ChkTeX v1.7.10

Jens T. Berger Thielemann

February 26, 2026

1 Introduction

This program has been written in frustration because some constructs in LaTeX are sometimes non-intuitive, and easy to forget. It is *not* a replacement for the built-in checker in LaTeX; however it catches some typographic errors LaTeX oversees. In other words, it is Lint for LaTeX. Filters are also provided for checking the LaTeX parts of CWEB documents.

It is written in ANSI C and compiles silently under GCC using "`-Wall -ansi -pedantic`" and almost all warning flags. This means that you can compile & use the program on your favorite machine. Full source included.

The program also supports output formats suitable for further processing by editors or other programs, making errors easy to cycle through. For example, recent versions of AUCTeX (the Emacs mode) interface beautifully with ChkTeX.

The program itself does not have any machine requirements; However compiling for other platforms has not been done for a long time now so the code has been removed. If interest rises it can be resurrected.

2 Features

ChkTeX begins to get quite a few bells & whistles now. However, you should be aware of that in most cases, all this is transparent to the user. As you will see, ChkTeX offers the ability to adapt to many environments and configurations.

- Supports over 40 warnings. Warnings include:
  - Commands terminated with space. Ignores "`\tt`", etc.
  - Space in front of references instead of "`~`".
  - Forgetting to group parenthesis characters when sub-/superscripting.
  - Italic correction ("`\/`") mistakes (double, missing, unnecessary).
  - Parenthesis and environment matching.
  - Ellipsis detection; also checks whether to use "`\dots`", "`\cdots`" or "`\ldots`".
  - Enforcement of normal space after abbreviation. Detects most abbreviations automagically.
  - Enforcement of end-of-sentence space when the last sentence ended with

capital letter.
- Math-mode on/off detection.
- Quote checking, both wrong types (“`"`”) and wrong direction.
- Recommends splitting three quotes in a row.
- Searching for user patterns.
- Displays comments.
- Space in front of “`\label`” and similar commands.
- Use of “`x`” instead of “`$\times$`” between numbers.
- Multiple spaces in input which will be rendered as one space (or multiple spaces, where that is undesirable).
- Warns about text which may be ignored.
- Mathematical operators typeset as variables.
- No space in front of/after parenthesis.
- Demands a consistent quote style.
- Punctuation inside inner math mode/outside display math mode.
- Use of TEX primitives where LATEX equivalents are available.
- Space in front of footnotes.
- Bogus characters following commands.
- Ability to suppress warnings on a single line.

- Fully customizable. Intelligent resource format makes it possible to make ChkTEX respect your LATEX setup. Even command-line options may be specified globally in the “`chktexrc`” file.
- Supports “`\input`” command; both TEX and LATEX version. Actually includes the files. “`TEXINPUTS`”-equivalent search path.
- Intelligent warning/error handling. The user may promote/mute warnings to suit his preferences. You may also mute warnings in the header of a file; thus killing much unwanted garbage.
- Scripts included for checking CWEB files written in LATEX. (Requires perl v5).
- Supports both LATEX 2.09 and LATEX 2_ε.
- Flexible output handling. Has some predefined formats and lets the user specify his own format. Uses a “`printf()`” similar syntax. “`lacheck`” compatible mode included for interfacing with systems which only support lacheck.
- Written in ANSI C. “`configure`” script included for easy setup and installation on UNIX systems.

Still, it is important to realize that the output from ChkTEX is only intended as a *guide* to fixing faults. However, it is by no means always correct. This means that correct LATEX code may produce errors in ChkTEX, and vice versa: Incorrect LATEX code may pass silently through.

3 Legal stuff

4 Availability

This program is on CTAN; thus it can be found at any mirrors of those. It is also part of TeX Live 2011.

5 Installation

A few words on installation on various platforms:

UNIX: Type "`configure`", "`make`" and finally "`make install`". To make sure everything proceeded correctly, type "`make check`". If you don't have superuser privileges and thus access to the default system areas, you should type "`configure --help`" to help you set up correct paths.

If you haven't installed any software like this before, that is distributed in source form, here are some guidelines to help you install it locally at your account. Please note that a mail to the system administrator may be less work for you.

We assume that you have put the archive ("`chktex.tar.gz`") in a subdir of yours, with path "`~/tmp`". We further assume that your shell is "`csh`" or "`tcsh`". Do the following:

1. First of all, unpack the archive contents.

```
> cd ~/tmp
> gunzip chktex.tar.gz
> tar xf chktex.tar
```

2. Now, we can configure the program. There are some configuration options you should know about:

   "`--enable-pcre`": Allows using PCRE (perl compatible regular expressions) for use defined warnings. The default is to use PCRE if it is installed on your system as determined by "`pcre-config`". You can use "`--disable-pcre`" if you plan to distribute this for systems in which you cannot ensure PCRE will be installed.
   User defined regular expressions are defined using `UserWarnRegex` in the "`chktexrc`" file. See warning 44 for more information.

   "`--enable-lacheck-replace`" This enables a quick hack for using ChkTEX instead of lacheck. This is done by installing a stub script which "overrides" the original lacheck executable. In this way, tools which support lacheck can be easily made to support chktex instead.

   "`--enable-debug-info`" ChkTEX has an ability to spit out various diagnostic messages using the "`-d`" flag. This behaviour is on by default. By adding the flag "`--disable-debug-info`" to the commandline, this will not be compiled in.
   This may be useful if you're running short of disk space (the time savings are neglible).

   If you are installing the program on your local account, use the following command:

   ```
   > configure --prefix ~/
   ```

   Add eventual extra flags as specified above. This command will generate a significant amount of output, this can usually be ignored.

3. Finally, we can just build the program and install it.

   ```
   > make
   > make install
   ```

4. Finished! The program is now installed and ready to use. You may now tell other people to put your bindir in their path in order to benefit from your work. All that remains is to make the shell aware of your installation.

   ```
   > rehash
   ```

   To make the remaining parts of your system aware of this, you'll have to log out and re-log in, I'm afraid. However, you should delay this until you've completed this installation procedure.

5. If you wish to make sure that everything is OK (you ought to), you may now ask ChkTEX to do a self-test:

   ```
   > make check
   ```

Other platforms: First of all, you have to copy the "`config.h.in`" file to a file named "`config.h`". Then, edit it to reflect your system. Do the same with "`OpSys.h`" (this file has been reduced significantly). If you wish, you

may define "DATADIR" to the path you want the global resource file to be put.

Now, I would suggest that you take a peak at the "OpSys.c" file, and edit it appropiately, for more comfort. This should not be necessary, though, at least not the first time.

Finally, you may now compile and link all .c files. Do not forget to define "HAVE_CONFIG_H" to 1 (on the command-line, for instance). If the "config.h" you wish to use has another name, define "CONFIG_H_NAME" to that (in that case, don't define "HAVE_CONFIG_H").

Put the directory path of the "chktexrc" file in a environment variable named "CHKTEXRC". The files "deweb.in" and "chkweb" should be moved to a directory in your path. These files may need further setup, as they haven't got the location of perl initialized.

If your compiler/the compiled program complains (or crashes!), you may try the hints listed below. Please note that it only makes sense to try these hints if your compiler fails to produce a working program.

1. Increase the preprocessor buffers and line buffers. The ChkTEX sources define macros sized 3–4k (expanding to about the same), and passes arguments sized about 1k.
2. Use the magic switch which lets us use large "switch(...){...}" statements; some of these statements have about 120 "case" entries.
3. The sources require that at least the first 12 of each identifier is significant.

Note: You *must* install the new "chktexrc" file; ChkTEX will fail to function otherwise!

After doing this, you may enhance ChkTEX's behaviour by reading/editing the "chktexrc" file.

6 Usage

6.1 ChkTEX

6.1.1 Synopsis

A UNIX-compliant template format follows:

```
chktex  [-hiqrW] [-v[0-...]] [-l <rcfile>]
        [-[wemn] <[1-42]|all>] [-d[0-...]]
        [-p <pseudoname>] [-o <outputfile>] [-[btxgI][0|1]]
        file1 file2 ...
```

6.1.2 Comamndline Options

These are the options ChkTEX currently accepts. Please note that single-lettered options requiring a numerical or no argument may be concatenated. E.g. saying "-v0qb0w2" is the same as saying "-v0 -q -b0 -w2", except for being less to type.

Enough general talk; here's a rather detailed description of all options:

Misc. options: General options which aren't related to some specific subpart of ChkTeX.

`-h [--help]` Gives you a command summary.

`-i [--license]` Shows distribution information.

`-l [--localrc]` Reads a resource-file formatted as the global resource-file "`chktexrc`", in addition to the global resource-file. This option needs the name of the resource-file as a parameter. See also `-g`.

`-r [--reset]` This will reset all settings to their defaults. This may be useful if you use the `CMDLINE` directive in your "`chktexrc`" file, and wish to do something unusual.

`-d [--debug]` Needs a numeric argument; a bitmask telling what to output. The values below may be added in order to output multiple debugging info.

Value	Dumps...
1	All warnings available and their current status.
2	Statistics for all lists in the resource file.
4	The contents of all lists in the resource file.
8	Misc. other status information.
16	Run-time info (note that this isn't widely used).

The info is produced after all switches and resource files have been processed.

It is possible to install versions of ChkTeX that ignore this flag; this means that it is not certain that this flag works.

`-W [--version]` Displays version information, and exits.

Muting warning messages: Controls whether and in what form error messages will appear. Usually they accept a specific warning number (e.g. "`-w2`"), but you may also say "`all`" (e.g. "`-wall`") which does the operation on *all* warnings.

`-w [--warnon]` Makes the message number passed as parameter a warning and turns it on.

`-e [--erroron]` Makes the message number passed as parameter an error and turns it on.

`-m [--msgon]` Makes the message number passed as parameter a message and turns it on. Messages are not counted.

`-n [--nowarn]` Turns the warning/error number passed as a parameter off.

`-L [--nolinesupp]` Turns off suppression of messages on a per line basis. This is meant to be used to ensure that no new errors have crept into a document while editing.

Output control flags: Determines the appearance and destination of the error reports.

`-q [--quiet]` Shuts up about copyright information.

`-o [--output]` Normally, all errors are piped to `stdout`. Using this option with a parameter, errors will be sent to the named file instead. Only information relative to the LaTeX file will be sent to that file. Memory problems and similar will as as always be sent to `stderr`. If a file with the name given already exists, it will be renamed to "`foobar.bak`" ("`foobar.$cl`" under MS-DOS), "`foobar`" being the name of the file. See also "`-b`".

`-v [--verbosity]` Specifies how much and how you wish the error reports to be displayed. This is specified in the "`chktexrc`" file; we'll list the default values below. If you wish, you may thus edit the "`chktexrc`" file to add or modify new formats.

The default is mode 1 (that is, the *second* entry in the "`chktexrc`" file), using `-v` without any parameter will give you mode 2.

0 Will show the information in a way that should be suitable for further parsing by `awk`, `sed` or similar. The format is as follows:

`File:Line:Column:Warning number:Warning message`

The colons may be replaced with another string; use the `-s` switch for this.

As the program does not output all errors in quite order, this output format is also suitable for piping through "`sort`".

1 Shows the information in a way which is more comprehensible for humans, but which still doesn't need anything but a glass tty.

2 Shows the information in a fancy way, using escape codes and stuff. It is the indeed most readable of all modes; however, it needs proper set up of the "`ChkTeX.h`" at compilation time. UNIX boxes, however, will find the information automatically.

3 Shows the information suitable for parsing by Emacs; this is the same format as `lacheck` uses. More formally, it is the following:

`"File", line Line: Warning message`

To utilize this, type `M-x compile RET`. Delete whatever is written in the minibuffer, and type `chktex -v3 texfile.tex`, and you should be able to browse through the error messages. Use `C-x ‘` to parse the messages.

4 More or less the same as `-v3`, but also includes information on where the error actually was found. Takes somewhat longer time to parse, but much more informative in use.

`-f [--format]` Specifies the format of the output. This is done using a format similar to "`printf()`", where we support the specifiers listed below.

Code	Description
`%b`	String to print **b**etween fields (from `-s` option).
`%c`	**C**olumn position of error.
`%d`	Length of error (**d**igit).
`%f`	Current **f**ilename.
`%i`	Turn on **i**nverse printing mode.
`%I`	Turn off **i**nverse printing mode.
`%k`	**k**ind of error (warning, error, message).
`%l`	**l**ine number of error.
`%m`	Warning **m**essage.
`%n`	Warning **n**umber.
`%u`	An **u**nderlining line (like the one which appears when using "`-v1`").
`%r`	Part of line in front of error ('S' − 1).
`%s`	Part of line which contains error (**s**tring).
`%t`	Part of line after error ('S' + 1).

Other characters will be passed literally; thus you can say "`%%`" to achieve a single percent sign in the output. Please note that we may introduce other specifiers in the future, so don't abuse this feature for other characters.

Also, note that we do *not* support field lengths (yet). This may come in the future, if I get the time...

The `-v` command is implemented by indexing into the "`chktexrc`" file; read that for seeing how each format is implemented. If you find yourself using a particular format often by using the -f switch, consider putting it in the "`chktexrc`" file instead.

`-V [--pipeverb]` Which entry we'll use in the "`chktexrc`" file whenever `stdout` isn't a terminal.

The default is to use the same mode as specified with the `-v` switch; using `-V` without any parameter will give you mode 1.

This switch was implemented because GNU less has problems with the escape codes ChkTeX uses for displaying inverse text. Under UNIX, there's another way around, though, which is slightly more elegant. Add the following line to your "`.envir`" file:

```
setenv LESS -r
```

`-p [--pseudoname]` With this switch, you can provide the filename which will be used when we report the errors. This may be useful in scripts, especially when doing pipes. It is in other words similar to C's `#line` directive.

We will only assume this name for the uppermost file; files that this one in turn `\input` are presented under their original names. This seems most logical to me.

`-s [--splitchar]` String to use instead of the colons when doing `-v0`; e.g. this string will be output between the fields.

Boolean switches: Common for all of these are that they take an optional parameter. If it is `0`, the feature will be disabled, if it is `1`, it will be

enabled. All these features are on by default; and are toggled if you don't give any parameter.

`-b [--backup]` If you use the `-o` switch, and the named outputfile exists, it will be renamed to `filename.bak`.

`-I [--inputfiles]` Execute `\input` statements; e.g. include the file in the input. Our input parsing does of course nest; we use an input-stack to keep track of this.

`-H [--headererr]` Show errors found in front of the `\begin{document}` line. Some people keep *lots* of pure TEX code there, which errors can't be detected reliably (in other words, we will in most cases just produce a lot of garbage).

`-g [--globalrc]` Read in the global resource file. This switch may be useful together with the `-l` option.

`-t [--tictoc]` Display a twirling baton, to show that we're working. `-v0` does an `-t0`, too, as it assumes that the user then uses the program non-interactively. This is now a no-op.

`-x [--wipeverb]` Ignore the "`\verb`" command found within the LATEX file and its argument is completely by the checking routines. This is done by simply overwriting them. If you somehow don't like that (for instance, you would like to count brackets inside those commands, too), use this switch.

If you don't specify any input LATEX-files on the commandline, we'll read from `stdin`. To abort `stdin` input, press the following keycombinations:

Machine	**Key-combination**
UNIX	`Ctrl` + `D`
MS-DOS	`Ctrl` + `Z`, followed by return.

By default, we use GNU's "`getopt()`" routine.

- Options may be given in any order; the names of the LATEX-files do not have to be the last arguments. This behaviour may be turned off by creating an environment variable named "`POSIXLY_CORRECT`".
- The special argument "`--`" forces an end of option-scanning.
- Long-named options begin with "`--`" instead of "`-`". Their names may be abbreviated as long as the abbreviation is unique or is an exact match for some defined option. If they have an argument, it follows the option name in the argument, separated from the option name by a "`=`", or else the in next argument.

6.1.3 The "`chktexrc`" file

You should also take a look at the "`chktexrc`" file. As it is self-documenting, you should be able to get the meaning of each keyword by simply reading the file. In fact, since not all options are described in this documentation it is necessary that you read the "`chktexrc`" file in order to understand them all. The method for *finding it* has grown rather complex. An outline is given below.

If ChkTeX finds multiple files when searching, each and every one will be read in the order specified below. The "`Keyword = { item item ...}`" may thus be necessary to reset previously defined lists.

In this list, "`$foo`" is assumed to be the environment variable "`foo`":

1. First, we'll take a look at the directory which was specified as "`DATADIR`" during compilation. On UNIX boxes, this usually evaluates to something similar to "`/usr/local/share/chktexrc`", under MS-DOS it is set to "`\emtex\data\chktexrc`".

2. Look in the following system directories:

Machine	**Directory**
UNIX	"`$HOME/.chktexrc`" or "`$LOGDIR/.chktexrc`"
MSDOS	Program installation path

3. Look for it in the directory pointed to by an environment variable, as specified in the table below:

Machine	**Directory**
UNIX	"`$CHKTEXRC/.chktexrc`"
MSDOS	"`$CHKTEXRC\chktexrc`", "`$CHKTEX_HOME\chktexrc`" or "`$EMTEXDIR\data\chktexrc`"

4. Look for it in the current directory. On UNIX boxes, we expect the filename to be "`.chktexrc`"; on other machines "`chktexrc`".

If you for some reason wish to undo what the previous files may have done, you may say "`CmdLine { -g0 -r }`" somewhere in the "`chktexrc`" file; this will reset all previous settings.

6.1.4 Per Line and File Suppressions

There are many cases in which ChkTeX will give a warning about a construct which, although it usually indicates a mistake, is intentional. In these cases it can be extremely annoying to have this message appear everytime ChkTeX is run. For this reason you can use LaTeX comments to suppress a certain message on a single line. This can be done by adding a (case-insensitive) comment at the end of the line of the form

```
% chktex ##
```

where `##` is the number of the message to be suppressed. For example the line

```
$[0,\infty)$
```

will produce a warning (number 9) about mismatched `]` and `)`. However the lines

```
$[0,\infty)$ % chktex 9
$[0,\infty)$ % ChkTeX 9
```

will not produce such a message. In this case, message number 17 will still appear at the end, stating that the numbers of `]` and `)` don't match for the entire file.

To suppress two different errors on the same line you will need two instances of `chktex` in the comment. This format is a little cumbersome, but it shouldn't be needed often, and hopefully will help avoid accidental suppressions.

```
Jordan--H\"older on $[0,\infty)$ % chktex 8 chktex 9
```

One problem inherent in line-by-line suppressions is that during editing another error of the same type may creep into the same line. Therefore, I suggest using the `-L` or `--nolinesupp` option to disallow line based suppressions once just before the document is finished. At that point you should go back over all the warnings and decide if they should be fixed.

In addition to line-specific suppressions, you can create a suppression which will be in effect for the remainder of the file. This can be used, for example, to turn off warning 18 (about `"`) in a file which uses a package (like "`babel`") where `"` is an active character. The syntax is nearly the same, namely

```
% chktex-file ##
```

6.1.5 The "`chktexrc`" file format

The chktexrc file is essentially a bunch of variable assignments. There are two types of variables, those that take single items and those that take lists.

In turn, there are two types of lists, case-sensitive and case-insensitive. Case-sensitive lists are delimited by `{` and `}` while case-insensitive are delimited by `[` and `]`. Only some variables support case insensitive lists, since in many cases it doesn't make sense and would be unnecessarily slow. Those variables that *do* support case-insensitive lists will be marked as such throughout the file.

Variables can be set with or without an equals sign. If included, the `=` causes the variable to be overwritten. This is the only thing that makes sense for variables taking a single item and so we always include it in that case. For list variables, omitting the equals sign will cause the items in the list to be appended instead of overwriting the entire list.

Below are all the ways in which a variable can be set. Note that lists can span lines, though this is not shown here for brevity.

```
VariableName = item
# Overwrites
VariableName = { Item1 Item2 ... }
VariableName = [ item1 item2 ... ]
VariableName = { Item1 Item2 ... } [ item item ... ]
VariableName = [ item1 item2 ... ] { Item Item ... }
# Appends
VariableName { Item3 Item4 ... }
VariableName [ item3 item4 ... ]
VariableName { Item3 Item4 ... } [ item item ... ]
VariableName [ item3 item4 ... ] { Item Item ... }
```

Comments begin with `#`, and continue for the end of the line. Blank lines plus leading and trailing spaces are ignored. Items are separated by spaces. Newlines are considered spaces, but can't be escaped. You can use double quotes `"` to surround an item with spaces, or you can escape spaces as described later.

Detection of tokens like `}` are somewhat context sensitive—they have to be preceded by a space (or newline). This allows them to be part of an item without escaping. Since some variables require such characters, this generally makes life easier.

To include characters that might otherwise interfere, escape sequences are provided. They are similar to those in C, but use `!` instead of \ for obvious reasons. The entire list is below.

Sequence	**Resulting character**
`!`	Space
`!"`	"
`!#`	#
`!!`	!
`!{`	{
`!}`	}
`![`	[
`!]`	]
`!=`	=
`!b`	Backspace
`!n`	New line
`!r`	Carriage return
`!t`	Tab
`!f`	Form feed
`!xNN`	NN must be a hexadecimal number (00 - ff), both characters must be included.
`!dNNN`	NNN must be a decimal number (000 - 255), all three characters must be included. Unspecified results if NNN > 377.
`!NNN`	NNN must be a octal number (000 - 377), all three characters must be included. Unspecified results if NNN > 377.

6.1.6 Settings in the "`chktexrc`" file

All available settings follow.

`QuoteStyle` Simple Variable

The type of quote-style you are using. There are currently two styles:

Traditional:

"An example," he said, "would be great."

Logical:

"An example", he said, "would be great".

The default value is

```
QuoteStyle = Logical
```

`TabSize` Simple Variable

The width of a tab. This is used for formatting the error message. Only positive integers are allowed.

The default value is

```
TabSize = 8
```

`CmdSpaceStyle` Simple Variable

How to treat a command is followed by punctuation. In all cases the warnings are also governed by the main warning settings, namely warnings 12 and 13 about interword and intersentence spacings. These can be found on page **??**.

If set to Ignore, then it won't print any warnings when punctuation follows a command.

If CmdSpaceStyle is set to InterWord, then it will print warnings when interword spacing should (potentially) be used. For example, without a command the following will trigger warning 12

```
I've seen a UFOs, etc. in my life.
```

And if set to InterWord, so will

```
I've seen a UFOs, \etc. in my life.
```

If set to InterSentence, then it will print warnings when intersentence spacing should (potentially) be used. For example, without a command the following will trigger warning 13

```
I've seen an UFO! Right over there!
```

And if set to InterSentence, so will

```
I've seen an \UFO! Right over there!
```

Setting CmdSpaceStyle to Both will cause warnings to be printed in both cases.

The default value is

```
CmdSpaceStyle = Ignore
```

`CmdLine` List Variable

Default command-line options. For instance, you might like to put `-v2` here.

The default value is

```
CmdLine
{
}
```

`UserWarn` Lowercase-able List Variable

Arbitrary strings to warn about. You can put here to help you find your own little foibles. See also UserWarnRegex.

These patterns will be searched for throughout the text; regardless of whether they appear as normal text, commands, in math mode, etc. They are *not* found in comments.

Suppose you define a special command like this:

```
\def\unknown{\large\bf??}
```

which you use whenever you don't have some information at the time of writing. Thus, it makes sense to warn on it, and this is a convenient way to do so.

The default value is

```
UserWarn
{
    \unknown
    # One should write \chktex or Chk\TeX - never ChkTeX.
    ChkTeX
}
[ # You may put case-insensitive patterns here.
]
```

`UserWarnRegex` List Variable

A more sophisticated version of UserWarn using regular expressions. Use of these will be automatically disabled if ChkTeX was built without regular expression support. Because ChkTeX can be with support for either POSIX or PCRE flavors of regular expression, some of the following will not apply in all cases. An expression can be defined only when PCRE is enabled by prepending the expression with `PCRE:` and similarly with `POSIX:`.

These patterns will be searched for, no matter whether they appear as normal text, commands, or arguments. However, they will *not* match in verbatim environments (see VerbEnvir).

Remember that you have to escape (with a `!`) the characters `"#!=`, as well as spaces and `{}[]` if they are proceeded by a space.

When using PCRE regular expressions, you can use `(?i)` to make the expression case insensitive. See the man pages (man pcresyntax) or the nicely formatted http://perldoc.perl.org/perlre.html for documentation on the regular expression syntax. Note, however, that some the features of perl regular expression are not available such as running code (callouts), and replacing.

An initial PCRE-style comment `(?# ... )` can be used to change what is displayed, thereby reminding yourself how to fix the problem. This works even for POSIX expressions.

The default value is

```
UserWarnRegex
{
    (?!#Always! use! \nmid)\\not! *(\||\\mid)

    # Capitalize section when saying Section 6.
    (?!#-1:Capitalize! before! references)PCRE:\b(chapter|(sub)?section|theorem|lemma
    (?!#1:Capitalize! before! references)POSIX:([^[:alnum:]]|^)(chapter|(sub)?section

    # Spell it introduction
    # PCRE:(?i)\bintro\b(?!#Spell! it! out.! This! comment! is! not! used.)
    # POSIX:([^[:alnum:]]|^)intro([^[:alnum:]]|$)

    # Pretty tables--see http://texdoc.net/texmf-dist/doc/latex/booktabs/booktabs.pdf
    (?!#-2:Use! \toprule,! \midrule,! or! \bottomrule! from! booktabs)\\hline
    # This relies on it being on a single line, and not having anything
    # else on that line.  With PCRE we could match balanced [] and {},
    # but I wonder if it's worth the complexity...
    (?!#-2:Vertical! rules! in! tables! are! ugly)\\begin\{(array|tabularx?\*?)\}(\[.*

    (?!#-3:Optional! arguments! []! inside! optional! arguments! []! must! be! enclose

}
```

`TeXInputs` List Variable

A list of paths where ChkTEX should look for files it `\inputs`. By default, the current directory is searched (not recursively, use `//` for that, see below).

A `//` postfix is supported: if you append a double path-separator we'll recursively search that directory's directories. MS-DOS users must append instead, e.g. `C:\EMTEX\\`. In order to search an entire directory tree, you must use *three* slashes, e.g. `c:\\\` or `///`.

The default value is

```
TeXInputs
{
}
```

`OutFormat` List Variable

Output formats which can be selected from the command-line. The `-v` option simply indexes into this list. By default, entry number *two* in this list is chosen (counting from 0), and `-v` without any parameter selects entry number *three*.

For explanation of the `%` format specifiers see the documentation of the `--format` command-line argument on page **??**.

Recall that to use `!` is the escape character, not \.

The default value is

```
OutFormat
{
    # -v0; silent mode
    %f%b%l%b%c%b%n%b%m!n
    # -v1; normal mode
    "%k %n in %f line %l: %m!n%r%s%t!n%u!n"
    # -v2; fancy mode
    "%k %n in %f line %l: %m!n%r%i%s%I%t!n!n"
    # -v3; lacheck mode
    "!"%f!", line %l: %m!n"
    # -v4; verbose lacheck mode
    "!"%f!", line %l: %m!n%r%s%t!n%u!n"
    # -v5; no line number, ease auto-test
    "%k %n in %f: %m!n%r%s%t!n%u!n"
    # -v6; emacs compilation mode
    "!"%f!", line %l.%c:(#%n) %m!n"
}
```

`Silent` Lowercase-able List Variable

Commands which should not trigger a warning if terminated by a space. This warning will not trigger in math mode.

You can also specify regular expressions in the `[]` section in case you have many custom macros that can be safely terminated with a space.

The default value is

```
Silent
{
    \rm \em \bf \it \sl \sf \sc \tt \selectfont
    \rmfamily \sffamily \ttfamily \mdseries \bfseries \itshape
    \slshape \scshape \relax
    \vskip \pagebreak \nopagebreak

    \textrm \textem \textbf \textit \textsl \textsf \textsc \texttt

    \clearpage \ddots \dotfill \flushbottom \fussy \indent \linebreak
    \onecolumn \pagebreak \pushtabs \poptabs \scriptsize \sloppy
    \twocolumn \vdots
    \today \kill \newline \thicklines \thinlines

    \columnsep \space \item \tiny \footnotesize \small \normalsize
    \normal \large \Large \LARGE \huge \Huge \printindex

    \newpage \listoffigures \listoftables \tableofcontents
    \maketitle \makeindex

    \hline \hrule \vrule
```

```
    \centering

    \noindent \expandafter

    \makeatletter \makeatother

    \columnseprule

    \textwidth \textheight \hsize \vsize

    \if \fi \else

    \csname \endcsname

    \z@ \p@ \@warning \typeout

    \dots \ldots \input \endinput \nextline \leavevmode \cdots
    \appendix \listfiles \and \quad \bigskip \medskip \smallskip
    \hskip \vfill \vfil \hfill \hfil \topmargin \oddsidemargin
    \frenchspacing \nonfrenchspacing
    \begingroup \endgroup \par

    \vrefwarning \upshape \headheight \headsep \hoffset \voffset
    \cdot \qquad \left \right \qedhere \xspace

    \addlinespace \cr \fill \frontmatter
    \toprule \midrule \bottomrule
}[
    # Here you can put regular expressions to match Silent macros.  It
    # was designed for use with many custom macros sharing a common
    # prefix, but can of course be used for other things.

    # Support ConTeXt to at least some extent
    \\start.* \\stop.*
]
```

`HyphDash` List Variable

The number of dashes allowed between two alphabetic characters. Use 0 to always return an error. For example:

foo-bar
Use of two–dashes is not usually allowed in English.
like this—see?

For English, this will often be a single dash (hyphen). If you like m-dashes with no spaces between them and the surrounding words, then it should include 3 as well. There *are* cases when an n-dash is valid between two alphabetic characters. See DashExcpt.

The default value is

```
HyphDash { 1 3 }
```

`NumDash` List Variable

The number of dashes allowed between two numeric characters. Use 0 to always return an error. This does *not* apply in math mode. For example:

123–456 is a range
$12 - 4$ % okay because it's in math mode

For English, this should be 2 because an n-dash is used to indicate a range of numbers and subtraction should be in math mode where this does not apply.

The default value is

```
NumDash  { 2 }
```

`WordDash` List Variable

The number of dashes allowed between two space characters. Use 0 to always return an error. For example:

not like - this,
or like – this.
like this — see?

The default value is

```
WordDash { 3 }
```

`DashExcpt` List Variable

Exceptions to the dash rules above. For example, an n-dash `--` between words is usually wrong, but in some cases it is correct, such as when naming a theorem. The Birch–Swinnerton-Dyer conjecture is one example where the difference matters. You can tell that Birch is one person and Swinnerton-Dyer is another based on the dashes used.

Adding line suppressions for these is possible, but can quickly become tedious if a certain theorem is referenced often. For this reason exceptions can be specified here. They are case-sensitive.

Unfortunately, there are no warnings if the dashes are surrounded by differing types of characters. For example:

like this —see? (space and alphabet)
a–123 (number and alphabet)
a.–b. (other character, namely `.`)

Similarly, no warnings are issued if the hyphenation is correct, according to the other rules, for example:

Birch-Swinnerton-Dyer

The default value is

```
DashExcpt
{
    Birch--Swinnerton-Dyer
}
```

`WipeArg` List Variable

Commands whose arguments aren't LaTeX code, and thus should be ignored.

After the command, you may place arguments (separated from the command with a colon) that should be wiped. Use `[]` for optional arguments, `{}` for required ones, and `*` if the command supports a star variant. Some commands (e.g. `\cmidrule`) use `()` to delimit an optional argument and so this syntax is supported as well.

For instance, if you would like to wipe the `\newcommand` command, you would declare it as `\newcommand:*[][]{}` since it has a star variant, two optional arguments, and one required argument.

These commands may be "evaluated" before they're wiped, so you will typically list file handling commands and similar here.

The default value is

```
WipeArg
{
    \label:{} \ref:{} \eqref:{} \vref:{} \pageref:{} \index:[]{}
    \cite:[][]{} \nocite:{}
    \input:{} \verbatiminput:[]{} \listinginput:[]{}{}
    \graphicspath:{}
    \verbatimtabinput:[]{} \include:{} \includeonly:{}
    \bibitem:[]{}
    \cline:{} \cmidrule:[](){}
    \href:{}{}
    # Cleveref -- there are many others that could be here as well...
    \cref:*{} \cpageref:*{} \crefrange:*{}{} \cpagerefrange:*{}{}
    \Cref:*{} \Cpageref:*{} \Crefrange:*{}{} \Cpagerefrange:*{}{}
    # natbib
    \citet:*[][]{} \citep:*[][]{} \citealt:*{} \citealp:*[]{} \citeauthor:*{}
    \Citet:*[][]{} \Citep:*[][]{} \Citealt:*{} \Citealp:*[]{} \Citeauthor:{}
    \citetext:{} \citeyear:*{} \citeyearpar:{}
    # biblatex - not including special commands
    \autocite:*[][]{} \autocites:*[][]{} \Autocite:*[][]{} \Autocites:*[][]{}
    \parencite:*[][]{} \parencites:*[][]{} \Parencite:*[][]{} \Parencites:*[][]{}
```

```
    \footcite:*{} \footcites:*[][]{} \Footcite:*[][]{} \Footcites:*[][]{}
    \textcite:*{} \textcites:*[][]{} \Textcite:*[][]{} \Textcites:*[][]{}
    \citeauthor:*{} \citeauthors:*[][]{} \Citeauthor:*[][]{} \Citeauthors:*[][]{}
    \citeyear:*{} \citeyears:*[][]{} \Citeyear:*[][]{} \Citeyears:*[][]{}
    \citetitle:*{} \citetitles:*[][]{} \Citetitle:*[][]{} \Citetitles:*[][]{}
    # tipa which uses "
    \textipa:{}
    # LuaTeX
    \directlua:{} \luaescapestring:{}
    # beamer - this doesn't handle environments, and there are
    # probably tons more commands as well...
    \item:<> \mode:<> \uncover:<> \only:<> \alert:<> \onslide:*<>
    \visible:<> \invisible:<> \alt:<> \temporal:<> \label:<>
}
```

`MathEnvir` List Variable

Environments which typeset their contents as mathematics. This turns on/off some warnings.

A `*` variant is automatically added for each keyword.

The default value is

```
MathEnvir
{
    displaymath math eqnarray array equation
    align alignat gather flalign multline
    dmath dgroup darray
}
```

`TextEnvir` List Variable

Environments which typeset their contents as text, for use inside mathematics. This turns on/off some warnings.

The default value is

```
TextEnvir
{
    dsuspend
}
```

`MathCmd` List Variable

Commands whose argument will be typeset as mathematics. The commands are assumed to have one mandatory argument which is in math mode. This turns on/off some warnings.

The default value is

```
MathCmd
{
    \ensuremath
}
```

`TextCmd` List Variable

Commands whose argument will *not* be typeset as mathematics even if it would otherwise be in math mode. The commands are assumed to have one mandatory argument which is in text mode. This turns on/off some warnings.

The default value is

```
TextCmd
{
    \text \intertext \shortintertext \mbox \condition
}
```

`VerbEnvir` List Variable

Environments containing non-LaTeX content of some kind, and therefore should not trigger any warnings.

A `*` variant is automatically added for each keyword.

The default value is

```
VerbEnvir
{
    verbatim comment listing verbatimtab rawhtml errexam picture texdraw
    filecontents pgfpicture tikzpicture minted lstlisting IPA
}
```

`Abbrev` Lowercase-able List Variable

Abbreviations not automatically handled by ChkTeX.

ChkTeX automagically catches most abbreviations; the ones we need to list here, are those which are most likely to be followed by a word with an upper-case letter which is not the beginning of a new sentence.

The case-insensitive abbreviations are not fully case-insensitive. Rather, only the first character is case-insensitive, while the remaining characters are case-sensitive.

To speed up the searching process somewhat, we require that these end in a `.` which should not be a problem in practice.

Much of this work (both the abbreviations below, and the regular expressions necessary to catch the remaining automatically) have been provided by Russ Bubley, <russ@scs.leeds.ac.uk>.

The default value is

```
Abbrev
{
    # Ordinals
    1st. 2nd. 3rd. 4th.
    # Titles
```

```
    Mr. Mrs. Miss. Ms. Dr. Prof. St.
    #
    # Days
    # Mon. Tue. Wed. Thu. Fri. Sat. Sun.
    #
    # Months
    # Jan. Feb. Mar. Apr. May. Jun. Jul. Aug. Sep. Oct. Nov. Dec.
    #
    # Letters
    # Kt. Jr.
    #
    # Corporate
    # Co. Ltd.
    #
    # Addresses
    # Rd. Dr. St. Ave. Cres. Gdns. Sq. Circ. Terr. Pl. Arc. La. Clo. Ho. Est. Gn.
    #
    # Misc.
    # oe. pbab. ps. rsvp. Tx.
}
[
    # The first letter is case-insensitive in the abbrevs in this
    # list. Due to the nature of the checking algorithm used for
    # this, entries consisting of only one character will be
    # silently ignored.
    #
    # Latin
    # cf. "et al." etc. qed. qv. viz.
    #
    # Corporate
    # inc. plc.
    #
    # Misc
    # fax. pcs. qty. tel. misc.
]
```

`IJAccent` List Variable

Commands which add accents above characters. This means that `\i` or `\j` (`\imath` and `\jmath` in math mode) should be used instead of `i` and `j`.

Other accent commands such as `\c`, `\d`, and `\b`, put their accent under the character, and thus should be used with normal `i`s and `j`s.

The default value is

```
IJAccent
{
    \hat \check \breve \acute \grave \tilde \bar \vec \dot \ddot
```

```
    \' \` \^ \" \~ \= \. \u \v \H \t
}
```

`Italic` List Variable

Commands which need italic correction when the group is terminated.

The default value is

```
Italic
{
    \it \em \sl \itshape \slshape
}
```

`NonItalic` List Variable

Commands which makes the font non-italic.

The default value is

```
NonItalic
{
    \bf \rm \sf \tt \sc
    \upshape
}
```

`ItalCmd` List Variable

Commands which put their argument into italic (and thus possibly needs italic correction in the end).

This is currently empty, since `\textit`, `\textsl`, and `\emph` automatically add italic correction.

The default value is

```
ItalCmd
{
}
```

`PostLink` List Variable

Commands in front of which a page break is highly undesirable. Thus there should be no space in front of them.

The default value is

```
PostLink
{
    \index \label
}
```

`NotPreSpaced` List Variable

Commands that should not have a space in front of them for various reasons. Much the same as PostLink, but produces a different warning.

The default value is

```
NotPreSpaced
{
    \footnote \footnotemark \/
}
```

`Linker` List Variable

Commands that should be prepended with a `~`. For example

```
look in table~\ref{foo}
```

to avoid the references being split across lines.

The default value is

```
Linker
{
    \ref \vref \pageref \eqref \cite
}
```

`CenterDots` List Variable

Commands or characters which should have `\cdots` in between. For example, $1 + 2 + 3 + \cdots + n$.

The default value is

```
CenterDots
{
    = + - \cdot \div & \times \geq \leq < >
}
```

`LowDots` List Variable

Commands or characters which should have `\ldots` in between. For example, $1, 2, 3, \ldots, n$.

The default value is

```
LowDots
{
    . , ;
}
```

`MathRoman` List Variable

Words that should appear in roman (upright) in math mode. There are certain aliases for mathematical operators (like sin or cos) that appear in roman rather than the usual italic (slanted) font.

These entries do not need a leading slash since the mistake is often to *not* include the leading slash.

The default value is

```
MathRoman
{
    log lg ln lim limsup liminf sin arcsin sinh cos arccos cosh tan
    arctan tanh cot coth sec csc max min sup inf arg ker dim hom det
    exp Pr gcd deg bmod pmod mod
}
```

`Primitives` List Variable

Commands that are used in TEX but have become unnecessary in LATEX, as there are LATEX commands which do the same. Purists should thus avoid these in their code.

The default value is

```
Primitives
{
   \above \advance \catcode \chardef \closein \closeout \copy \count
   \countdef \cr \crcr \csname \delcode \dimendef \dimen \divide
   \expandafter \font \hskip \vskip \openout
}
```

`NoCharNext` List Variable

Commands and a set of characters that should *not* follow them. For example, in math mode, `\left` should be followed by a delimiter which is to change size. Therefore, it should not be followed by the end of math mode `$` or a grouping character `{` or `}`.

The format is `\command:characters`.

The default value is

```
NoCharNext
{
   \left:{}$ \right:{}$
}
```

`VerbClear` Simple Variable

The character to replace verbatim text with.

The arguments of commands listed in WipeArg, as well as `\verb+...+` commands, are replaced with an innocuous character to prevent that data from inadvertently producing a warning.

This should not contain an alphabetic character (in case the user writes (`\foo\verb+bar+`), neither should it contain be one of LaTeX's reserved characters (`#$%&~_^\{}`), or any parenthesis character (`()[]{}`). If possible, don't use a punctuation character or any spacing characters either. All of these characters have warnings associated with them and thus could cause spurious warnings to appear.

The asterisk is also unsuitable, as some commands behave in another way if they are appended with an asterisk. Which more or less leaves us with the pipe.

Please note that this may also be a `string`, which will be repeated until the proper length is reached.

The default value is

```
VerbClear = "|"
```

6.1.7 Hints

I've tried to collect some advice that may be useful — if you have a favourite hint, feel free to send it to me!

- If you use "`german.sty`" or several of "`babel`" languages which use `"` as an active character; it may be wise to put "`-n18`" in the "`CmdLine`" entry in the "`chktexrc`" file. This will probably reduce the amount of false warnings significantly. Alternately, you can put `% chktex-file 18` in your files which use one of these packages so that other files will still have these checks performed.
- Put "`-v`" in the "`CmdLine`" entry of the "`chktexrc`" file; this makes the fancy printing the default.
- If you're working on a large project, it may pay off to make a local resource file which is included in addition to the global one. In this one, add the necessary info to reduce the amount of false warnings — these usually don't do anything but hide the real warnings.
- Create a total ignore environment, which ChkTeX will ignore completely. In here, you can place all that code which outsmarts ChkTeX completely. For instance, add the following lines at the top of your LaTeX file:

  ```
  % ChkTeX will ignore material within this environment
  \newenvironment{ignore}{}{}
  ```

  In addition, you should add the item "`ignore`" to the "`VerbEnvir`" entry in the "`chktexrc`" file.

6.1.8 Bugs

No fatal ones, I think, but the program currently has some problems when a LaTeX command/parameter stretch over a two lines — some extra spaces may be inserted into the input. I regard the program as fairly well tested; using the SAS/C "`cover`" utility I was able to make sure that approximately 95% of the code has actually been run successfully in the final version. This does indeed leave some lines; most of these are procedure terminating brackets or "can't happen" lines, though.

We've got some problems when isolating the arguments of a command. Although improved, it will certainly fail in certain cases; ChkTeX can for instance not handle arguments stretching over two lines. This also means that "`WIPEARG`" entries in the "`chktexrc`" file will only have the first half of their argument wiped if the argument stretches over two lines. We will, however, take care not to wipe parenthesis in such cases, in order to avoid false warnings.

Long lines are broken up into chunks and handled separately. The exact length is platform dependent, though is guaranteed to be at least 256 bytes. The first portions of the line will have line numbers that are 1 less than they should be. Some errors can be missed and some can be added erroneously. A warning will be issued if lines are too long.

Before submitting a bug report, please first see whether the problem can be solved by editing the "`chktexrc`" file appropriately.

6.2 ChkWEB

This shell script is provided for checking CWEB files. The template is as follows:

```
chkweb [options] file1 file2 ...
```

As you may see from the script, it is only a trivial interface towards `deweb` and ChkTeX. It does not support any individual options on the command line — all options found will be passed onto ChkTeX. If "`--`" or a filename is found, the remaining parameters will be ignored. The only real intelligence it features is that it will try to append `.w` to filenames it can't find.

If no filenames are given, we will read from `stdin`.

6.3 DeWEB

This program strips away C code and CWEB commands from CWEB sources. It is called with the following synopsis:

```
deweb file1 file2 ...
```

`deweb` filters away all C & CWEB commands from a CWEB source code. This leaves only the LaTeX code. This stripped code, in turn, may then be passed to a suitable syntax checker for LaTeX, like ChkTeX and `lacheck`, or spell-checkers like `ispell`.

When `deweb` strips away the C code from your CWEB source, it tries to preserve line breaks. This means that the error reports from *your favorite tool* will be correct regarding to line numbers. In most cases, the column position will also be correct. This significantly simplifies finding the errors in the LaTeX

source (in contrast to the output from `cweave`, which output is truly difficult to figure anything out from).

`deweb` accepts a list of filenames on the argument line, and will send its output to `stdout`. If no filenames are given, it will read from stdin, acting as a filter. No options are currently accepted.

Macho users may try to pipe the output from `deweb` directly into LaTeX, theoretically, this should work. This would ease the debugging of the LaTeX code significantly, as when LaTeX complains about wrong syntax, you'll be able to find the erroneous line much more easily. Don't expect that the output looks very much like the final one, though.

`deweb` should now understand all correct `CWEB` opcodes. If it complains about not understanding a correct opcode, please inform the author.

6.3.1 Bugs

`deweb` will not even *compile* under Perl versions before perl v5. Unfortunately, this means that we can't even tell the user why we failed; Perl will just complain about not being able to compile the regexps.

7 Explanation of error messages

Below is a description of all error-messages ChkTeX outputs. Error messages set in *italic type* are turned off by default. Where margin paragraphs are listed in the text, they refer to the keyword in the "`chktexrc`" file which controls the discussed warning.

Silent

Warning 1: `Command terminated with space.`

> You tried to terminate a command with a blank space. Usually, this is an error as these are ignored by LaTeX. In most cases, you would like to have a real space there.
>
> You can also specify regular expressions to match commands which can safely be terminated with a space. They are specified in the "`chktexrc`" file in `[]`, which in some other cases is used to indicate case-insensitive matching. This is used for example to support the `\startXXX` macros of ConTeXt.

```
\LaTeX_is a typesetter.
```
LaTeXis a typesetter.

```
\LaTeX\ is a typesetter.
```
LaTeX is a typesetter.

Linker

Warning 2: `Non-breaking space ('~') should have been used.`

> When reading a document, it is not very pretty when references are split across lines. If you use the `~` character, LaTeX will assign a very high penalty for splitting a line at that point. ChkTeX issues this warning if you have forgot to do this.

```
Please refer to figure_\ref{foo}.
```
Please refer to figure 11.

```
Please refer to figure~\ref{foo}.
```
Please refer to figure 11.

Warning 3: `You should enclose the previous parenthesis with '{}'.`

This is a warning which you may ignore, but for maximum aesthetic pleasure, you should enclose your bracket characters with '{}'s.

```
$_[(ab)^{-1}]_\^{-2}$
```
$[(ab)^{-1}]^{-2}$

```
${[{(ab)}^{-1}]}^{-2}$
```
$[(ab)^{-1}]^{-2}$

Italic
ItalCmd
NonItalic

Warning 4: `Italic correction ('\/') found in non-italic buffer.`

If you try to use the \/ command when ChkTeX believes that the buffer is not outputted as italic, you'll get this warning.

```
This is an\/ example
```
This is an example.

```
This is an example.
```
This is an example.

Italic
ItalCmd
NonItalic

Warning 5: `Italic correction ('\/') found more than once.`

If the buffer is italic, and you try to use the \/ command more than once, you'll get this warning.

```
This {\it example\/\/} is not amusing.
```
This *example* is not amusing.

```
This {\it example\/} is not amusing.
```
This *example* is not amusing.

Italic
ItalCmd
NonItalic

Warning 6: `No italic correction ('\/') found.`

You get this error if ChkTeX believes that you are switching from italic to non-italic, and you've forgot to use the \/ command to insert that extra little spacing. If you use the "`em`" option, you may ignore this warning.

```
This {\it example_} is not amusing, either.
```
This *example* is not amusing, either.

```
This {\it example\/} is not amusing, either.
```
This *example* is not amusing, either.

IJAccent

Warning 7: `Accent command 'command' needs use of 'command'.`

If you're using accenting commands, 'i' and 'j' should lose their dots before they get accented. This is accomplished by using the `\i`, `\j`, `\imath` and `\jmath` command.

```
This is an example of use of accents: \'{i}.
```
This is an example of use of accents: í.

```
This is an example of use of accents: \'{\i}.
```
This is an example of use of accents: í.

HyphDash
NumDash
WordDash
DashExcpt

Warning 8: `Wrong length of dash may have been used.`

This warning suggests that a wrong number of dashes may have been used. It does this by classifying the dash according to the the character in front and after the dashes.

If they are of the same type, ChkTEX will determine which keyword to use in the "`chktexrc`" file. If not, it will shut up and accept that it doesn't know.

Character type	**Keyword in "`chktexrc`" file**
Space	`WordDash`
Number	`NumDash`
Alphabetic character	`HyphDash`

This is more or less correct, according to my references. One complication is that most often a hyphen (single dash) is desired between letters, but occasionally an n-dash (double dash) is required. This is the case for theorems named after two people e.g. Jordan–Hölder. A hyphen would indicate that it was one person with a hyphenated name e.g. Gregorio Ricci-Curbastro. If this is rare enough, it can be dealt with via line based suppressions. However, exceptions can also be handled by adding them to the `DashExcpt` list. The "words" in this list are considered to be correct regardless of any other settings. Adding `Jordan--H\"older` to this list will cause no warning to be issued. There is still the problem that no warning will be raised for Jordan-Hölder (unless added explicitly via regular expression), so care must still be taken.

Some manuals—particularly American manuals—also suggest *not* adding space around an m-dash (triple dash). Hopefully this check can be improved even more (suggestions?).

```
It wasn't anything - just a 2---3 star--shots.
```
It wasn't anything - just a 2—3 star–shots.

```
It wasn't anything --- just a 2--3 star-shots
```
It wasn't anything — just a 2–3 star-shots.

Warning 9: `'%s' expected, found '%s'.`
Warning 10: `Solo '%s' found.`

Either brackets or environments don't match. ChkTEX expects to find matching brackets/environments in the same order as their opposites were found, and no closing delimiters which haven't been preceded by an opening one.

While bracket matching is not an explicit error, it is usually a sign that something is wrong.

CenterDots
LowDots

Warning 11: `You should use '%s' to achieve an ellipsis.`

Simply typing three "." in a row will not give a perfect spacing between the dots. The `\ldots` is much more suitable for this. Similar problems are noted for two periods in a row (instead of three) since lacheck does.

In math mode, you should also distinguish between `\cdots` and `\ldots`; take a look at the example below.

```
Foo...bar. $1,...,3$. $1+...+3$. $1,\cdots,3$.
```
Foo...bar. $1, ..., 3$. $1 + ... + 3$. $1, \cdots, 3$.

```
Foo\dots bar. $1,\ldots,3$. $1+\cdots+3$. $1,\ldots,3$.
```
Foo… bar. $1, \ldots, 3$. $1 + \cdots + 3$. $1, \ldots, 3$.

Abbrev
CmdSpaceStyle

Warning 12: `Interword spacing ('\ ') should perhaps be used.`

One of the specified abbreviations were found. Unless you have previously said `\frenchspacing`, you'll have incorrect spacing, which one should avoid if possible. The error will be suppressed if you have used `\frenchspacing`.

You can also specify case-insensitive abbreviations in `[]`, though only the first letter is actually case-insensitive.

```
This is an example, i.e._a demonstration.
```
This is an example, i.e. a demonstration.

```
This is an example, i.e.\ a demonstration.
```
This is an example, i.e. a demonstration.

CmdSpaceStyle

Warning 13: `Intersentence spacing ('\@') should perhaps be used.`

LaTeX's detection of whether a period ends a sentence or not, is only based upon the character in front of the period. If it's uppercase, it assumes that it does not end a sentence. While this may be correct in many cases, it may be incorrect in others. ChkTEX thus outputs this warning in every such case.

The error will be suppressed if you have used `\frenchspacing`.

```
I've seen an UFO!_Right over there!
```
I've seen an UFO! Right over there!

```
I've seen an UFO\@! Right over there!
```
I've seen an UFO! Right over there!

Warning 14: `Could not find argument for command.`

ChkTEX will in some cases need the argument of a function to detect an error. As ChkTEX currently processes the LATEX file on a line-by-line basis, it won't find the argument if the command which needed it was on the previous line. On the other hand, this *may* also be an error; you ought to check it to be safe.

```
$\hat$
```
This will give a LATEX error...

```
$\hat{a}$
```
$\hat{a}$

Warning 15: `No match found for '%s'.`

This warning is triggered if we find a single, *opening* bracket or environment. While bracket matching is not an explicit error, it is usually a sign that something is wrong.

`MathEnvir`

Warning 16: `Mathmode still on at end of LaTeX file.`

This error is triggered if you at some point have turned on math-mode, and ChkTEX couldn't see that you remembered to turn it off.

Warning 17: `Number of 'character' doesn't match the number of 'character'.`

Should be self-explanatory. ChkTEX didn't find the same number of an opening bracket as it found of a closing bracket.

Warning 18: `You should use either `` or '' as an alternative to ' " '.`

Self-explanatory. Look in the example, and you'll understand why.

```
This is an "example"
```
This is an "example"

```
This is an ``example''
```
This is an “example”

Warning 19: *You should use "’" (ASCII 39) instead of "’" (ASCII 180).*

On some keyboards you might get the wrong quote. This quote looks, IMHO, *ugly* compared to the standard quotes, it doesn't even come out as a quote! Just see in the example. In fact, this doesn't even compile anymore, so the warning is probably obsolete.

```
‘‘There’s quotes and there’s quotes ’’
```
```
‘‘There’s quotes and there’s quotes’’
```
“There’s quotes and there’s quotes”

UserWarn

Warning 20: User-specified pattern found: %s.

A substring you've specified using `UserWarn` in the “`chktexrc`” file, has been found. See also warning 44 which allows using regular expressions. You can also specify case-insensitive versions in `[]`.

Warning 21: *This command might not be intended.*

I implemented this because a friend of mine kept on making these mistakes. Easily done if you haven't gotten quite into the syntax of LaTeX.

```
\LaTeX\ is an extension of \TeX\. Right?
```
LaTeX is an extension of TeX Right?
```
\LaTeX\ is an extension of \TeX. Right?
```
LaTeX is an extension of TeX. Right?

Warning 22: *Comment displayed.*

ChkTeX dumps all comments it finds, which in some cases is useful. I usually keep all my notes in the comments, and like to review them before I ship the final version. For commenting out parts of the document, the “`comment`” environment is better suited. Setting this warning allows you to see notes you have left in comments.

Warning 23: Either ’’\,’ or ’\,’’ will look better.

This error is generated whenever you try to typeset three quotes in a row; this will not look pretty, and one of them should be separated from the rest.

```
‘‘‘Hello’, I heard him said’’, she remembered.
```
“‘Hello’, I heard him said”, she remembered.
```
‘‘\,‘Hello’, I heard him said’’, she remembered.
```
“ ‘Hello’, I heard him said”, she remembered.

Warning 24: `Delete this space to maintain correct pagereferences.`

This message, issued when a space is found in front of a `\index`, `\label` or similar command (can be set in the "`chktexrc`" file). Sometimes, this space may cause that the word and the index happens on separate pages, if a pagebreak happens just there.

Warning 42 is similar in that it warns about spaces in front of footnotes. The difference is that the warning text makes more sense for that case.

```
Indexing text_\index{text} is fun!
```

```
Indexing text\index{text} is fun!
```

Warning 25: `You might wish to put this between a pair of '{}'`

This warning is given whenever ChkTEX finds a "`^`" or a "`_`" followed by either two or more numeric digits or two or more alphabetic characters. In most situations, this means that you've forgotten some {}'s.

```
$5\cdot10^10$
```
$5 \cdot 10^1 0$

```
$5\cdot10^{10}$
```
$5 \cdot 10^{10}$

Warning 26: `You ought to remove spaces in front of punctuation.`

This warning is issued if ChkTEX finds space in front of an end-of-sentence character.

```
Do you understand_?
```
Do you understand ?

```
Do you understand?
```
Do you understand?

Warning 27: `Could not execute LaTeX command.`

Some LaTeX commands will be interpreted by ChkTEX; however, some of them are sensible to errors in the LaTeX source. Most notably, the `\input` command requires that the input file exist...

Italic
ItalCmd
NonItalic

Warning 28: `Don't use \/ in front of small punctuation.`

Italic correction should generally *not* be used in front of small punctuation characters like '.' and ','; as it looks better when the preceding italic character leans "over" the punctum or comma.

```
It is just a {\it test\/}, don't think anything else.
```
It is just a *test*, don't think anything else.

```
It is just a {\it test}, don't think anything else.
```
It is just a *test*, don't think anything else.

Warning 29: `$\times$ may look prettier here.`

In ASCII environments, it is usual to use the 'x' character as an infix operator to denote a dimension. The mathemathical symbol × provided by the `$\times$` command is better suited for this.

```
The program opens a screen sized 640x200 pixels.
```
The program opens a screen sized 640x200 pixels.

```
The program opens a screen sized $640\times200$ pixels.
```
The program opens a screen sized 640×200 pixels.

Warning 30: *`Multiple spaces detected in output.`*

This warning, intended for the novice, will remind you that even if you *type* multiple spaces in your input, only a single space will come out. Some ways to come around this is listed below.

```
White          is a beautiful colour.
```
White is a beautiful colour.

```
White~~~~~{ }{ }{ }\ \ \ is a beautiful colour.
```
White is a beautiful colour.

VerbEnvir

Warning 31: `This text may be ignored.`

Certain implementations of the `verbatim` environment and derivations of that, ignore all text on the same line as `\end{verbatim}`. This will warn you about this.

Warning 32: `Use ` to begin quotation, not '.`

Warning 33: `Use ' to end quotation, not `.`

Warning 34: `Don't mix quotes.`

Proper quotations should start with a ` and end with a '; anything else isn't very pretty. Both these warnings are relative to this; look in the example below.

```
There are ``examples'' and there are ``examples''.
```
There are ''examples" and there are "examples".

```
There are ``examples'' and there are ``examples''.
```
There are "examples" and there are "examples".

MathRoman

Warning 35: `You should perhaps use ‘cmd’ instead.`

Most mathematical operators should be set as standard roman font, instead of the math italic LaTeX uses for variables. For many operators, LaTeX provides a pre-defined command which will typeset the operator correctly. Look below for an illustration of the point.

```
$sin^2 x + cos^2 x = 1$
```
$$sin^2 x + cos^2 x = 1$$
```
$\sin^2 x + \cos^2 x = 1$
```
$$\sin^2 x + \cos^2 x = 1$$

Warning 36: `You should put a space in front of/after parenthesis.`

Warning 37: `You should avoid spaces in front of/after parenthesis.`

Outside math mode, you should put a space in front of any group of opening parenthesis, and no spaces after. If you have several after each other, you should of course not put a space in between each; look in the example below. Likewise, there should not be spaces in front of closing parenthesis, but there should be at least one after.

```
This( an example( Nuff said )), illustrates( ‘‘my’’ )point.
```
This(an example(Nuff said)), illustrates(“my”)point.

```
This (an example (Nuff said)), illustrates (‘‘my’’) point.
```
This (an example (Nuff said)), illustrates (“my”) point.

QuoteStyle

Warning 38: `You should not use punctuation in front of/after quotes.`

For best looking documents, you should decide on how you wish to put quotes relative to punctuation. ChkTeX recognizes two styles; you may specify which you use in the “`chktexrc`” file. A description on each style follows:

Traditional: This style is the most visually pleasing. It always puts the punctuation *in front of* the quotes, which gives a continuous bottom line.

However, it may in certain cases be ambigious. Consider the following example from a fictious “`vi(1)`” tutorial (quote taken from the Jargon file):

```
Then delete a line from the file by typing ‘‘dd.’’
```
Then delete a line from the file by typing “dd.”

That would be very bad — because the reader would be prone to type the string d-d-dot, and it happens that in “`vi(1)`” dot repeats the last command accepted. The net result would be to delete *two* lines! This problem is avoided using logical style, described below.

Logical: This style uses quotes as balanced delimiters like parentheses. While this is not the most visual pleasing, it is can't be misunderstood. The above sentence would then become:

```
Then delete a line from the file by typing ``dd''.
```
Then delete a line from the file by typing "dd".

Warning 39: Double space found.

This warning is triggered whenever ChkTEX finds a space in front of a hard space, or vice versa. This will be rendered as two spaces (which you usually don't wish).

```
For output codes, see table_~_\ref{foo}.
```
For output codes, see table 1.1.

```
For output codes, see table~\ref{foo}.
```
For output codes, see table 1.1.

MathEnvir

Warning 40: You should put punctuation outside inner/inside display math mode.

As recommended in the TEXbook, you should try to put punctuation outside inner math mode, as it is formatted better.

Similarily, you should let any final punctuation in display math mode end up within it. Look at the following example, which was taken from the TEXbook:

```
for $x = a,b$, or $c$.
```
for $x = a, b$, or c.

```
for $x = a$, $b$, or $c$.
```
for $x = a$, b, or c.

Primitives

Warning 42: *You ought to not use primitive TeX in LaTeX code.*

This warning is triggered whenever you use a raw TEX command which has been replaced by a LATEX equivalent. If you consider yourself a purist (or want to be sure your code works under LATEX3), you should use the LATEX equivalent.

Warning 42: You should remove spaces in front of '%s'

NotPreSpaced

Some commands should not be prepended by a space character, for cosmetical reasons. This notes you of this whenever this has happened.

```
This is a footnote_\footnotemark[1] mark.
```
This is a footnote [1] mark.

```
This is a footnote\footnotemark[1] mark.
```
This is a footnote[1] mark.

NoCharNext

Warning 43: `'%s' is normally not followed by '%c'.`

LaTeX' error message when calling `\left \{` instead of `left \{` is unfortunately rather poor. This warning detects this and similar errors.

Warning 44: `User Regex: %s.`

UserWarnRegex

A pattern you've specified using `UserWarnRegex` in the "`chktexrc`" file, has been found. See also warning 20 which allows specification of simple string matching.

Depending on how ChkTeX was configured, you can use either PCRE regular expressions, POSIX extended regular expressions, or none at all. A warning will be issued if ChkTeX was built without regular expression support, but you try to use one.

By default the matching portion of the line is printed to help distinguish between user specified regular expressions. However, if the regular expression begins with a PCRE comment (which has a syntax of `(?# ... )`), then that comment will be printed instead. This can be used to remind yourself why you were searching for the problem or how to fix it. This applies even if POSIX regular expressions are used since ChkTeX itself parses a single initial PCRE-style comment.

Note: If a regular expression (not a comment) starts with `PCRE:` or `POSIX:` it will be used only if support for that regular expression engine has been compiled in. It is primarily meant to make testing easier but, can be used to allow better regular expressions if PCRE is available. If you want a regular expression that starts with `PCRE:` or `POSIX:` then you can enclose one of the characters in brackets like `[P]CRE:`.

An example, included in the "`chktexrc`" file, is given below. Remember that you have to escape (with `!`) spaces and `#` as well as a few other characters. One should always use `\nmid` instead of `\not\mid` because the results are much better.

```
\\not! *(\||\\mid)
User Regex: \not\mid.
```

or with an initial comment

```
(?!#Always! use! \nmid)\\not! *(\||\\mid)
      User Regex: Always use \nmid.
```

You can use `% chktex 44` to suppress user regular expression warnings on a given line, but this is often undesirable since all such warnings are suppressed this way. For this reason you can "name" user regular expression warnings with negative numbers. For example `% chktex 4` will suppress the system warning number 4, but

`% chktex -4` will suppress the user regular expression warning number 4. Since one might wish to add, remove, or rearrange user warnings in the "`chktexrc`" file, you must explicitly name particular warnings rather than relying on position in that file.

In order to name one, include an initial PCRE-style comment with the first characters being a number (positive or negative—the absolute value will be used). The numbers are limited by the number of bits in a `long`, usually giving 1–63 as possible names. You can give more than one regular expression the same name, and suppressing that name will suppress all regular expressions with that name.

Using the example from before, all of the following will be suppressed with `% chktex -4`. Note that the name *will* be printed as written so that you know which number to suppress.

```
(?!#4:Always! use! \nmid)\\not! *(\||\\mid)
(?!#-4Always! use! \nmid)\\not! *(\||\\mid)
(?!#-4! Always! use! \nmid)\\not! *(\||\\mid)
```

Warning 45: `Use \[ ... \] instead of $$ ... $$.`

In LaTeX documents, using `\[...\]` is strongly preferred over `$$...$$`. This is because using `$$` will change the vertical spacing in the equations making them inconsistent.

Warning 46: *`Use \( ... \) instead of $ ... $.`*

In LaTeX documents using `\(...\)` is slightly preferred over `$...$`. Some error messages might be clearer with `\(...\)`.

Warning 47: `'%s' expected, found '%s' (ConTeXt).`
Warning 48: `Solo '%s' found (ConTeXt).`

Similar to warnings 9 and 10, these tracks environments, but ConTeXt environments rather than LaTeX. They are tracked as separate messages to allow being turned on and off independently thereby allowing people to use macros that start with `\start`.

Warning 49: `Expected math mode to be %s here.`

Similar to warning 16, this warns when ChkTeX got confused about math mode. In theory, for a well formed LaTeX document this should never happen, but the following (invalid) documents will cause the warning. This is because the `$` will exit math mode (which `\ensuremath` entered), and then exiting `\ensuremath` finds that we are in text mode which shouldn't happen.

```
\ensuremath{ \frac{x}{3} $ y\times 7}
```

8 Future plans

In a somewhat prioritized sequence, this is what I'd like to put into the program — if I have the time.

- De-linearize the checker. Currently, it works on a line-by-line basis, in most respects, at least. I hope to be able to remove this barrier; as this will reduce the amount of false warnings somewhat.
- Probably some more warnings/errors; just have to think them out first. Suggestions are appreciated — I've "stolen" most that similar programs provides, and am running out of ideas, really.

  It would also be nice to investigate the field of "globally" oriented warnings; i.e. warnings regarding the document as a whole. Currently, ChkTeX operates mainly on a local/"greedy" basis.

  If you have suggestions/ideas on this topic, they're certainly welcome, including references to literature.
- Fix a few more bugs.

9 Notes

9.1 Wish to help?

As most other living creatures, I have only a limited amount of time. If you like ChkTeX and would like to help improving it, here's a few things I would like to receive. The following ideas are given:

- Does anyone have a LaTeX → `troff` conversion program? It would be really nice if I could extract the relevant sections from this manual, and present them as a man page. I will not, however, convert this manual to TeXinfo in order to be able to do this; IMHO TeXinfo documents have far too limited typographic possibilities.

  This doesn't mean that I'm not willing to restructure the document at all. This manual already uses some kind of preprocessing in order to achieve HTML output via LaTeX2`html`, I'm willing to do the same in order to produce `troff` output.
- Help me port the program! This is a prioritized one. It's no fun writing ANSI C when people haven't got a C compiler.

  Of course, I'll provide whatever help necessary to modify the sources to fit to the new platform. Take contact if you're interested. I will include your compiled binary in the distribution, and give you credit where appropiate.

  Just one request: If you have to modify the sources in order to make ChkTeX compile & work on the new platform, *please* enclose your changes in something like "`#ifdef __PLATFORM__...code...#endif`"! It makes life so much easier later, when we try to merge the two source trees.
- Reports on problems configuring and compiling ChkTeX on supported (and unsupported) systems are welcomed.

- Filters for other file formats. I do believe that there are several formats using LaTeX for its formatting purposes, combining that with something else. If you can write a program or script which filters everything away but the LaTeX code, it will surely be appreciated (and included). Look at the `deweb` script to see what I mean.
- Interfaces for other editors are also welcomed.
- If you update the "`chktexrc`" file in anyway that is not strictly local, I would appreciate to receive your updated version.
- Suggestions for new warnings are always welcomed. Both formal (i.e. regexps or similar) and non-formal (plain English) descriptions are welcomed.

Of course, people doing any of this will be mentioned in this document, and thus receive eternal glory and appreciation.

9.2 Caps and stuff

Where trademarks have been used, the author is aware of that they belong to someone, and has tried to stick to the original caps.

10 About the author

A quick summary of who I am and what I do:

I'm 21 years old, and live in Oslo, the capital of Norway. I'm currently studying maths and computer science at the University of Oslo; planning to get a degree within mathematical modeling, with a dash of physics and emphazing the computer part of the study. More precisely, in autumn'96 my studies consist of mathematical analysis, statistics & probability calculations plus studying the relationship between society and computers.

At home I now possess 4 computers, of which 1 is regular use: A vanilla Amiga 1200, expanded only by a HD. The others are a `80286` PC and an Amiga 500, both semi-out-of-order. The last one is a Commodore VIC-20, which for some peculiar reason never seems to be used. Plans are to get a Linux-capable PC, though.

Most of the time in front of these computers (including SGI Indy's and SPARC stations at our university) is spent on C and shell programming, plus some text-processing.

C and shell programming are not my only knowledge areas regarding computers, however. I write the following languages more or less: Perl, Motorola `68000` assembly code, ARexx, Simula, C++, LaTeX, HTML, AmigaGuide, Amos Basic and Installer LISP. Once I also mastered Commodore Basic V2, the "language" included with my VIC-20.

However, I also try to not to end up as a computer nerd. Thus, in addition to the compulsory (?) interest for computers, I am a scout. Still running into the woods, climbing the trees, falling down and climbing up once more, in other words. To be more specific, I am a now a troop leader for 'Ulven' scout-group; Norwegian Scouts Association. I am also a active rover in 'Vålerenga' scout-group.

Certainly a lot more to tell (I play the piano and like cross-country skiing, for instance); but I'll stop here before you fall asleep...

11 Thanks

The author wishes to thank the following people (in alphabetical order):

Russ Bubley
`russ@scs.leeds.ac.uk`
He has been the main external beta-tester for this program, sending me loads and loads of understandable and reproducible bug reports. If you somehow think that ChkTEX is well-behaved and free from bugs, send warm thoughts to Russ. He has also provided ideas for enhanced checks and so forth.

In addition, he sent me a huge list of 238 common English abbreviations, for inclusion in the "`chktexrc`" file! Together with the enhanced abbreviation recognizer, I do now believe most abbreviations should be caught...

Finally, he has also given me valuable hints for improving the program's outputting routine, and given me lots of suggestions for filtering unnecessary/false warnings away.

Gerd Böhm
`Gerd.Boehm@physik.uni-regensburg.de`
Improved and bug-fixed the MS-DOS port of ChkTEX v1.4, sending me ready-to-yank code patches. The original port didn't respect all the peculiarities of the MS-DOS file-system, unfortunately.

Antonio DiCesare
`dicesare@vodafone.it`
He provided many feature requests and bug reports for the 1.7.1 version making it a much better release than it would have otherwise been. He also helped expand several keywords in the "`chktexrc`" file.

Mojca Miklavec
`mojca.miklavec.lists@gmail.com`
Found and helped debug a problem (fixed in 1.7.2) occurring only on some platforms, 32 bit Macs for one.

Baruch Even
`chktex@ev-en.org`
Maintainer of ChkTEX for about a decade.

Lars Frellesen
`frelle@math-tech.dk`
Sent a few bug reports regarding the filtering of messages. He has also helped me to expand the "`SILENT`" keyword in the "`chktexrc`" file.

Wolfgang Fritsch
`fritsch@hmi.de`
Author of the OS/2 port, which he did using the emx compiler. Please direct questions regarding strictly to that port to him (I would like to receive a carbon copy, though).

Stefan Gerberding
`stefan@inferenzsysteme.informatik.th-darmstadt.de`
First one to report the Enforcer hit in v1.2 when using ChkTEX as a pipe. Also came with suggestions to make ChkTEX more easily compile on early gcc compilers.

He has also kept on beta-testing later versions of ChkTEX, giving me bug-reports and enhancements requests.

Kasper B. Graversen
`kbg2001@internet.dk`
Lots of creative suggestions and improvements. Several of the warnings implemented were based on his ideas. In addition, he has given advice for improving the existing warnings.

Has also provided some OS-oriented code.

Frank Luithle
`f_luithle@outside.sb.sub.de`
Wrote a translation for v1.0. Unfortunately, he remained unreachable after that. . . :-/

Nat
`nat@nataa.frmug.fr.net`
Reported the same bug as Gerberding. In addition, he taught me a few tricks regarding the use of gcc + made me understand that the ANSI standard isn't unambigious; at least the `getenv()` call seem to be open for interpretations. Many possible incompatibilities have been removed due to these lessons.

Michael Sanders
`sanders@umich.edu`
Has found some of the bugs in this beast; both obscure and long-lived. Has also provided motivation to clarify this document.

Bjørn Ove Thue
`bjort@ifi.uio.no`
Author of the MSDOS port; please direct questions regarding strictly to that port to him (I would like to receive a carbon copy, though).

Martin Ward
`Martin.Ward@durham.ac.uk`
Sent a few bug-reports; also gave me information upon where to find regexp code. He also provided a Perl script for checking ordinary text, which ideas I was able to implement in ChkTEX. In addition, he sent me the source code for `lacheck`; which also inspired some of the warnings.

12 Contacting the author

If you wish to contact me for any reason or would like to participate in the development of ChkTEX, please write to:

Jens Berger
Spektrumvn. 4
N-0666 Oslo
Norway
E-mail: <jensthi@ifi.uio.no>

Any signs of intelligent life are welcomed; that should exclude piracy.

Since the original author is unreachable, the maintainer these days is:

Ivan Andrus
E-mail: <darthandrus@gmail.com>

Have fun.