaTeX ties, ~ characters, are replaced with normal spaces (exception: unless they follow a / character or at the beginning of a line, when they're assumed to be part of a url or a pathname), rather than a no-break space character, because in our experience most ties intended for the TeX output would just cause trouble in plain text or HTML.

Regarding normal whitespace: all leading and trailing horizontal whitespace (that is, SPC and TAB) is removed. All internal horizontal whitespace sequences are collapsed to a single space.

After the conversions, all brace characters ({} ) are simply removed from the returned string. This turns out to be a significant convenience in practice, since many LaTeX commands which take arguments don't need to do anything for our purposes except output the argument.

On the other hand, backslashes are not removed. This is so the caller can check for \\ and thus discover untranslated commands. Of course there are many other constructs that might not be translated, or translated wrongly. There is no escaping the need to carefully look at the output.

Suggestions and bug reports are welcome for practical needs; we know full well that there are hundreds of commands not handled that could be. Virtually all the behavior mentioned here would be easily made customizable, if there is a need to do so.

## FUNCTIONS

### **convert( \$latex\_string, %options )**

Convert the text in `$latex_string` into a plain(er) Unicode string. Escape sequences for accented and special characters (e.g., `\i`, `\"a`, ...) are converted. A few basic formatting commands (e.g., `{\it ...}`) are removed. See the *LaTeX::ToUnicode::Tables* submodule for the full conversion tables.

These keys are recognized in `%options`:

#### **entities**

Output `&#xUUUU`; entities (valid in XML); in this case, also convert the `<`, `>`, `&` metacharacters to entities. Recognized non-ASCII Unicode characters in the original input are also converted to entities, not only the translations from TeX commands.

The default is to output literal (binary) Unicode characters, and not change any metacharacters.

#### **german**

If this option is set, the commands introduced by the package 'german' (e.g. `"a eq ä`, note the missing backslash) are also handled.

#### **html**

If this option is set, the output is simplistic html rather than plain text. This affects only a few things: 1) the output of urls from `\url` and `\href`; 2) the output of markup commands like `\textbf` (but nested markup commands don't work); 3) two other random commands, `\enquote` and `\path`, because they are needed.

## hook

The value must be a function that takes two arguments and returns a string. The first argument is the incoming string (may be multiple lines), and the second argument is a hash reference of options, exactly what was passed to this `convert` function. Thus the hook can detect whether html is needed.

The hook is called (almost) right away, before any of the other conversions have taken place. That way the hook can make use of the predefined conversions instead of repeating them. The only changes made to the input string before the hook is called are trivial: leading and trailing whitespace (space and tab) on each line are removed, and, for HTML output, incoming ampersand, less-than, and greater-than characters are replaced with their entities.

Any substitutions that result in Unicode code points must use `\\x{nnnn}` on the right hand side: that's two backslashes and a four-digit hex number.

As an example, here is a skeleton of the hook function for TUGboat:

```
sub LaTeX_ToUnicode_convert_hook {
    my ($string,$options) = @_ ;

    my $endcw = $LaTeX::ToUnicode::endcw;
    die "no endcw regexp in LaTeX::ToUnicode??" if ! $endcw;

    ...
    $string =~ s/\\newline$endcw/ /g;

    # TUB's \acro{} takes an argument, but we do nothing with it.
    # The braces will be removed by convert().
    $string =~ s/\\acro$endcw//g;
    ...
    $string =~ s/\\CTAN$endcw/CTAN/g;
    $string =~ s/\\Dash$endcw/\\x{2014}/g; # em dash; replacement is string
    ...

    # ignore \begin{abstract} and \end{abstract} commands.
    $string =~ s,\\(begin|end)$endcw\\{abstract}\\s*,g;

    # Output for our url abbreviations, and other commands, depends on
    # whether we're generating plain text or HTML.
    if ($options->{html}) {
        # HTML.
        # \tbsurl{URLBASE} -> <a href="https://URLBASE">URLBASE</a>
        $string =~ s,\\tbsurl$endcw\\{([~]*}\\}
            ,<a href="https://$1">$1</a>,gx;
        ...
        # varepsilon, and no line break at hyphen.
        $string =~ s,\\eTeX$endcw,\\x{03B5}<noBr>-</noBr>TeX,g;
    }
```

```

    } else {
        # for plain text, we can just prepend the protocol://.
        $string =~ s,\\tbsurl$endcw,https://,g;
        ...
        $string =~ s,\\eTeX$endcw,\\x{03B5}-TeX,g;
    }
    ...
    return $string;
}

```

As shown here for `\eTeX` (an abbreviation macro defined in the TUGboat style files), if markup is desired in the output, the substitutions must be different for HTML and plain text. Otherwise, the desired HTML markup is transliterated as if it were plain text. Or else the translations must be extended so that TeX markup can be used on the rhs to be replaced with the desired HTML (`&lt;nobr&gt;` in this case).

For the full definition (and plenty of additional information), see the file `ltx2crossrefxml-tugboat.cfg` in the TUGboat source repository at <https://github.com/TeXUsersGroup/tugboat/tree/trunk/capsules/crossref>.

The hook function is specified in the `convert()` call like this:

```
LaTeX::ToUnicode::convert(..., { hook => \&LaTeX_ToUnicode_convert_hook })
```

## debuglevel( \$level )

Output debugging information if `$level` is nonzero.

## \$endcw

A predefined regexp for terminating TeX control words (not control symbols!). Can be used in, for example, hook functions:

```

my $endcw = $LaTeX::ToUnicode::endcw;
$string =~ s/\\newline$endcw/ /g; # translate \newline to space

```

It's defined as follows:

```
our $endcw = qr/(?<=[a-zA-Z])(?=[^a-zA-Z]|$)\s*/;
```

That is, look behind for an alphabetic character, then look ahead for a non-alphabetic character (or end of line), then consume whitespace. Fingers crossed.

## AUTHOR

Gerhard Gossen <gerhard.gossen@googlemail.com>, Boris Veytsman <boris@varphi.com>, Karl Berry <karl@freefriends.org>

<https://github.com/borisveytsman/bibtexperllibs>

## **COPYRIGHT AND LICENSE**

Copyright 2010-2026 Gerhard Gossen, Boris Veytsman, Karl Berry

This is free software; you can redistribute it and/or modify it under the same terms as the Perl5 programming language system itself.



### 3 LaTeX::ToUnicode::Tables

Character tables for LaTeX::ToUnicode

#### VERSION

version 0.54

#### CONSTANTS

##### @LIGATURES

Standard TeX character sequences (not `\commands`) which need to be replaced: `---` with U+2014 (em dash), etc. Includes: em dash, en dash, inverted exclamation, inverted question, left double quote, right double quote, left single quote, right single quote. They are replaced in that order.

##### %MARKUPS

Hash where keys are the names of formatting commands like `\tt`, without the backslash, namely: `bf cal em it rm sc sf sl small tt`. Values are the obvious HTML equivalent where one exists, given as the tag name without the angle brackets: `b em i tt`. Otherwise the value is the empty string.

##### %ARGUMENT\_COMMANDS

Hash where keys are the names of TeX commands taking arguments that we handle, without the backslash, such as `enquote`. Each value is a reference to a list of two strings, the first being the text to insert before the argument, the second being the text to insert after. For example, for `enquote` the value is `["'", "'"]`. The inserted text is subject to further replacements.

Three such commands are currently handled: `\emph`, `\enquote`, and `\path` (although `\path` does not require its argument to be in braces, that is the only form we recognize). We also recognize and ignore a few other commands taking arguments, such as `\adjust` and `\textgreek`. (The idea of ignoring `\textgreek` is that if the argument is UTF-8 Greek characters, most likely they will be rendered fine by the browser or whatever. We can't solve the hyphenation problem.)

##### %CONTROL\_SYMBOLS

A hash where the keys are non-alphabetic `\commands` (without the backslash), other than accents and special cases. These don't take arguments. Although some of these have Unicode equivalents, such as the `\`, thin space, it seems better to keep the output as simple as possible; small spacing tweaks in TeX aren't usually desirable in plain text or HTML.

The values are single-quoted strings `'\x{...}'`, not double-quoted literal characters `<"\x{...}">`, to ease future parsing of the TeX/text/HTML.

This hash is necessary because TeX's parsing rules for control symbols are different from control words: no space or other token is needed to terminate control symbols.

## **%CONTROL\_WORDS**

Keys are names of argument-less commands, such as `\LaTeX` (without the backslash). Values are the replacements, often the empty string.

## **%SYMBOLS**

Keys are the commands for extended characters, such as `\AA` (without the backslash.)

## **%ACCENT\_SYMBOLS**

Two-level hash of accented characters like `\'{a}`. The keys of this hash are the accent symbols (without the backslash), such as `'` and `'`. The corresponding values are hash references where the keys are the base letters and the values are single-quoted `'\x{...}'` strings.

## **%ACCENT\_LETTERS**

Same as `%ACCENT_SYMBOLS`, except the keys are accents that are alphabetic, such as `\c` (without the backslash as always).

As with control sequences, it's necessary to distinguish symbols and alphabetic commands because of the different parsing rules.

## **%GERMAN**

Character sequences (not necessarily commands) as defined by the package `'german'`/`'ngerman'`, e.g. `"a` (a with umlaut), `"s` (german sharp s) or `"'"` (german left quote). Note the missing backslash.

The keys of this hash are the literal character sequences.

## **AUTHOR**

Gerhard Gossen <gerhard.gossen@googlemail.com>, Boris Veytsman <boris@varphi.com>, Karl Berry <karl@freefriends.org>

<https://github.com/borisveytsman/bibtexperllibs>

## **COPYRIGHT AND LICENSE**

Copyright 2010-2026 Gerhard Gossen, Boris Veytsman, Karl Berry

This is free software; you can redistribute it and/or modify it under the same terms as the Perl5 programming language system itself.