



A short manual for T_EXworks

Alain Delmotte, Stefan Löffler, and others

lowering the entry barrier to the T_EX world

Copyright © 2010–2021 Alain Delmotte, Stefan Löffler, and contributors. Some rights reserved.

This manual is free documentation: you can redistribute it and/or modify it under the terms of either (i) the CC-BY-SA license as published by Creative Commons (either version 3 of the License, or (at your option) any later version) or (ii) the GNU General Public License as published by the Free Software Foundation (either version 2 of the License, or (at your option) any later version) or (iii) both in parallel.

This document is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

Details of the licenses are available at <http://creativecommons.org/licenses/by-sa/3.0/> and <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>. The sources used to create this document are available at <https://github.com/TeXworks/manual>.

Contents

Contents	i
1 Introduction	1
1.1 Icons and style	3
2 Installation	5
2.1 Under Windows	5
2.2 Under Linux	6
2.3 Under macOS	7
2.4 Ready!	7
3 First steps	8
3.1 Interface summary	8
3.2 Creating a document	9
3.2.1 Writing the document	9
3.2.2 Typesetting the document and viewing it	10
3.2.3 The work of \LaTeX	11
3.3 And when errors occur?	12
3.4 Changing \TeX works parameters for convenience	15
4 Going further: Editing tools	16
4.1 Creating a document from a template	16
4.2 Creating a project using several source files	16
4.3 Spell-checking	17

4.4	Search and replace	18
4.4.1	Standard functions	18
4.4.2	Advanced search and replace (regular expressions) . . .	19
4.5	Other tools for editing and error tracking	21
4.5.1	Standard tools	21
4.5.2	Commenting	21
4.5.3	Matching delimiters	22
4.5.4	Smart quotes	22
4.6	Auto-completion	22
5	Going further: Other tools	25
5.1	SyncTeX'ing between source and preview	25
5.2	Special comment strings	25
5.3	Formatting the source for legibility	27
5.4	Showing the tags	28
5.5	Organising the windows	28
5.6	Cleaning the working folder	28
5.7	Portable Usage & Changing the Configuration	29
6	Advanced use: Scripting	31
6.1	Introduction to Scripting	31
6.2	Installing Scripts	32
6.3	Using Scripts	33
7	Beyond this manual	34
A	Customizing T_EXworks	36
A.1	Syntax highlighting	36
A.2	Keyboard shortcuts	38
A.2.1	Predefined shortcuts	39
A.2.2	Actions listed alphabetically	41
A.2.3	Actions listed by menu	42
A.2.4	Actions for typesetting tools	44
A.2.5	Actions for scripts	44
A.3	Roots for completion	45
B	Regular expressions	57
B.1	Introduction	57
B.2	Codes to represent special sets	58
B.3	Repetition	59
B.4	Alternatives and assertions	60

Contents	iii
<hr/>	
B.5 Final notes	61
C Compiling T_EXworks	62
Bibliography	64
Index	65

Introduction

Donald E. Knuth decided to create a new typesetting system, which would be called \TeX , because there had been a change in the printing system used for the volumes of his book *The Art of Computer Programming* and Knuth found the result of the new system awful.

The goal of \TeX was then to have a system which would always produce the same documents independently of the actual machine they were processed on. Knuth also designed the *Computer Modern* family of typefaces and the METAFONT language for font description.

The work initiated in 1977 was finished (the languages were “frozen”) in 1989. \TeX and METAFONT are not evolving any more except for minor bug fixes (\TeX versions are numbered following the decimals of π —now 3.1415926—and METAFONT the decimals of the number “e”—now 2.718281).

\TeX provides basic tools (commands/instructions/“primitives”) to define typesetting; almost every detail has to be defined, but the language allows the creation of macros for repeatedly used constructs. So collections of macros are loaded through format files (i.e., pre-compiled large macro collections).

Knuth created an original default format (600 commands, more or less) which is called *Plain \TeX* . This facilitates creating documents.

The most widely used format is \LaTeX (Leslie Lamport, 1985), which provides more global commands and structures for documents (article, book, . . .) allowing easier and faster work, but sometimes with loss of flexibility due to the more or less rigid framework. But there are many other formats and \TeX -variants in use as well, such as $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\TeX}$, $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\LaTeX}$, Con \TeX t, or $\text{\X}\text{\TeX}$, each having specific goals and advantages (and drawbacks).

To extend the format, one loads “packages” which are collections of macros specific to some aspect of typesetting.

From its specification in the late 1970s, the \TeX family had to evolve

until now, last version March 2008, to take into account the developments in the typesetting world outside T_EX.

Some of the problems to answer were/are:

- taking into account other languages with “alphabets” larger than the ASCII¹ one or with non-Latin characters altogether,
- having more fonts, there is not much variety in the fonts created with METAFONT (few font creators use it),
- creating documents in other formats than the normal DVI²,
- using the rich possibilities of other typesetting systems and formats like PostScript and PDF,
- having more calculation and scripting facilities,...

To answer these questions and others, many “engines” and programmes have been created around T_EX, including pdftex, pdf_latex, dvips, ps2pdf, and METAPOST, which opens the T_EX world to the possibilities of PostScript and PDF. X_qT_EX and X_qL^AT_EX to be able to use the “normal” fonts found on the different machines and to be able to cope with writing systems different from the left to right systems which originated in Europe (Latin and Cyrillic letters and associates)—right to left, vertically, pictograms,... Or LuaT_EX and LuaLaT_EX to have a powerful scripting language.

To use T_EX and the systems of its family, one has to create a “source” document as T_EX is only a system to “transform” a source document into a (beautifully) typeset document. This source is a simple text with typesetting instructions and one needs a programme to create it: the editor.

There are many editors able to create a T_EX source; some are general editors, others are specifically designed for T_EX: here T_EXworks comes in.

T_EXworks is a project to create a text editor for use with the T_EX family of tools; we will refer to these as (L^A)T_EX. Instead of creating a new sophisticated program, equipped with multiple tool-bars to meet any need, T_EXworks provides a simple editor, offering at first sight only a limited set of tools for text editing as well as a single button and a menu to typeset a (L^A)T_EX text.

¹“American Standard Code for Information Interchange”: a character encoding scheme including only Latin characters found in English, some common punctuation characters, and a few other symbols such as % or \$

²“Device Independent”: format of files produced by T_EX













The idea to create the editor came to *Jonathan Kew*, the initiator and leader of the project, after a long period of reflection on the reasons why potential users tend to keep away from (L)T_EX, as well as pondering the success of the **T_EXShop** editor on the Mac.




Finally the goal was also to provide the same editor on many operating systems: T_EXworks currently runs on Linux, macOS and Windows. The interface is always the same and the program offers the same functionality on all three platforms.

After this introduction, the second section of this manual explains how to install the software. In the third section, we describe the interface and create a first document as well as show the basics of T_EXworks. In the forth and fifth section, the advanced tools provided by T_EXworks are presented; you should read these sections only after mastering the basic working of T_EXworks. These advanced tools allow much more effective working practices. The sixth section gives a brief introduction to scripting. This section focuses on using ready-made scripts, not on writing your own scripts (which is beyond the scope of this manual and will be presented elsewhere). After that, the seventh section in which some pointers to further information about T_EXworks and sources for help are compiled concludes the main part.

Finally, the appendices provide additional information how T_EXworks can be customized, about the regular expression search/replace system, and how T_EXworks can be compiled from source. A short bibliography and an index conclude this manual.

1.1 Icons and style

Because a picture is often worth a thousand words, icons and special styling is used throughout this manual to avoid cumbersome paraphrases or mark specialties. Keyboard keys are usually depicted as , with the exception of a few special keys. These are: , , ,  (return), , , , ,  (space),  (backspace), and  (tab).

In addition, mouse clicks are depicted as  (left click) and  (right click; on macOS with a one-button mouse, this is usually available by holding down  while clicking).

Apart from input instructions, several passages throughout this manual are marked by special styling.

Information that is only valid or relevant for a particular operating system is marked like this:

Win

This only concerns you if you use Windows.

Of course you can also read it if you use another operating system.

It just will not be of much use to you.

Code examples are set in a fixed-space, typewriter font, with lines above and below to set it apart from the rest of the text:

```
Hello \TeX-World!
```

Closely related to this, chapter 3 contains several tutorials, which are typeset just like the code examples above, but with an additional notebook icon next to it.

Installation

\TeX works is only a text editor; to be able to create documents with $(\text{\LaTeX})\text{\TeX}$ and to typeset them to PDF, we also need what is called a *\TeX distribution*. This is a collection of programs and other files which will be automatically called by \TeX works during its work. Thus, you need to install a distribution: we will do that *before* starting \TeX works for the first time, as this way, \TeX works will automatically find what it needs.

TeX Live (<http://www.tug.org/texlive/>), a combination of \TeX , Mac \TeX and XEm \TeX , is available for all three operating systems (Linux, macOS, Windows).

*nix

For Linux: most Linux distributions include a \TeX distribution, but it may not be installed by default and you will have to use the Linux package management tools to do that. Alternatively, you can also download and install TeX Live directly from <http://www.tug.org/texlive/>.

Mac

For the Mac: **MacTeX**, a distribution based on \TeX and Xe \TeX , is available; see <http://www.tug.org/mactex/>.

Win

For Windows: a very popular distribution is **MiKTeX** (<http://www.miktex.org/>). MikTeX has an update programme, which has also been ported to Linux.

For details on portable usage, and changing local configuration locations, please see the section Portable Usage & Changing the Configuration – section 5.7 (on page 29).

2.1 Under Windows

Most of the larger \TeX distributions already contain \TeX works as a package. Sometimes, these versions even have some distribution-specific enhancements.

So, the preferred way of installing T_EXworks on Windows is to use the package manager of your distribution. In this case, you can skip the next few paragraphs. Be sure to read the end of this section, though, as it provides important information about customizing T_EXworks to your needs.

If you want to obtain an “official” version, obtain T_EXworks by downloading the setup from the T_EXworks web site <http://tug.org/texworks/> after the installation of the T_EX distribution.

Simply install T_EXworks by running the setup file. During the installation, you will be asked where to install the program, if you want to create shortcuts, and if you want to always open `.tex` files with T_EXworks. There are reasonable default values that should work well for most users.

If you want full control over how and where T_EXworks is put, you can also download the `.zip` archive from the website and unpack it wherever you like. Note that in this case, shortcuts and file associations must be created manually.

When you start T_EXworks for the first time, it creates a folder named `C:\Users\<your name>\AppData\Roaming\TUG\TeXworks`. This folder will contain some sub-folders for auto-completion, configuration, dictionaries, templates, and interface translation/localisation files—we will see these in more detail later.¹

NB: At the time of writing, if `<your name>` contains any non-ASCII characters (for example accented characters), some functions of T_EXworks may not work correctly. For example, the spell-checker and forward/reverse synchronization between the source and `.pdf` will be impaired.

2.2 Under Linux

Several common Linux distributions already have packages for T_EXworks. They are adequate for most users and facilitate installing T_EXworks considerably.

If your Linux distribution does not provide recent, adequate packages, you need to build T_EXworks from source yourself, which is fairly easy on Linux. After the installation of the T_EX distribution, go to <https://github.com/TeXworks/texworks/wiki/Building> and follow the instructions suitable for your Linux distributions. Also see section C.

Once the program is installed, start T_EXworks. The folders `.local/share/TUG/TeXworks` and `.config/TUG` will be created in your home directory.

¹T_EXworks will save its preferences in the registry: `\HKEY_CURRENT_USER\Software\TUG\TeXworks`. If this is erased, it will be recreated with default values at the next use.

2.3 Under macOS

If you want to obtain an “official” version, obtain T_EXworks by downloading the archive from the T_EXworks web site <http://tug.org/texworks/> after the installation of the T_EX distribution.

It comes as a standalone `.app` package that does not require any Qt files installed into `/Library/Frameworks`, or other libraries into `/usr/local/lib`. Just copy the `.app` anywhere you like and run it.

On macOS, the T_EXworks resource folder will be created in your `Library` folder (`~/Library/Application Support/TUG/TeXworks`), inside your home directory. Preferences are stored in `~/Library/Preferences/org.tug.TeXworks.plist` which you can delete if you ever suspect it is causing problems.

2.4 Ready!

Finally, some files may need to be added to the “personal” files that T_EXworks creates. As the exact location of these depends on your platform, this will be referred to as `<resources>` or the **T_EXworks resource folder** throughout this manual. On Windows, this is `C:\Users\<your name>\AppData\Roaming\TUG\TeXworks`, on Linux it is `.local/share/TUG/TeXworks`, and on macOS it is `~/Library/Application Support/TUG/TeXworks` by default. The easiest way to locate this folder in recent versions of T_EXworks is to use the ***Help→Settings and Resources...*** menu item. It opens a dialog which shows you where T_EXworks saves its settings and where it looks for resources.

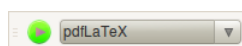
After installation and first run, have a look in the sub-folders of the T_EXworks resource folder and delete any `qt_temp.xxxx` files; they are temporary files left behind and could interfere with the normal ones, which are installed in the same folder, later on.

First steps

Let's now see how to create a first document: for this you'll need to type some text in the editor window of T_EXworks. (L^A)T_EX is *not* WYSIWYG¹ software, so you'll have to type the text and the instructions for formatting it and you'll see the result only after “typesetting” the text. This looks a little bit dry, but one very quickly gets used to it and it is well worth the effort.

3.1 Interface summary

When one opens the editor, it shows a very sparse interface: a title bar, a menu bar, two small toolbars, a large typing zone (white) and, at the bottom, a status bar. We are in the *source/editor* window. If you have already typeset the document previously, the resulting .pdf will be shown on the right hand side in the *preview* window.

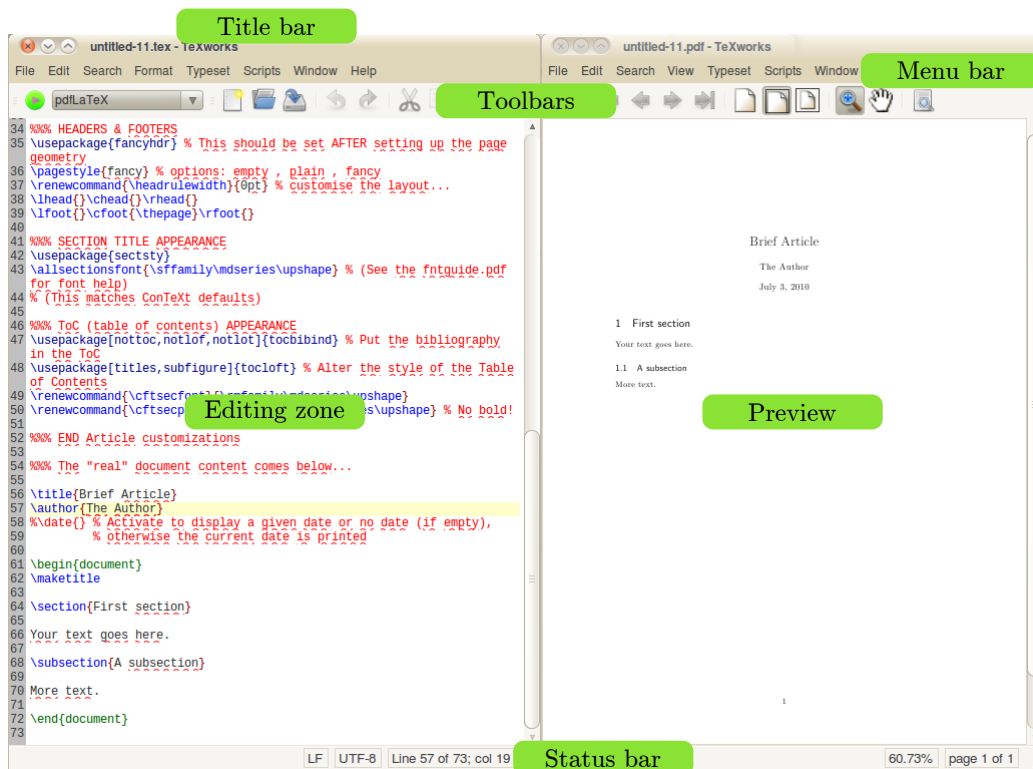


The first toolbar has a button to typeset and an drop-down menu to choose the format for typesetting (we'll choose pdfLaTeX). Knowing that the keyboard shortcut for typesetting is **Ctrl** **T** (macOS: **⌘** **T**) and that we almost never change the format, we could even hide this toolbar. The selection of a format for compiling can also be changed through the **Typeset** menu.



The second toolbar provides the standard functions: New document, Open, Save | Undo, Redo | Cut, Copy, Paste | Search, Replace.

¹ *What You See Is What You Get.*



Even though they are not looking like real buttons, the widgets in the status bar can be clicked. The widgets showing the current position (line or page, respectively), for example, open a dialog to enter a line or page to jump to when clicked. The other widgets typically open contextual menus where some settings can be changed.

3.2 Creating a document

3.2.1 Writing the document

As an example of the use of TeXworks, we will work with L^AT_EX, but any other T_EX system is possible. In particular, if you need to use some special fonts²—a mandatory font for an official template, non-Latin alphabets, etc.—the X_YL_AT_EX system is very powerful.³

²You can only use fonts L^AT_EX knows, most of which are coming in packages included in your distribution. You cannot use your “normal” fonts, unfortunately. For more information, see for example <http://faq.tug.org/> and <http://www.tug.dk/FontCatalogue/>.

³See the bibliography for pointers to X_YL_AT_EX and X_YL^AT_EX.

Let's create our first document now. Enter the following text exactly as shown. To show some of the features of T_EXworks/L^AT_EX, it is in French intentionally.



```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{geometry}
\geometry{a4paper}

\usepackage[frenchb]{babel}

\title{Premier document}
\author{Un TeXnicien}
\date{}






\begin{document}
\maketitle

Voici un texte accentué en français!

\end{document}
```

Save the file in a folder for test documents (e.g., <home>\TeXworks_tests); call the file `first.tex`. Note that it should have a `.tex` extension.

3.2.2 Typesetting the document and viewing it





Next we start typesetting⁴ by clicking the green button  or by   (macOS:  ).

A new panel opens between the typing area and the status bar: the *output panel*, labeled *Log*; everything L^AT_EX is doing is displayed there⁵. When L^AT_EX finishes this panel disappears (if there no error occurred) and a new window will appear; in this new window, the *Preview window*, you can see a page with a title “Premier document” followed by the name of the author “Un TeXnicien”, both centred, the text “Voici un texte accentué en français!”, and a page number at the bottom centre.

⁴We also use the words “compilation” and “to compile” for the same action; indeed L^AT_EX works on the source file to produce a `.pdf` output, so there is a compilation.

⁵see page 12 for a picture of the panel

Notice that the mouse cursor is like a magnifier in the new window. If you press (and hold) the left button of the mouse you can see the text under the magnifier much bigger (it is a magnifier, isn't it!); you can move the magnifier and so inspect the text in detail.

To go back to the source, you can just click in its window or better use   (macOS  ). This shortcut toggles between the two windows. See also section 5.1 to automatically move to a specific location in the output from the source or vice versa.

3.2.3 The work of L^AT_EX

Let's shortly analyse the result to understand what L^AT_EX did and why, now. Introductions and full tutorials can be found on the internet: see for example *lshort*⁶ which should be installed as part of your T_EX distribution, and is also available from CTAN.⁷

First, we ask to create a document of the *article* class: this defines the global layout of the document.

Next, we say that the input document (the source) is saved with the Unicode encoding *utf-8* and that it may contain characters which are not present in the standard ASCII without accents. We also want to use an output encoding T1 (the modern T_EX encoding); we also want an A4 document and not the default *US letter* size. Finally, we make it clear that the typography should follow the French rules using the **babel** package.⁸ Those general instructions for the work are done by packages called with options.

Lastly, we finish the declaration part of the document, the *preamble*, giving the title, the author, and the date of the document; here we specify no date.

Next comes the body of the document, which describes the actual content, between the lines `\begin{document}` and `\end{document}` (these are L^AT_EX commands).

Let's do some experiments to show the effect of these instructions. For this, we put a % in front of the instructions; the % and everything after it will be considered as comment, which will be ignored by L^AT_EX.⁹


Comment out the line `\usepackage[utf8]{inputenc}`, and typeset the

⁶ *The (Not So) Short Introduction to L^AT_EX 2_ε*

⁷ *Comprehensive TeX Archives Network*, a network of mirror servers of the central CTAN; there, one can find almost everything about T_EX, L^AT_EX, and more: <http://www.ctan.org>

⁸ This influences, e.g., automatic hyphenation of words or the way punctuation characters are typeset

⁹ Notice that the comments are, by default (this can be changed), coloured red by T_EXworks, so we see them well.

file. You should see that the accented characters are now displayed incorrectly in the preview window. If, in addition, you also comment out the line `\usepackage[frenchb]{babel}`, L^AT_EX will give an error. Just hit  to continue the typesetting.

After these experiments, let's modify the text as follows:



```
\begin{document}
\maketitle
\tableofcontents
```

```
\section{Petite démonstration}
```

Voici un texte accentué en français!

Suite du texte entré après avoir fait un retour chariot. Dans l'éditeur on peut demander un passage automatique à la ligne du texte saisi; mais le numéro de ligne n'est incrémenté que par un retour chariot.

Nouvelle ligne en passant une ligne dans la source: c'est la manière d'indiquer un changement de paragraphe.

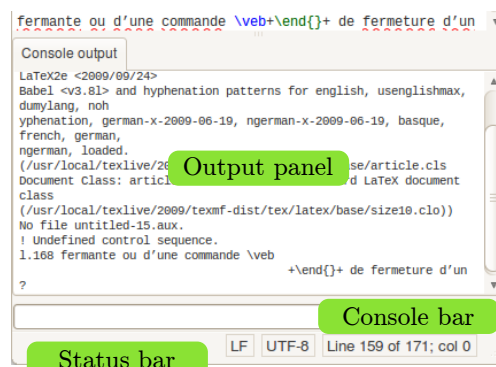
```
\end{document}
```

Redo the previous experiments and observe the changes which appear.

Note that entering only one carriage return doesn't create a new paragraph. In L^AT_EX, one has to have an empty line for that. In T_EXworks, the line number of the source (on the right in the status bar) numbers the lines created with carriage return, not the wrapped lines.

3.3 And when errors occur?


When you create a document for typesetting with L^AT_EX, you cannot avoid making mistakes: forgetting a closing brace or an `\end{}` to close an environment, using mathematical commands without switching to mathematical mode, etc. When you compile and there is an error, L^AT_EX stops, giving you a chance to deal with the problem. This is shown by the stopping of the scrolling action in the output panel, and an error message being displayed, with L^AT_EX waiting for an instruction to know what it should do.







One sees the *typing cursor* in the line between the output panel and the status bar: the *console bar*.

The error message is on many lines, for example like this:


```
! Undefined control sequence.
1.168 ... fermeante ou d'une commande \veb
                                         +\end{ }+ de fermeture d'un...
?
```


L^AT_EX says that it doesn't recognize the command name, sometimes suggests to see the manual or to type **h** (plus ) for help, points to the line number where it noticed the error¹⁰ (here 168), and shows the place of the error at the cut of the line (here at `\veb`). Finally, it shows that it waits for an action from us by displaying a single question mark.

There are different possible actions:

- Type **h**  and ask to continue as if nothing happened; sometimes this allows to finish compiling, but there will be an error in the result.
- Type **h**  to ask for help; this help is not always clearer than the error message, but often gives a clue.
- Type **i**  to tell L^AT_EX that we will propose a replacement text. Enter the text followed by ; it will be used, beginning at the start of the error, but you should correct the source afterwards as L^AT_EX never changes that.

¹⁰Unfortunately, this does not always have to be the place where the actual mistake was made in the sources. This is discussed later.

- Type `x`  to stop compilation. This is the traditional (L^A)T_EX way to kill a typesetting process.

We can also kill the typesetting by repeating the action used to start it: the green typesetting button will have changed to a red one with a white cross . By clicking on that button or by hitting `Ctrl` `T` (macOS: `⌘` `T`) again, the L^AT_EX process is terminated. The output panel is still visible and so one can still see the error message.

You should note that sometimes an error appears far from its actual position. For example, when opening an environment but not closing it, L^AT_EX doesn't see the error before it encounters another end of environment without closing of the first one. The error is often only picked up at the `\end{document}` command, which shows that another environment was not closed!

Sometimes, an error still occurs during subsequent runs of (L^A)T_EX even after it was corrected in the document. This can happen because (L^A)T_EX creates a number of intermediary files which can still contain the original, erroneous code. Therefore, it is advisable to remove those files after fixing an error. T_EXworks provides a command to facilitate this—see section 5.6.

After an error occurred, the output panel remains open—even after consecutive typesetting runs—to help you fix the problem (which can sometimes take several attempts and restarts of (L^A)T_EX). Once all problems are solved, you can close the panel by the **Windows**→**Hide Output Panel** menu item. As an alternative, you can also configure to output panel to hide automatically as soon as the typesetting process finishes successfully by setting **Edit**→**Preferences...**→**Typesetting**→**Hide output panel** to *On success*. Since one can easily overlook other problems in the document (e.g., undefined references) that do not cause (L^A)T_EX to fail, this option is only recommended to expert users.

To help you in finding and fixing error (at least if you are using L^AT_EX), T_EXworks comes with a small script that extracts all errors, warnings, and other noteworthy messages from the console output and presents them to you in a simple, tabular form.



Here, you see a list of L^AT_EX messages, color-coded and sorted by severity. Red represents errors, yellow indicates warnings, and blue stands for over- and underfull box warnings. Next to the colour bar, you see the name of the file in which the error was detected. Next to that, you see the line number (if the script was able to determine that), as well as an excerpt from the console output telling you what the error was. Moreover, the filename is a link which will take you to the file (and, if a line number could be determined, also the line) where L^AT_EX reported the error. Hopefully, you can quickly fix any errors that may occur this way.

In case you are not using L^AT_EX, this script may be of little use, particularly if the console output is formatted differently. In this case, you can simply disable the error parsing hook script (see section 6.2).

3.4 Changing T_EXworks parameters for convenience

If the default font of the editor doesn't suit you, it is possible to change it from **Format**→**Font...** by selecting a new one in the dialogue box which appears. This change will apply only to the current window, and until T_EXworks is restarted.

From the **Typeset** menu or from the drop-down on the **Typesetting tool bar**, you can change the compilation format. Again this change will only be temporary and for the current document.







To have permanent changes, you need to change the *preferences* through the **Edit**→**Preferences...** menu item, using the **Editor** tab for the font and the **Typesetting** tab for the compilation format: the default format is at the bottom of the tab.

Going further: Editing tools

When you have had some practise with T_EXworks, you'll find the need for more effective tools. Many of them are bundled with T_EXworks. We are going to see some of them now.

4.1 Creating a document from a template

Most documents you will create will use the same instructions in the preamble, the same layout settings, similar heading and so on. You can use predefined templates to get started quickly or create your own with all of these settings already in place.

Use **File**→**New from template...** or    (macOS:   ). A dialogue box opens to allow you to select one of the templates. After selecting one and pressing OK, a document is created and you can start to work.

If you want to create a personal template, you just have to create a suitable document with everything you always want to do (and perhaps marking places to fill in the rest) and save it as a `.tex` file in the `<resources>\templates` folder, or a sub-folder of it, if you wish.

4.2 Creating a project using several source files

When the source becomes long, it is sometimes difficult to navigate and maintain it. In that case, it is useful to split the source into different smaller files: one file will be the main document, with the preamble and the `document`

environment, as well as calls to the “sub-documents”¹, which could in turn contain separate chapters, for example.

But there might be a problem if you want to start typesetting/compilation in a sub-document: as there is neither a preamble nor a `document` environment there, L^AT_EX will stop immediately with an error.

To tell T_EXworks that it should typeset the main document, one adds at the very beginning of the sub-document the instruction:

```
% !TeX root = path/main_file.tex
```

for example:

```
% !TeX root = manual.tex
```

If the main file is in the same folder, its name is enough, as in the above example. Otherwise, you must also give the path to the main document (preferably relative to the sub-document in question, e.g., `../manual.tex`). Notice that the slash `/` and not the backslash `\` should be used as directory separator even on Windows.

Further, with MiKTeX, the call to a sub-document `\input{name.tex}` should include the extension `.tex` to ensure proper SyncTeX functionality (see section 5.1).

4.3 Spell-checking

You can turn on automatic spell-checking of your source document from **Edit** → **Spelling** → **<language>**. It is also possible to ask T_EXworks to enable spell-checking by default by setting a dictionary in **Edit** → **Preferences...** → **Editor** → **Spell-check language**.

During typing, every word the spell-checker considers wrong is underlined by a red wavy line. A right-click on the word opens a contextual menu in which there are some replacement suggestions. Click on the desired word to make the replacement.

Before using the spell-checker, you need to install dictionaries in the right folder of T_EXworks: `<resources>\dictionaries`. The `<resources>` folder can be accessed easily via **Help** → **Settings and Resources...**

¹Called by the commands `\input{}` or `\include{}`, see L^AT_EX manuals for more information.

*nix

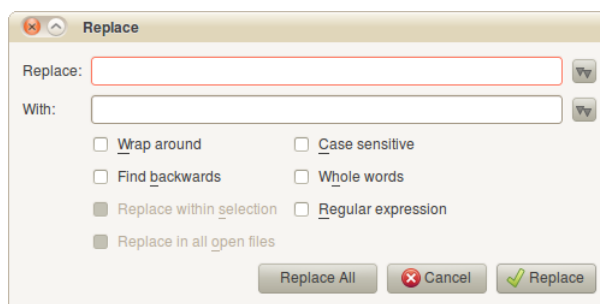
On Linux, the dictionaries are usually taken from the folder `/usr/share/myspell/dicts`—the default path for myspell dictionaries. Note, though, that the maintainer of your T_EXworks package may have changed this to reflect the file system layout of your Linux distribution. You can override this default by setting the `TW_DICPATH` environment variable before running T_EXworks.

One can use the available dictionaries for OpenOffice.org and other free software;² if you have Mozilla Thunderbird with spell-checking, you can copy its `.aff` and `.dic` files as well, for example.

4.4 Search and replace

4.4.1 Standard functions

The options of the menu *Search—Find...*, *Find again*, *Replace...*, *Replace again*, and *Go to Line...* (`Ctrl F`, `Ctrl G`, `Ctrl R`, `Ctrl Shift ↑ R`, and `Ctrl L`, respectively)—are standard actions (macOS: `⌘ F`, `⌘ G`, `⌘ R`, `⌘ Shift ↑ R`, and `⌘ L`); the first and the third open a dialogue box:



Here, the usual options are available: *Wrap around*, *Find backwards*, *Search/Replace within selection*, or *Find all occurrences*. The following options are also usual: *Case sensitive* and *Whole words*. By default, the search is forward, towards the end of the document.

The option *Search/Replace in all open files* is also a frequent choice, but not as much as the others; this allows, for example, replacement in all the files of a project—pay attention, though, as this is very powerful.

The last option, *Regular expression*, is detailed in the next sub-section.

In the *Search* menu there are other options:

²See, for example, <http://extensions.services.openoffice.org/dictionary>. The `.txt` files can be renamed to `.zip` and then uncompressed to find the required `.dic` and `.aff` files.

Copy to Find copies the currently selected text into the **Find** field of the Find dialogue or the **Replace** field of the Replace dialogue; you still need to open the dialogues separately

Copy to Replace copies the currently selected text into the **With** field of the **Replace** dialogue

Find Selection uses the current selection for a search without opening the **Find** dialogue—very fast

Show selection scrolls the view to the currently selected text—useful if word wrapping is turned off and you moved in the document using the vertical scroll bar on the right

4.4.2 Advanced search and replace (regular expressions)

The regular expressions provide a very powerful tool, but they require some effort to be well understood. To understand them fully would require a manual of its own³, so we'll only give some simple ideas of use. For more advanced uses, as well as lists of the most used codes, see section B.

Suppose we have the following text:



```
Voici du texte pour tester les expressions régulières dans du texte
accentué.
Voici du texte pour tester les expressions régulières dans du texte
accentué.
Voici du texte pour tester les expressions régulières. Voici du
texte pour tester les expressions régulières.
truc          truc
tél.: 010-99-99-99
tél.: 00.32.10.99.99.99
tél.: 00/32-10/99.99.99
```

We want to

1. insert an empty line after each “accentué” (to create paragraphs in L^AT_EX), but not after the three telephone numbers;
2. replace each *tab* character between the two words “truc” of the fourth paragraph by three spaces; and finally

³Such manuals exist on the internet.

3. make the telephone numbers consistent by replacing the various punctuation characters by spaces.

For 1., in the dialogue box **Replace** ( ) for *Replace:* we put `>\n<`⁴ and in *With:* `>\n\n<`. `>\n<` is the code to match or insert a line feed. You will need to select the first four paragraphs and the beginning of the fifth (the first telephone number) and to tick the *Replace within selection* and *Regular expression* options; if this was not done and an empty line has been inserted after each line, select the telephone lines and do the reverse action: replace `>\n\n<` by `>\n<`. So we replaced one line feed by two, creating an empty line.

For 2., use `>\t<` and `>\t\t\t<`⁵. `>\t<` is the code which represents a tab, while a space is typed in literally (here represented as `>\t<`).

For 3., find `>-\|\.|/ <` and replace with `>\t<`. Here, `>| <` provides alternatives (–, ., or /); for the dot we have used `>\. <` because the dot alone is a regular expression code which represents any character and we would have replaced all the characters by spaces! We therefore have to use a code—prefixing the dot with a backslash tells specifies that the normal meaning of the dot should be used instead of the special meaning it usually has in regular expressions.

If one has strings of the same character but of different lengths (for example 3, 4, or 5 times the same character e) and one wants to truncate all these strings to a string with less characters (for example 2), one can ask to replace the string `>e{3,5} <` by `>ee <`.

If one wants to insert the same string at the beginning of some paragraphs separated or not by an empty line, for example `>\noindent\t<` or `>\item\t<`, one can replace `>\n\n<` or `>\n<` by `>\n\n\\noindent\t<` or `>\n\\noindent\t<`. Pay attention, we have a double `\` in front of `noindent` to get one (`\noindent`) because `\` is an escape character in regular expressions (we’ve met it before in the expression `\.`)!





If it were making sense, we could replace all the letters between “a” and “m” by “\$” using `>[a-m] <` and `>$ <`.

⁴the `><` are used here only to show the limits of the entered text and they should not actually be entered.










⁵These are three space characters.

4.5 Other tools for editing and error tracking

4.5.1 Standard tools

It is always possible to undo an action using *Edit*→*Undo* or  (macOS: ): this way you can undo stepwise! The inverse action, redo, is available as *Edit*→*Redo* or  (macOS: ).⁶

TeXworks also provides the standard editing tools such as the clipboard; therefore one can select, cut/copy and paste a piece of text normally.







You can select with the mouse by dragging over the desired text, or by double-clicking to select a word. Using the keyboard, holding down  while moving using the arrow keys will select text. You can also move and select word by word moving left or right holding  down ( on macOS). The clipboard shortcuts are the ones you'll find in almost every program:  to cut,  to copy, and  to paste (,  and , respectively, on macOS).




You can easily change the case of a selection—put everything upper case or lower case—using *Edit*→*Change case* and next, depending on the desired effect, *ALL UPPERCASE*, *all lowercase*, or *Toggle Case* (which toggles the case of each letter individually).


It is also convenient to show the line numbers, as all error messages refer to these numbers; you can toggle the line numbers, on the left of the editing panel, from *Format*→*Line Numbers*.

4.5.2 Commenting





When preparing a document with (L^A)TeX, it is often useful to prevent compilation of a portion of text to be able to locate an error; you can do this piece by piece until you find the part which causes the error. For that, commenting the source block by block is needed.

We have seen that the symbol % marks the beginning of a comment. To comment a big piece of text, it is sufficient to select it and ask to mark it as comment *Format*→*Comment* or  (macOS:  ). To remove the comment, select the lines and choose *Format*→*Uncomment* or  (macOS:  ).⁷

⁶  and  work as well on Windows.  work as well on macOS.

⁷On some keyboards, like the French one, it is impossible to use  or

4.5.3 Matching delimiters

A frequent error is to forget a closing symbol: parenthesis, bracket, square bracket, *etc.* T_EXworks helps with a tool to show the pairs of symbols: when the cursor moves over one of these symbols, its partner is briefly highlighted. You can also select an entire block using **Edit**→**Balance Delimiters** or by the shortcut   (macOS:  ). Thus, you will immediately see the scope of the block.

4.5.4 Smart quotes

Another similar error, but this time semantic and not hindering typesetting, is in the use of quotes when one wants to give focus to some text.


There are two types of quotation marks in English: the ‘single’ quotes and the “double” quotes. They are formed by ‘ and ’; these are not the quotation marks used in programming and found on the keyboard: " and '. Using the T_EXworks smart quotes system, one can use the latter as normal to automatically produce the typographically correct single/double opening and close quotes.



In a .tex document, select one of the smart quotes system: **Format**→**Smart Quotes**→**TeX Ligatures**, →**TeX Commands**, →**Unicode Characters**. Then, when you want to start a quoted section in your text, let’s say enclosed in double quotes, type ", then the text to be quoted, and finish again by "; T_EXworks will automatically insert the correct opening quotes `` and later the correct closing ones '' . The three options give the same result in the typeset document, but **TeX Ligatures** should work best in most cases.


Finally, it is possible to define personal quotation marks systems (in the file `smart-quotes-modes.txt` in the `configuration` folder of the resource folder).

4.6 Auto-completion

Another tool which rapidly becomes indispensable is auto-completion. Indeed, when you use (L)T_EX, you have to continuously enter codes to, for example, create environments; you also have to remember to close every group you open.











Auto-completion allows you to type a keyword, hit the  key, and have T_EXworks insert the (L)T_EX command or environment code automatically.



   ; these shortcuts can be changed, however—see section A.2

As an example to insert “ \LaTeX ”, we have to type `\LaTeX`. This is not difficult, but entering “\” followed by the word “ \LaTeX ” with alternating capitals and lower case letters could become annoying after a while.⁸ With auto-completion, you just enter `latex` and hit  to get `\LaTeX`. You just have to take care that there is no *letter* directly preceding or succeeding `latex`—e.g., `al latex`—, or else the mechanism might not pick up the correct keyword.

Another example is `bmin`, which gives

```
\begin{minipage}{}  
•  
\end{minipage}•
```

with the cursor between the empty pair of curly brackets where you need to enter the size of the minipage. See the section A.3 for a list of the keywords for auto-completion. Notice the “•” in the minipage environment. They are placeholders which can be reached by   (  on the Mac), repeating this shortcut cycles forward through the placeholders; by    (  ), you can also cycle backwards.

If a partial keyword is given, repeatedly hitting  will cycle through possible completions. For example, `bali` (the `b` commonly indicates the beginning of an environment, `\begin{}`) creates the `align` environment after one , next `align*`, and after that, in succession, `alignat`, `alignat*`, `aligned`, `alignedat`, and `alignedat` with options; to access the last environments directly, they have their own codes which start by `bali` (`balis`, `baliat`, `baliats`, `balied`, `baliedat` and `baliedato`).

If you want to create your own keywords, you can add a `.txt` file in the `completion` folder inside the `resources` folder. The entries in the file should have the following format:

```
bfigo:=\begin{figure}[#INS#]#RET##RET#\end{figure}•  
\bibliography{#INS#}•
```

In the first case, `bfigo` is the assigned keyword (with `:=`) to be converted into a `figure` environment with an optional argument; there are two carriage returns (`#RET#`) after the `begin`, i.e., an empty line, and the cursor is placed between the square brackets (at the position of `#INS#`). “•” is a place holder as introduced before.


⁸In particular with keyboard layouts where \ is not directly accessible.

In the second case, we give ourselves a shortcut, which will let us type the first part of `\bibliography{}` and have T_EXworks convert it to the full name plus braces (with the cursor between them). In this case, the keyword is the instruction itself.

Note that the `.txt` file containing the auto-completion information needs to be UTF-8 encoded—this is the default encoding for all files created with T_EXworks.

Going further: Other tools

5.1 SyncTeX'ing between source and preview

When you are reading a document in the preview window and see something to change, it is convenient to go immediately to the corresponding place in the source. To do so, hold down `Ctrl` (macOS: `⌘`) and click at the appropriate place in the preview; the cursor will move and highlight the corresponding location in the source window. The same is true in the other direction: `Ctrl`  in the source will highlight the same line in the preview window.¹


Win

Here a remark for users under Windows: this only works if **all** the names for folders/files/...do **not** have accented characters. If, for example, your document is in `C:\Documents and Settings\Propriétaire\My Documents\thesis` it will not work because of the `é` in `Propriétaire`!

5.2 Special comment strings

Special comments, at the very beginning of the files, can be used to manage two other aspects of the compilation.

By default, `TEXworks` uses the “utf-8” encoding for loading and saving files, but some files could be saved in another encoding. Common ones are “latin1”, which is the dominant encoding on Windows when using western languages, and “Apple Roman” which is dominant on macOS.

¹It is also possible to use a right-click () to open a context window and select “jump to PDF” or “jump to source”.

To set a different encoding for a specific file one can put the following at the beginning of that file:

```
% !TeX encoding = latin1
```

Note that without this line, you must tell T_EXworks the correct encoding manually. Otherwise, your data could be corrupted! To override T_EXworks's default choice of encoding, use the menu that appears when you click on the editor window's middle status bar widget.

If you opened a file in T_EXworks that was not saved as utf-8 but is lacking the `% !TeX encoding` line, it might be displayed with (some) weird characters. In that case, you can specify the correct encoding via the status bar widget menu in the same way, but it is *imperative* that you then use ***Reload using selected encoding*** from the same menu! This forces T_EXworks to open the document again with the encoding you selected, the weird characters should be replaced by normal ones, and only then it is safe to continue to work normally. To avoid having to repeat this procedure each time you open this file, you should either switch to utf-8 for saving it in the future or add a proper `% !TeX encoding` line.

If we want to compile a file with another programme than the default T_EX or L^AT_EX, we put at the beginning of the file:

```
% !TeX program = <the_programme>
```

for example:

```
% !TeX program = xelatex
```

Pay attention to this last instruction. You have to use the name of the programme here which should be used for the whole project, as the first encountered programme when starting typesetting is used (which is the one from the sub-document you are in). T_EXworks will use that programme, even if another name appears in the main document!

When opening a document which contains a `% !TeX program` line, the specified programme will become the one to use and its name will appear in the drop down menu in the toolbar; you can, however, override this by selecting a different one from the drop down list, if you want.

In addition, you can set the spell checking language by a similar comment line:

```
% !TeX spellcheck = <language_code>
```





The language codes available on your system are listed in parentheses in **Edit** → **Spelling** next to the human-readable name of the language.





5.3 Formatting the source for legibility

To facilitate legibility of the source, one can use indentation as programmers do:

```
\begin{itemize}
  \item First element of the list;
  \item second element;
  \item last element:
  \begin{itemize}      % beginning of a sub-list
    \item first sub-element;
    \item second sub-element.
  \end{itemize}
\end{itemize}
```

This increases legibility, but works well only on short lines, without text wrapping; or if one chooses not to use text wrapping by unchecking **Format** → **Wrap lines**.

The command **Format** → **Indent** or the shortcut   (macOS:  ) will indent the line, or the selected lines, by inserting a tab character. You can repeat the process to increase the indent.

To remove one level of indentation, use **Format** → **Unindent** or the shortcut   (  on macOS).²

As *indent* only indents the first part of very long (wrapped) lines, this is not very satisfactory in some cases. But one can ask T_EXworks to split a long line (longer than the width of the editing window) into short ones adding a hard coded line feed. **Format** → **Hard Wrap...** opens a dialog box in which you can specify the width of the lines; you can also re-format lines which have already been split.

²See the modified shortcuts if your keyboard layout does not allow these actions.

5.4 Showing the tags

When a document is becoming long and you want to move to a specific place (a chapter, a section, a subsection, ...) you normally need to scroll the editing window to find the desired location, or use the Find dialog if you remember a keyword in the chapter's title.

To the same end, though a lot more comfortable, you can also use the structural information in the document to navigate the source: the menu item **Window**→**Show**→**Tags** opens a panel showing the information detected by T_EXworks. Clicking on an item in the panel moves the cursor to the corresponding part in the source. That panel, like any other, can be resized by dragging its border.

The same action is possible in the PDF window from **Window**→**Show**→**Table of contents**, but this only works if one has created structure tags in the PDF file using the package `hyperref`.

5.5 Organising the windows

By default, the editor/source window opens on the left and the preview one on the right (when the corresponding PDF file exists), thus splitting the screen in two.

You can change the position of the windows in the **Window** menu. →**Stack** and →**Side by side** give the default effect if there is only one document open. If not, →**Stack** creates a mosaic with all the windows. The other options allow to place the windows for your convenience. It is also always possible to resize and move the windows manually, of course.

For the preview you can change the way it is presented and of course the zoom by **View**→**Actual size**, →**Fit to width** and →**Fit to window**; you can also zoom in and out. Shortcuts exist for all these actions and are shown next to the menu items.

5.6 Cleaning the working folder

Very soon when one uses (L^A)T_EX, one discovers that the working folder is cluttered by many files which have the name of the source file but different extensions: `.aux`, `.log`, `.toc`, `.lof`, `.lot`, `.bbl`, ...

All these are files needed by (L^A)T_EX to be able to create the table of contents, lists of figures/tables, the bibliography, the cross references and, also very importantly, to keep track of what it did (in the `.log` file).

Apart from the external files, images, pictures, . . . , the only files required are the `.tex` files, the sources of the document. One can erase all the others. Sometimes, this is even necessary when (L^A)T_EX gets stuck after an error.

This can be done using a T_EXworks command from the **File** menu with the **→Remove Aux files...** item.

When you use this command, a dialog box opens in which you can check/uncheck the files you want to remove.³ The dialog box will only list files that actually exist in the folder; if you removed all these auxiliary files before, you get a message box saying that there is no file to remove at the moment.

The list of auxiliary files which are taken into account is defined in the file `texworks-config.txt` in the `configuration` folder of the T_EXworks resources folder. You could add some if required.

5.7 Portable Usage & Changing the Configuration

We have seen in section 2 (on page 5) that the first time you use T_EXworks, it creates a resource folder and also that it saves default preferences.

It is possible to define a personal place where one wants the resource folder and the preferences. This can be handy when one wants a portable system (e.g., on an USB stick) or when one wants to easily access the templates or completion folders for modifications.

For this, create a file `texworks-setup.ini` in the programme folder in which you specify the path to the folder containing the completion, configuration, dictionaries, . . . folders and the configuration file (`texworks.ini`); there will be two lines:

```
inipath=C:/myfolder/TW_conf/  
libpath=C:/myfolder/TW_conf/
```

`inipath` for the configuration file and `libpath` for the necessary folders. Here, `TW_conf` would replace the resource folder `TeXworks`. Note that the referenced folder (here `TW_conf`) should exist—it will not be created—and that the `/` is used even on Windows (instead of the common `\`).

If one wants to put the resource folder in the programme folder as a subfolder, one can use an instruction like `inipath=./TW_conf/`; all relative

³The name of the main file is used to list the possible candidates for deletion.

paths are taken to be relative to the T_EXworks programme folder (on macOS, the folder containing the app package is used).

One can also add a line like

```
defaultbinpaths=C:/Program Files/MiKTeX 2.7/miktex/bin
```

to specify where the programmes of the T_EX distribution are located; but this instruction is not yet completely operational, especially under Windows.

Advanced use: Scripting

6.1 Introduction to Scripting

All the functions and utilities described so far were built into T_EXworks by default. While some of them could be configured or customized to a certain extent, they are intended to suit the most common needs of a general audience. However, the T_EX world is very large and diverse. In order to enable users to address their special needs—from simply making some text bold to fulfilling special requirements for the next book or scientific paper you want to publish—the core functionality of T_EXworks can be extended or modified by the use of scripts.

Scripts are simple text files that you can open, read, or modify in any text editor (including T_EXworks, of course). They are written in a specific scripting language that is essentially a programming language. At the time of writing, T_EXworks supports QtScript¹ (built-in), Lua (with a plugin), and Python (with a plugin). To see which scripting languages are available on your system, use the *Scripts*→*Scripting T_EXworks*→*About Scripts...* menu item.

Writing scripts is beyond the scope of this manual, but is documented elsewhere². Here, only the installation and usage of scripts will be discussed.

T_EXworks distinguishes between two types of scripts: standalone scripts and hook scripts. The primary purpose of standalone scripts is to add new functionality to the program. If you need a new function, such as a command to make the selected text bold, a standalone script is the one to choose. These scripts get an item in the *Scripts* menu, and you can run them simply by

¹A scripting language similar to JavaScript provided by Qt.

²See, for example, Paul Norman's page <http://twscript.paulanorman.com/docs/index.html>.

clicking on that menu item (or by using a keyboard shortcut, if the script provides one).

Hook scripts, on the other hand, are meant to extend existing \TeX works functions. They are hooked into the code at specific places, e.g., after the typeset process has finished or after a file was loaded, and can add or modify whatever \TeX works is doing. One example for this would be a script that analyses a newly loaded file and sets the spell-checking language based on `babel` commands found in the document. Thus, hook scripts do not show up in the ***Scripts*** menu but are instead run automatically when the \TeX works function they modify is used.

You can easily determine which type of script you have by opening the script file. Near the top of the file, you should find a line similar to

```
// Type: standalone
```

Alternatively—once the script is installed—you can use the dialogue available from ***Scripts***→***Scripting \TeX***works→***Manage Scripts*** to display this information.

6.2 Installing Scripts

A word of caution first: do not install scripts from a source you do not trust! Before installing scripts, you should make sure that the file you are about to install indeed does what you expect. Scripts are very powerful—they can do almost everything a normal program can do. So while there are some security precautions built into \TeX works, you should still be aware that scripts could potentially harm your computer and cause (among other things) crashes and data loss. In particular, scripts can read, create, and modify arbitrary files on your hard drive.

That said, installing scripts is very simple. Script files are generally installed in `<resources>/scripts` or subdirectories of it. These subdirectories are shown as submenus of the ***Scripts*** menu, so they can be used to group and categorize scripts. This is especially useful if you use many different scripts that would otherwise make the ***Scripts*** menu very confusing. One easy way to open the `scripts` folder is the ***Scripts***→***Scripting \TeX***works→***Show Scripts Folder*** menu item.

Since scripts are usually simple plain-text files, they do not come with fancy installers. To install them, simply copy or decompress (if archived, e.g., in a .zip file) the script file—and any other required files that you may have received—into `<resources>/scripts` or a subdirectory of it.

After having installed a new script file, T_EXworks needs to become aware of it. It automatically scans for all scripts during start-up, so you could close all T_EXworks windows and restart the application. An alternative is provided by the ***Scripts*→*Scripting T_EXworks*→*Reload Script List*** menu item which rescans all scripts without otherwise interfering with the program.

You can also disable scripts (or whole directories of scripts) if you want to. This can be useful if you do not need some scripts for some time and do not want them to clutter the ***Scripts*** menu, but do not want to uninstall them entirely. Or if you want to prevent hook scripts from being run automatically. To do this, open the “Manage Scripts” dialogue with the ***Scripts*→*Scripting T_EXworks*→*Manage Scripts*** menu item. Simply uncheck the script you want to disable and it won’t bother you again.

6.3 Using Scripts

Using scripts is simple. Hook scripts are used automatically—you don’t need to do anything. Standalone scripts show up in the ***Scripts*** menu or one of its submenus. If you cannot find a script you are looking for, or if you find a script you do not know the purpose of, you can use the “Manage Scripts” dialogue to get additional information (like the author, a brief description, etc.) about it.

Some scripts need to run other programs on your system. One example would be a script that opens the pdf in the system’s default previewer, e.g., for printing. Since running arbitrary commands can in some situations be particularly dangerous, this functionality is disabled by default. You will notice this when a dialogue pops up informing you of an error in the script, or a similar message is displayed in the status bar. To enable scripts to execute system commands, open the preferences dialogue via the ***Edit*→*Preferences...*** menu item. There, go to the “Scripts” tab and check the “Allow scripts to run system commands” option. If you want to disable this function again later just uncheck the option. Note that this option applies equally to all scripts—there is currently no way to allow command execution only for some scripts.

Beyond this manual

In this manual, the authors tried to give an overview over T_EXworks and a concise introduction to get you started. T_EXworks is constantly evolving and improving, however, so the information presented here will never be complete.

Additional, frequently updated documents are posted in the wiki hosted on GitHub at <https://github.com/TeXworks/texworks/wiki>. Particularly noteworthy are the following pages:

SpellingDictionaries describes how to obtain and install dictionaries for the spell-checker on various systems. <https://github.com/TeXworks/texworks/wiki/SpellingDictionaries>

TipsAndTricks provides a compilation of useful things to know at a glance, such as the % !TEX root construct. <https://github.com/TeXworks/texworks/wiki/TipsAndTricks>

AdvancedTypesettingTools lists the configurations for several typesetting tools that are not included in T_EXworks by default, such as latexmk or the dvips workflows. <https://github.com/TeXworks/texworks/wiki/AdvancedTypesettingTools>

If you run into problems with T_EXworks, it is advisable to browse the mailing list archives accessible via <http://tug.org/pipermail/texworks/>. If you use T_EXworks regularly or are interested in learning about problems and solutions when using it for some other reason, you can also consider subscribing to the list at <http://tug.org/mailman/listinfo/texworks> to stay up-to-date. For the occasional post to the mailing list, you can also use the *Help*→*Email to mailing list* menu item. Please make sure you replace the default subject by something describing your issue and to include

all information that might help resolving it. That way, you are much more likely to get many helpful replies.

If you find a bug in `TeXworks` or want to suggest a new feature you would like to see in a future version, you should have a look at the issue list at GitHub (<https://github.com/TeXworks/texworks/issues>). Before posting a new item, please make sure that a similar report or request is not already on the list and that the issue list is indeed the right place, though. If in doubt, please ask on the mailing list first.

Happy `TeX`ing!



Customizing T_EXworks

A.1 Syntax highlighting

Among its many other features, T_EXworks also include syntax highlighting. This means that certain things like L^AT_EX commands, environments, or comments are coloured, underlined, or highlighted in some other way. T_EXworks also provides the ability to switch between different highlighting schemes¹, and to define your own ones. This is useful if you often work with types of files for which no highlighting scheme is provided by default, or if you want to adjust the highlighting schemes to better match your system's colour scheme.

To modify the highlighting schemes, you have to edit the plain-text file `<resources>/configuration/syntax-patterns.txt`. This file can contain any number of individual sections, each defining a single highlighting scheme to be displayed in the menu structure of T_EXworks. To define a section, just write the name enclosed in square brackets on a line of its own. Naturally, these names should not include the `]` character. By default, the following two sections are defined:

```
[LaTeX]
[ConTeXt]
```

In addition, you can add comments to the file by starting a line with `#`. Empty lines are ignored.

Each section consists of an arbitrary number of styling rules. Each such instruction consists of three parts: a formatting instruction, a spell-check flag,

¹Use **Format**→**Syntax Coloring** to change the highlighting scheme for the current document, and **Edit**→Preferences...→**Editor**→**Syntax Coloring** to set the default one.

and a regular expression² defining what part of a text to match. These parts must all be on the same line, and separated by whitespaces (e.g., spaces or tabstop characters). Take for example the following line from the default LaTeX section:

```
red      Y      %.*
```

The first part, `red`, defines the format (in this case, a red foreground colour is specified). The second part, `Y`, defines that spellchecking should be enabled for text that matches this particular rule. Sometimes, it is useful to put `N` here to disable spellchecking. For example, if spellchecking would be enabled for L^AT_EX commands, most documents would be flooded with red underlines indicating misspelled words when in fact they are only special commands. Finally, the third part specifies that this rule should be applied to all text preceded by `%`.

Let us take a closer look at the three parts of each rule. In its most general form, the first part—the format instruction—looks like

```
<foreground_colour>/<background_colour>;<fontflags>
```

The `<fontflags>` can be specified independent of the colours (note, though, that it must always be preceded by a `;`). The background colour (together with the `/`) can be omitted, but if you specify it, you also have to specify the foreground colour.

Each colour can either be specified by an SVG name³ or by a hexadecimal value (`#rrggbb`⁴) similar as in web documents. The `<fontflags>` can be any combination of the letter `B` (bold), `I` (italic), and `U` (underlined).

Examples of valid formatting instructions are:

```
red
white/#000000
;B
blue;I
#000000/#ffff00;U
```

²For some details on regular expressions, see B

³See <https://www.w3.org/TR/SVG11/types.html#ColorKeywords> for a list of valid names.

⁴Because `#` is also used to mark comments if given as the first character of a line, you need to add a space, tab, or similar before specifying a hexadecimal foreground color.

A.2 Keyboard shortcuts

The use of keyboard shortcuts greatly facilitates typing in and the management of the source and the preview windows. Their use is much more effective than the use of buttons for frequently-used actions.

Below, you'll find the shortcuts for source and preview windows. Note that on macOS, **Ctrl** actually refers to the *Command key*, which is the usual modifier for keyboard shortcuts. Although the keyboard shortcuts are specified with **Ctrl**, this will appear as the *Command-key* symbol in menus. (To refer to the actual *Control key* on the Mac, the shortcut file should use the name **Meta**).










Note that the shortcuts listed below are the default shortcuts for the English interface of T_EXworks. Different languages may use different shortcuts.

All the shortcuts can be redefined either to create new shortcuts or to modify the existing ones to match personal uses or change shortcuts not adapted to one particular keyboard layout. The list of possible actions to associate with shortcuts is given after the predefined shortcuts.

To define your own shortcuts, put a file named `shortcuts.ini` in the `<resources>/configuration` folder, next to `auto-indent-patterns.txt`, `delimiter-pairs.txt`, ..., `texworks-config.txt`.

For example, this file could contain:

```
actionHard_Wrap = Shift+F3
actionLast_Page = Ctrl+End
actionFirst_Page = Ctrl+Home
actionWrap_Lines = F3
actionLine_Numbers = F4
actionBalance_Delimiters = F9
```

The first line defines that using   should open the hardwrap dialogue box in the source window; the second ( ) should bring you to the last page and   (third line) should take you to the first page; with  you want to wrap/unwrap lines in the source, with  you will show/hide line numbers and with  you intend to select the text between corresponding delimiters in the source.

A.2.1 Predefined shortcuts

For working in the source window:

Shortcut	Action
Ctrl+'	Go to Preview
Ctrl+=	Show Selection
Ctrl+A	Select All
Ctrl+Alt+S	Save All
Ctrl+B	Balance Delimiters
Ctrl+C	Copy
Ctrl+E	Copy to Find
Ctrl+F	Find...
Ctrl+G	Find Again
Ctrl+H	Find Selection
Ctrl+L	Go to Line...
Ctrl+N	New
Ctrl+O	Open...
Ctrl+Q	Quit TeXworks
Ctrl+R	Replace...
Ctrl+S	Save
Ctrl+Shift+C	Insert Citations...
Ctrl+Shift+E	Copy to Replace
Ctrl+Shift+N	New from Template...
Ctrl+Shift+R	Replace Again
Ctrl+Shift+S	Save As...
Ctrl+Shift+Z	Redo
Ctrl+Shift+[Uncomment
Ctrl+Shift+]	Comment
Ctrl+T	Typeset
Ctrl+V	Paste
Ctrl+W	Close
Ctrl+X	Cut
Ctrl+Z	Undo
Ctrl+[Unindent
Ctrl+\	Hide Console Output
Ctrl+]	Indent

Moving the cursor (hold  to select):

Shortcut	Action
→	1 character right
Ctrl+→	1 word right
←	1 character left
Ctrl+←	1 word left
↑	1 line up
↓	1 line down
PgUp	1 screen up
PgDown	1 screen down
Home	Begin of line
Ctrl+Home	Begin of document
End	End of line
Ctrl+End	End of document

For working in the preview window:

Shortcut	Action
Alt+Left	Go to previous view
Ctrl+'	Go to Source
Ctrl++	Zoom In
Ctrl+-	Zoom Out
Ctrl+1	Actual Size
Ctrl+2	Fit to Width
Ctrl+3	Fit to Window
Ctrl+4	Fit to Content Width
Ctrl+Backspace	Clear
Ctrl+C	Copy
Ctrl+F	Find...
Ctrl+G	Find Again
Ctrl+J	Go to Page...
Ctrl+N	New
Ctrl+O	Open...
Ctrl+P	Print PDF...
Ctrl+PgDown	Next Page
Ctrl+PgUp	Previous Page
Ctrl+Q	Quit TeXworks
Ctrl+Shift+F	Full Screen
Ctrl+Shift+N	New from Template...

Shortcut	Action
Ctrl+Shift+Z	Redo
Ctrl+T	Typeset
Ctrl+V	Paste
Ctrl+W	Close
Ctrl+X	Cut
Ctrl+Z	Undo

A.2.2 Actions listed alphabetically

actionAbout_Scripts	actionPageMode_Single
actionAbout_TW	actionPageMode_TwoPagesContinuous
actionActual_Size	actionPaste
actionApply_to_Selection	actionPlace_on_Left
actionAutoIndent_None	actionPlace_on_Right
actionAuto_Follow_Focus	actionPreferences
actionBalance_Delimiters	actionPrevious_Completion
actionClear	actionPrevious_Completion_Placeholder
actionClear_Recent_Files	actionPrevious_Page
actionClose	actionPrevious_ViewRect
actionComment	actionPrintPdf
actionCopy	actionQuit_TeXworks
actionCopy_to_Find	actionRedo
actionCopy_to_Replace	actionRemove_Aux_Files
actionCut	actionReplace
actionFind	actionReplace_Again
actionFind_Again	actionRevert_to_Saved
actionFind_Selection	actionSave
actionFirst_Page	actionSave_All
actionFit_to_Content_Width	actionSave_As
actionFit_to_Width	actionScroll
actionFit_to_Window	actionSelect_All
actionFont	actionSelect_Image
actionFull_Screen	actionSelect_Text
actionGoToHomePage	actionSettings_and_Resources
actionGo_to_Line	actionShow_Hide_Console
actionGo_to_Page	actionShow_Scripts_Folder
actionGo_to_Preview	actionShow_Selection
actionGo_to_Source	actionSide_by_Side

actionHard_Wrap	actionSmartQuotes_None
actionIndent	actionStack
actionInsert_Citations	actionSyntaxColoring_None
actionJump_To_PDF	actionTile
actionLast_Page	actionTo_Lowercase
actionLine_Numbers	actionTo_Uppercase
actionMagnify	actionToggle_Case
actionManage_Scripts	actionTypeset
actionNew	actionUncomment
actionNew_from_Template	actionUndo
actionNext_Completion	actionUnindent
actionNext_Completion_Placeholder	actionUpdate_Scripts
actionNext_Page	actionWrap_Lines
actionNone	actionWriteToMailingList
actionOpen	actionZoom_In
actionPageMode_Continuous	actionZoom_Out

A.2.3 Actions listed by menu

For the source window:

Edit	
actionBalance_Delimiters	actionPreferences
actionClear	actionRedo
actionCopy	actionSelect_All
actionCut	actionTo_Lowercase
actionInsert_Citations	actionTo_Uppercase
actionNone	actionToggle_Case
actionPaste	actionUndo

File	
actionClear_Recent_Files	actionRemove_Aux_Files
actionClose	actionRevert_to_Saved
actionNew	actionSave
actionNew_from_Template	actionSave_All
actionOpen	actionSave_As
actionQuit_TeXworks	

Format	
actionApply_to_Selection	actionLine_Numbers
actionAutoIndent_None	actionSmartQuotes_None
actionComment	actionSyntaxColoring_None

actionFont	actionUncomment
actionHard_Wrap	actionUnindent
actionIndent	actionWrap_Lines
Help	
actionAbout_TW	actionSettings_and_Resources
actionGoToHomePage	actionWriteToMailingList
Scripts	
actionAbout_Scripts	actionShow_Scripts_Folder
actionManage_Scripts	actionUpdate_Scripts
Search	
actionCopy_to_Find	actionGo_to_Line
actionCopy_to_Replace	actionReplace
actionFind	actionReplace_Again
actionFind_Again	actionShow_Selection
actionFind_Selection	
Typeset	
actionTypeset	
Window	
actionAuto_Follow_Focus	actionShow_Hide_Console
actionGo_to_Preview	actionSide_by_Side
actionPlace_on_Left	actionStack
actionPlace_on_Right	actionTile

For the preview window:

Edit	
actionCopy	actionRedo
actionPreferences	actionUndo
File	
actionClear_Recent_Files	actionOpen
actionClose	actionPrintPdf
actionNew	actionQuit_TeXworks
actionNew_from_Template	
Help	
actionAbout_TW	actionSettings_and_Resources
actionGoToHomePage	actionWriteToMailingList

Scripts	
<code>actionAbout_Scripts</code>	<code>actionShow_Scripts_Folder</code>
<code>actionManage_Scripts</code>	<code>actionUpdate_Scripts</code>
Search	
<code>actionFind</code>	<code>actionFind_Again</code>
Typeset	
<code>actionTypeset</code>	
View	
<code>actionActual_Size</code>	<code>actionNext_Page</code>
<code>actionFirst_Page</code>	<code>actionPageMode_Continuous</code>
<code>actionFit_to_Content_Width</code>	<code>actionPageMode_Single</code>
<code>actionFit_to_Width</code>	<code>actionPageMode_TwoPagesContinuous</code>
<code>actionFit_to_Window</code>	<code>actionPrevious_Page</code>
<code>actionFull_Screen</code>	<code>actionZoom_In</code>
<code>actionGo_to_Page</code>	<code>actionZoom_Out</code>
<code>actionLast_Page</code>	
Window	
<code>actionGo_to_Source</code>	<code>actionSide_by_Side</code>
<code>actionPlace_on_Left</code>	<code>actionStack</code>
<code>actionPlace_on_Right</code>	<code>actionTile</code>

A.2.4 Actions for typesetting tools

In addition to the static actions listed above, there are also actions for switching to a typesetting tool. All of these actions are of the general form `actionTypesetTool<tool_name>` where `<tool_name>` must be replaced appropriately. For example, the name corresponding to the pdfLaTeX typesetting tool is `actionTypesetToolpdfLaTeX`, that corresponding to My_{La}TeX tool is `actionTypesetToolMyLaTeX`.⁵

A.2.5 Actions for scripts


There are also actions for scripts. These are dynamic in nature, as they are created on-the-fly for the available scripts (which may change when you install scripts, remove them, or change some settings). All of these actions are of the form `Script:␣<script_title>`, where `<script_title>` must be

⁵Note that some characters (such as = and ; are reserved in ini files and must be escaped by %XX, where XX is the corresponding hexadecimal ASCII code. For example, the name corresponding to A=B is `actionTypesetToolA%3DB`.

replaced appropriately. If you have a script that shows up as `My Script`, for example, the corresponding action would be named `Script:␣My␣Script`.

A.3 Roots for completion

We give here the keywords for auto-completion as they are supplied by T_EXworks. They are given in the files `tw-basic.txt`, `tw-context.txt` (initially empty) and `tw-latex.txt` in the `<resource>\completion` folder.

We give them in three columns: the first two show the keywords, the third the (L^A)T_EX code produced. In some cases there is only the code, this means that you can start to enter the (L^A)T_EX code and try to complete it with .

During completion, the system inserts line feeds and puts the cursor at the first place where one has to enter information to complete the typing. To represent the line feeds we used \mathcal{R} and \mathcal{I} for the input point.

So, a line like “`\begin{abstract}\mathcal{R}\mathcal{I}\mathcal{R}\end{abstract}•`” should be interpreted as

```
\begin{abstract}
```

```
\end{abstract}•
```

with the cursor being position on the central, empty line.

It is possible to see that the keywords have some pattern. The mathematical variables have a keyword starting with `x`, when they are in a mathematical environment; when they are used alone in the text you add `d` in front. For example, `xa` and `dxa` give `\alpha`, if there is a capital there is a `c`, as `xo` for `\omega` and `xco` for `\Omega`. The keywords for environments start with `b`: `bali` for `\begin{align}` (`b` is a mnemonic for `\begin`). When the environment has possible options, there is one or more `o` added to the base name: `bminp` gives `\begin{minipage}{...}` while `bminpo` gives `\begin{minipage}[]{}...`.

Keywords defined in `tw-basic.txt` (defined in T_EX):

<code>xa</code>	<code>\xa</code>	<code>\alpha</code>
<code>xb</code>	<code>\xb</code>	<code>\beta</code>
	<code>\bsk</code>	<code>\bigskip</code>
		<code>\bigskip\mathcal{R}</code>
<code>xch</code>	<code>\xch</code>	<code>\chi</code>
<code>xcd</code>	<code>\xcd</code>	<code>\Delta</code>

xd	\xd	\delta
xe	\xe	\epsilon
xet	\xet	\eta
xcg	\xcg	\Gamma
xg	\xg	\gamma
		\hskip
		\indent
		\input
xio	\xio	\iota
xcl	\xcl	\Lambda
xl	\xl	\lambda
	\msk	\medskip
		\medskip \mathcal{R}
xm	\xm	\mu
		\noindent
xn	\xn	\nu
xco	\xco	\Omega
xo	\xo	\omega
		\par
xcph	\xcph	\Phi
xph	\xph	\phi
xcp	\xcp	\Pi
xp	\xp	\pi
xcps	\xcps	\Psi
xps	\xps	\psi
xr	\xr	\rho
		\scriptsize
xcs	\xcs	\Sigma
xs	\xs	\sigma
	\ssk	\smallskip \mathcal{R}
xt	\xt	\tau
tex	\tex	\TeX
texs	\texs	\TeX\
xcth	\xcth	\Theta
xth	\xth	\theta
xcu	\xcu	\Upsilon
xu	\xu	\upsilon
xve	\xve	\varepsilon
xvph	\xvph	\varphi
xvp	\xvp	\varpi
xvr	\xvr	\varrho

xvs	\xvs	\varsigma
xvth	\xvth	\vartheta
		\vskip
xcx	\xcx	\Xi
xx	\xx	\xi
xz	\xz	\zeta

Keywords defined in `tw-latex.txt` (defined in L^AT_EX):

ncol	\ncol	&
dd	\dd	\(\mathcal{I} \) •
	\adc	\addtocounter{\mathcal{I}}{•}
adcount		\addtocounter{\mathcal{I}}{•}\mathcal{R}
	\adl	\addtolength{\mathcal{I}}{•}
adlen		\addtolength{\mathcal{I}}{•}\mathcal{R}
		\author{\mathcal{I}}\mathcal{R}
		\begin{
babs	\babs	\begin{abstract}\mathcal{R}\mathcal{I}\mathcal{R}\end{abstract} •
balis	\balis	\begin{align*}\mathcal{R}\mathcal{I}\mathcal{R}\end{align*} •
balیات	\baliات	\begin{alignat*}{\mathcal{I}}\mathcal{R} • \mathcal{R}\end{alignat*} •
baliat	\baliat	\begin{alignat}{\mathcal{I}}\mathcal{R} • \mathcal{R}\end{alignat} •
baliedat	\baliedat	\begin{alignedat}\mathcal{R}\mathcal{I}\mathcal{R}\end{alignedat} •
baliedato	\baliedato	\begin{alignedat}[\mathcal{I}]\mathcal{R} • \mathcal{R}\end{alignedat} •
balied	\balied	\begin{aligned}\mathcal{I}\mathcal{R} • \mathcal{R}\end{aligned} •
bali	\bali	\begin{align}\mathcal{R}\mathcal{I}\mathcal{R}\end{align} •
bapp	\bapp	\begin{appendix}\mathcal{R}\mathcal{I}\mathcal{R}\end{appendix} •
barr		\begin{array}\mathcal{R}\mathcal{I}\mathcal{R}\end{array} •
bbmat	\bbmat	\begin{bmatrix}\mathcal{R}\mathcal{I}\mathcal{R}\end{bmatrix} •
bcase	\bcase	\begin{cases}\mathcal{R}\mathcal{I}\mathcal{R}\end{cases} •
bcent	\bcent	\begin{center}\mathcal{R}\mathcal{I}\mathcal{R}\end{center} •
bcenum	\bcenum	\begin{compactenum}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{compactenum} •
bcenumo	\bcenumo	\begin{compactenum}[\mathcal{I}]\mathcal{R}\item\mathcal{R} • \mathcal{R}\end{compactenum} •
bcitem	\bcitem	\begin{compactitem}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{compactitem} •
bcitemo	\bcitemo	\begin{compactitem}[\mathcal{I}]\mathcal{R}\item\mathcal{R} • \mathcal{R}\end{compactitem} •
bdes	\bdes	\begin{description}\mathcal{R}\item[\mathcal{I}]\mathcal{R} • \mathcal{R}\end{description} •
bdoc	\bdoc	\begin{document}\mathcal{R}\mathcal{I}\mathcal{R}\mathcal{R}\end{document}
benu	\benu	\begin{enumerate}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{enumerate} •
benuo	\benuo	\begin{enumerate}[\mathcal{I}]\mathcal{R}\item\mathcal{R} • \mathcal{R}\end{enumerate} •
beqns	\beqns	\begin{eqnarray*}\mathcal{R}\mathcal{I}\mathcal{R}\end{eqnarray*} •
beqn	\beqn	\begin{eqnarray}\mathcal{R}\mathcal{I}\mathcal{R}\end{eqnarray} •
bequs	\bequs	\begin{equation*}\mathcal{R}\mathcal{I}\mathcal{R}\end{equation*} •

bequ	<code>\bequ</code>	<code>\begin{equation}\mathcal{R}\mathcal{I}\mathcal{R}\end{equation}</code> •
bfig	<code>\bfig</code>	<code>\begin{figure}\mathcal{R}\mathcal{I}\mathcal{R}\end{figure}</code> •
bfigo	<code>\bfigo</code>	<code>\begin{figure}[I]\mathcal{R} \bullet \mathcal{R}\end{figure}</code> •
bflaligs	<code>\bflaligs</code>	<code>\begin{flalign*}\mathcal{R}\mathcal{I}\mathcal{R}\end{flalign*}</code> •
bflalig	<code>\bflalig</code>	<code>\begin{flalign}\mathcal{R}\mathcal{I}\mathcal{R}\end{flalign}</code> •
bfl	<code>\bfl</code>	<code>\begin{flushleft}\mathcal{R}\mathcal{I}\mathcal{R}\end{flushleft}</code> •
bflr	<code>\bflr</code>	<code>\begin{flushright}\mathcal{R}\mathcal{I}\mathcal{R}\end{flushright}</code> •
bgaths	<code>\bgaths</code>	<code>\begin{gather*}\mathcal{R}\mathcal{I}\mathcal{R}\end{gather*}</code> •
bgathed	<code>\bgathed</code>	<code>\begin{gathered}\mathcal{R}\mathcal{I}\mathcal{R}\end{gathered}</code> •
bgathedo	<code>\bgathedo</code>	<code>\begin{gathered}[I]\mathcal{R} \bullet \mathcal{R}\end{gathered}</code> •
bgath	<code>\bgath</code>	<code>\begin{gather}\mathcal{R}\mathcal{I}\mathcal{R}\end{gather}</code> •
bite	<code>\bite</code>	<code>\begin{itemize}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{itemize}</code> •
biteo	<code>\biteo</code>	<code>\begin{itemize}[I]\mathcal{R}\item\mathcal{R} \bullet \mathcal{R}\end{itemize}</code> •
blett	<code>\blett</code>	<code>\begin{letter}\{I\}\mathcal{R} \bullet \mathcal{R}\end{letter}</code> •
blist	<code>\blist</code>	<code>\begin{list}\{I\}\{ \bullet \}\mathcal{R}\item\mathcal{R} \bullet \mathcal{R}\end{list}</code> •
bminpo	<code>\bminpo</code>	<code>\begin{minipage}[I]\{ \bullet \}\mathcal{R} \bullet \mathcal{R}\end{minipage}</code> •
bminp	<code>\bminp</code>	<code>\begin{minipage}\{I\}\mathcal{R} \bullet \mathcal{R}\end{minipage}</code> •
bmults	<code>\bmults</code>	<code>\begin{multline*}\mathcal{R}\mathcal{I}\mathcal{R}\end{multline*}</code> •
bmult	<code>\bmult</code>	<code>\begin{multline}\mathcal{R}\mathcal{I}\mathcal{R}\end{multline}</code> •
bpict	<code>\bpict</code>	<code>\begin{picture}\mathcal{R}\mathcal{I}\mathcal{R}\end{picture}</code> •
bpmat	<code>\bpmat</code>	<code>\begin{pmatrix}\mathcal{R}\mathcal{I}\mathcal{R}\end{pmatrix}</code> •
bquot	<code>\bquot</code>	<code>\begin{quotation}\mathcal{R}\mathcal{I}\mathcal{R}\end{quotation}</code> •
bquo	<code>\bquo</code>	<code>\begin{quote}\mathcal{R}\mathcal{I}\mathcal{R}\end{quote}</code> •
bsplit	<code>\bsplit</code>	<code>\begin{split}\mathcal{R}\mathcal{I}\mathcal{R}\end{split}</code> •
bsubeq	<code>\bsubeq</code>	<code>\begin{subequations}\mathcal{R}\mathcal{I}\mathcal{R}\end{subequations}</code> •
btabb	<code>\btabb</code>	<code>\begin{tabbing}\mathcal{R}\mathcal{I}\mathcal{R}\end{tabbing}</code> •
btabs	<code>\btabs</code>	<code>\begin{table*}\mathcal{R}\mathcal{I}\mathcal{R}\end{table*}</code> •
btbls	<code>\btbls</code>	<code>\begin{table*}\mathcal{R}\mathcal{I}\mathcal{R}\end{table*}</code> •
btablo	<code>\btablo</code>	<code>\begin{table*}[I]\mathcal{R} \bullet \mathcal{R}\end{table*}</code> •
btblso	<code>\btblso</code>	<code>\begin{table*}[I]\mathcal{R} \bullet \mathcal{R}\end{table*}</code> •
btabl	<code>\btabl</code>	<code>\begin{table}\mathcal{R}\mathcal{I}\mathcal{R}\end{table}</code> •
btbl	<code>\btbl</code>	<code>\begin{table}\mathcal{R}\mathcal{I}\mathcal{R}\end{table}</code> •
btablo	<code>\btablo</code>	<code>\begin{table}[I]\mathcal{R} \bullet \mathcal{R}\end{table}</code> •
btblo	<code>\btblo</code>	<code>\begin{table}[I]\mathcal{R} \bullet \mathcal{R}\end{table}</code> •
btabs	<code>\btabs</code>	<code>\begin{tabular*}\{I\}\{ \bullet \}\mathcal{R} \bullet \mathcal{R}\end{tabular*}</code> •
btabx	<code>\btabx</code>	<code>\begin{tabularx}\{I\}\{ \bullet \}\mathcal{R} \bullet \mathcal{R}\end{tabularx}</code> •
btab	<code>\btab</code>	<code>\begin{tabular}\{I\}\mathcal{R} \bullet \mathcal{R}\end{tabular}</code> •
bbib	<code>\bbib</code>	<code>\begin{thebibliography}\{I\}\mathcal{R}\bibitem\{ \bullet \}\mathcal{R} \bullet \mathcal{R}\end{thebibliography}</code> •
bindex	<code>\bindex</code>	<code>\begin{theindex}\mathcal{R}\mathcal{I}\mathcal{R}\end{theindex}</code> •
btheo	<code>\btheo</code>	<code>\begin{theorem}\mathcal{R}\mathcal{I}\mathcal{R}\end{theorem}</code> •

btitle	\btitle	\begin{titlepage}\mathcal{R}\end{titlepage} •
btrivl	\btrivl	\begin{trivlist}\mathcal{R}\end{trivlist} •
bvarw	\bvarw	\begin{varwidth}{\mathcal{I}}\mathcal{R} • \mathcal{R}\end{varwidth} •
bverb	\bverb	\begin{verbatim}\mathcal{R}\end{verbatim} •
bvers	\bvers	\begin{verse}\mathcal{R}\end{verse} •
bfd		\bfseries
bibitemo		\bibitem[\mathcal{I}]{ • } \mathcal{R} •
bibitem		\bibitem{\mathcal{I}} \mathcal{R} •
bibstyle	\bibstyle	\bibliographystyle{\mathcal{I}}
biblio		\bibliography{\mathcal{I}}
		\boxed{\mathcal{I}}
		\caption{\mathcal{I}} \mathcal{R}
		\cdots
center		\centering
		\chapter{\mathcal{I}}
chap		\chapter{\mathcal{I}} \mathcal{R}
		\cite{\mathcal{I}}
		\cline{\mathcal{I}}
		\date{\mathcal{I}} \mathcal{R}
		\ddddot{\mathcal{I}}
		\dddot{\mathcal{I}}
		\ddots
		\ddot{\mathcal{I}}
		\documentclass[\mathcal{I}]{ • } \mathcal{R}
		\documentclass{\mathcal{I}} \mathcal{R}
		\dots
		\dotsb
		\dotsc
		\dotsi
		\dotsm
		\dotso
emd		\em
em		\emph{\mathcal{I}}
		\end{\mathcal{I}} \mathcal{R}
		\eqref{\mathcal{I}}
		\fboxrule{\mathcal{I}}
		\fboxsep{\mathcal{I}}
fbox		\fbox{\mathcal{I}}
		\footnotesize
foot		\footnote{\mathcal{I}}
frac		\frac{\mathcal{I}}{ • }

fboxoo	\fboxoo	\framebox[\mathcal{I}][\bullet]{\bullet}
fboxo	\fboxo	\framebox[\mathcal{I}]{\bullet}
hw		\headwidth
		\hline \mathcal{R}
		\hspace*{\mathcal{I}}
		\hspace{\mathcal{I}}
incgo		\includegraphics[\mathcal{I}]{\bullet}\mathcal{R}
incg		\includegraphics{\mathcal{I}}\mathcal{R}
		\include{\mathcal{I}}\mathcal{R}
		\input{\mathcal{I}}\mathcal{R}
		\intertext{\mathcal{I}}
		\item \mathcal{RI}
ito		\item[\mathcal{I}]\mathcal{R}\bullet
itd		\itshape
lbl	\lbl	\label{\mathcal{I}}
		\large
		\Large
latex	\latex	\LaTeX
latexs	\latexs	\LaTeX\
latexe	\latexe	\LaTeXe
latexes	\latexes	\LaTeXe\
		\ldots
		\listfiles \mathcal{R}
listf	\listf	\listoffigures \mathcal{R}
listt	\listt	\listoftables \mathcal{R}
		\makeatletter
		\makeatother
mboxoo	\mboxoo	\makebox[\mathcal{I}][\bullet]{\bullet}
mboxo	\mboxo	\makebox[\mathcal{I}]{\bullet}
mpar	\mpar	\marginpar{\mathcal{I}}
mbf	\mbf	\mathbf{\mathcal{I}}
mcal	\mcal	\mathcal{\mathcal{I}}
mit	\mit	\mathit{\mathcal{I}}
mnorm	\mnorm	\mathnormal{\mathcal{I}}
mrn	\mrn	\mathrm{\mathcal{I}}
msf	\msf	\mathsf{\mathcal{I}}
mtt	\mtt	\mathtt{\mathcal{I}}
mbox		\mbox{\mathcal{I}}
mdd		\mdseries
multc	\multc	\multicolumn{\mathcal{I}}{\bullet}{\bullet}
multic		\multicolumn{\mathcal{I}}{\bullet}{\bullet}{\bullet}

nct		$\backslash\mathrm{newcolumn}\mathrm{type}\{\mathcal{I}\}\{\bullet\}$
newct		$\backslash\mathrm{newcolumn}\mathrm{type}\{\mathcal{I}\}\{\bullet\}$
ncmoo		$\backslash\mathrm{newcommand}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\mathcal{R}$
newcoo		$\backslash\mathrm{newcommand}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\mathcal{R}$
ncmo		$\backslash\mathrm{newcommand}\{\mathcal{I}\}[\bullet]\{\bullet\}\mathcal{R}$
newco		$\backslash\mathrm{newcommand}\{\mathcal{I}\}[\bullet]\{\bullet\}\mathcal{R}$
ncm		$\backslash\mathrm{newcommand}\{\mathcal{I}\}\{\bullet\}\mathcal{R}$
newc		$\backslash\mathrm{newcommand}\{\mathcal{I}\}\{\bullet\}\mathcal{R}$
nenvoo		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\{\bullet\}\mathcal{R}$
neweoo		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\{\bullet\}\mathcal{R}$
nenvo		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}[\bullet]\{\bullet\}\{\bullet\}\mathcal{R}$
neweo		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}[\bullet]\{\bullet\}\{\bullet\}\mathcal{R}$
nenv		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}\{\bullet\}\{\bullet\}\mathcal{R}$
newe		$\backslash\mathrm{newenvironment}\{\mathcal{I}\}\{\bullet\}\{\bullet\}\mathcal{R}$
newlen		$\backslash\mathrm{newlength}\{\mathcal{I}\}\mathcal{R}$
nlen		$\backslash\mathrm{newlength}\{\mathcal{I}\}\mathcal{R}$
newlin		$\backslash\mathrm{newline}\mathcal{R}$
nline		$\backslash\mathrm{newline}\mathcal{R}$
newpg		$\backslash\mathrm{newpage}\mathcal{R}$
npg	$\backslash\mathrm{npg}$	$\backslash\mathrm{newpage}\mathcal{R}$
		$\backslash\mathrm{newtheorem}\{\mathcal{I}\}[\bullet]\{\bullet\}\mathcal{R}$
		$\backslash\mathrm{newtheorem}\{\mathcal{I}\}\{\bullet\}\mathcal{R}$
		$\backslash\mathrm{newtheorem}\{\mathcal{I}\}\{\bullet\}[\bullet]\mathcal{R}$
		$\backslash\mathrm{nocite}\{\mathcal{I}\}$
		$\backslash\mathrm{normalsize}$
		$\backslash\mathrm{pagebreak}\mathcal{R}$
pgref		$\backslash\mathrm{pageref}\{\mathcal{I}\}$
pgs		$\backslash\mathrm{pagestyle}\{\mathcal{I}\}\mathcal{R}$
pars		$\backslash\mathrm{paragraph}^*\{\mathcal{I}\}\mathcal{R}$
paro		$\backslash\mathrm{paragraph}[\mathcal{I}]\{\bullet\}\mathcal{R}$
par		$\backslash\mathrm{paragraph}\{\mathcal{I}\}\mathcal{R}$
parboxo		$\backslash\mathrm{parbox}[\mathcal{I}]\{\bullet\}\{\bullet\}$
pboxo	$\backslash\mathrm{pboxo}$	$\backslash\mathrm{parbox}[\mathcal{I}]\{\bullet\}\{\bullet\}$
parbox		$\backslash\mathrm{parbox}\{\mathcal{I}\}\{\bullet\}$
pbox	$\backslash\mathrm{pbox}$	$\backslash\mathrm{parbox}\{\mathcal{I}\}\{\bullet\}$
rboxoo	$\backslash\mathrm{rboxoo}$	$\backslash\mathrm{raisebox}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}$
rboxo	$\backslash\mathrm{rboxo}$	$\backslash\mathrm{raisebox}\{\mathcal{I}\}[\bullet]\{\bullet\}$
rbox	$\backslash\mathrm{rbox}$	$\backslash\mathrm{raisebox}\{\mathcal{I}\}\{\bullet\}$
ref		$\backslash\mathrm{ref}\{\mathcal{I}\}$
rncmoo		$\backslash\mathrm{renewcommand}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\mathcal{R}$
rnewcoo		$\backslash\mathrm{renewcommand}\{\mathcal{I}\}[\bullet][\bullet]\{\bullet\}\mathcal{R}$

rncmo		<code>\renewcommand{\mathcal}[1]{\bullet}</code>
rnewco		<code>\renewcommand{\mathcal}[1]{\bullet}</code>
rncm		<code>\renewcommand{\mathcal}[1]{\bullet}</code>
rnewc		<code>\renewcommand{\mathcal}[1]{\bullet}</code>
rnc		<code>\rmfamily</code>
		<code>\rule[1]{\bullet}{\bullet}</code>
		<code>\rule{\mathcal}[1]{\bullet}</code>
scd		<code>\scshape</code>
secs		<code>\section*{\mathcal}\mathcal</code>
seco		<code>\section[1]{\bullet}\mathcal</code>
sec		<code>\section{\mathcal}\mathcal</code>
		<code>\setlength{\mathcal}{\bullet}</code>
hw2tw		<code>\setlength{\headwidth}{\textwidth}\mathcal</code>
sfd		<code>\sffamily</code>
sld		<code>\slshape</code>
sqrto	<code>\sqrtto</code>	<code>\sqrt[1]{\bullet}</code>
sqrto	<code>\sqrt</code>	<code>\sqrt{\mathcal}</code>
stcount		<code>\stepcounter{\mathcal}\mathcal</code>
spars	<code>\spars</code>	<code>\subparagraph*{\mathcal}</code>
sparo	<code>\sparo</code>	<code>\subparagraph[1]{\bullet}</code>
spar	<code>\spar</code>	<code>\subparagraph{\mathcal}</code>
ssecs	<code>\ssecs</code>	<code>\subsection*{\mathcal}\mathcal</code>
sseco	<code>\sseco</code>	<code>\subsection[1]{\bullet}\mathcal</code>
ssec	<code>\ssec</code>	<code>\subsection{\mathcal}\mathcal</code>
sssecs	<code>\sssecs</code>	<code>\subsubsection*{\mathcal}\mathcal</code>
	<code>\ssseco</code>	<code>\subsubsection[1][\bullet]\mathcal</code>
ssseco		<code>\subsubsection[1]{\bullet}\mathcal</code>
sssec	<code>\sssec</code>	<code>\subsubsection{\mathcal}\mathcal</code>
tableof- contents		<code>\tableofcontents\mathcal</code>
toc	<code>\toc</code>	<code>\tableofcontents\mathcal</code>
tilde	<code>\tilde</code>	<code>\textasciitilde</code>
bf	<code>\bf</code>	<code>\textbf{\mathcal}</code>
—		<code>\textendash\</code>
—		<code>\textendash\</code>
it	<code>\it</code>	<code>\textit{\mathcal}</code>
	<code>\rm</code>	<code>\textrm{\mathcal}</code>
sc	<code>\sc</code>	<code>\textsc{\mathcal}</code>
sf	<code>\sf</code>	<code>\textsf{\mathcal}</code>
sl	<code>\sl</code>	<code>\textsl{\mathcal}</code>
tt	<code>\tt</code>	<code>\texttt{\mathcal}</code>

up	<code>\up</code>	<code>\textup{\mathcal{I}}</code>
tw	<code>\tw</code>	<code>\textwidth</code>
		<code>\text{\mathcal{I}}</code>
		<code>\thanks{\mathcal{I}}\mathcal{R}</code>
		<code>\title{\mathcal{I}}\mathcal{R}</code>
ttd		<code>\ttfamily</code>
upd		<code>\upshape</code>
url		<code>\url{\mathcal{I}}</code>
usepo		<code>\usepackage[\mathcal{I}]{\bullet}\mathcal{R}</code>
usep		<code>\usepackage{\mathcal{I}}\mathcal{R}</code>
		<code>\vdots</code>
		<code>\vspace*\mathcal{I}\mathcal{R}</code>
		<code>\vspace{\mathcal{I}}\mathcal{R}</code>
		<code>{abstract}\mathcal{R}\mathcal{I}\mathcal{R}\end{abstract}\bullet</code>
		<code>{align*}\mathcal{R}\mathcal{I}\mathcal{R}\end{align*}\bullet</code>
		<code>{alignat*}{\mathcal{I}}\mathcal{R}\bullet\mathcal{R}\end{alignat*}\bullet</code>
		<code>{alignat}{\mathcal{I}}\mathcal{R}\bullet\mathcal{R}\end{alignat}\bullet</code>
		<code>{alignedat}{\mathcal{I}}\mathcal{R}\bullet\mathcal{R}\end{alignedat}\bullet</code>
		<code>{aligned}\mathcal{R}\mathcal{I}\mathcal{R}\end{aligned}\bullet</code>
		<code>{aligned}[\mathcal{I}]\mathcal{R}\bullet\mathcal{R}\end{aligned}\bullet</code>
		<code>{align}\mathcal{R}\mathcal{I}\mathcal{R}\end{align}\bullet</code>
		<code>{appendix}\mathcal{R}\mathcal{I}\mathcal{R}\end{appendix}\bullet</code>
		<code>{array}\mathcal{R}\mathcal{I}\mathcal{R}\end{array}\bullet</code>
		<code>{bmatrix}\mathcal{R}\mathcal{I}\mathcal{R}\end{bmatrix}\bullet</code>
		<code>{cases}\mathcal{R}\mathcal{I}\mathcal{R}\end{cases}\bullet</code>
		<code>{center}\mathcal{R}\mathcal{I}\mathcal{R}\end{center}\bullet</code>
		<code>{compactenum}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{compactenum}\bullet</code>
		<code>{compactenum}[\mathcal{I}]\mathcal{R}\item\mathcal{R}\bullet\mathcal{R}\end{compactenum}\bullet</code>
		<code>{compactitem}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{compactitem}\bullet</code>
		<code>{compactitem}[\mathcal{I}]\mathcal{R}\item\mathcal{R}\bullet\mathcal{R}\end{compactitem}\bullet</code>
		<code>{description}\mathcal{R}\item[\mathcal{I}]\mathcal{R}\bullet\mathcal{R}\end{description}\bullet</code>
		<code>{document}\mathcal{R}\mathcal{R}\mathcal{I}\mathcal{R}\mathcal{R}\end{document}</code>
		<code>{enumerate}\mathcal{R}\item\mathcal{R}\mathcal{I}\mathcal{R}\end{enumerate}\bullet</code>
		<code>{enumerate}[\mathcal{I}]\mathcal{R}\item\mathcal{R}\bullet\mathcal{R}\end{enumerate}\bullet</code>
		<code>{eqnarray*}\mathcal{R}\mathcal{I}\mathcal{R}\end{eqnarray*}\bullet</code>
		<code>{eqnarray}\mathcal{R}\mathcal{I}\mathcal{R}\end{eqnarray}\bullet</code>
		<code>{equation}\mathcal{R}\mathcal{I}\mathcal{R}\end{equation}\bullet</code>
		<code>{figure}\mathcal{R}\mathcal{I}\mathcal{R}\end{figure}\bullet</code>
		<code>{figure}[\mathcal{I}]\mathcal{R}\bullet\mathcal{R}\end{figure}\bullet</code>
		<code>{flalign*}\mathcal{R}\mathcal{I}\mathcal{R}\end{flalign*}\bullet</code>
		<code>{flalign}\mathcal{R}\mathcal{I}\mathcal{R}\end{flalign}\bullet</code>

```

{flushleft}\RIR\end{flushleft} •
{flushright}\RIR\end{flushright} •
{gather*}\RIR\end{gather*} •
{gathered}\RIR\end{gathered} •
{gathered}[I]\R • \R\end{gathered} •
{gather}\RIR\end{gather} •
{itemize}\R\item\RIR\end{itemize} •
{itemize}[I]\R\item\R • \R\end{itemize} •
{letter}\{I\}\R • \R\end{letter} •
{list}\{I\}\{ • \}\R\item\R • \R\end{list} •
{minipage}[I]\{ • \}\R • \R\end{minipage} •
{minipage}\{I\}\R • \R\end{minipage} •
{multline*}\RIR\end{multline*} •
{multline}\RIR\end{multline} •
{picture}\RIR\end{picture} •
{pmatrix}\RIR\end{pmatrix} •
{quotation}\RIR\end{quotation} •
{quote}\RIR\end{quote} •
{split}\RIR\end{split} •
{subequations}\RIR\end{subequations} •
{tabbing}\RIR\end{tabbing} •
{table*}\RIR\end{table*} •
{table*}[I]\R • \R\end{table*} •
{table}\RIR\end{table} •
{table}[I]\R • \R\end{table} •
{tabular*}\{I\}\{ • \}\R • \R\end{tabular*} •
{tabularx}\{I\}\{ • \}\R • \R\end{tabularx} •
{tabular}\{I\}\R • \R\end{tabular} •
{thebibliography}\RIR\end{thebibliography} •
{theindex}\RIR\end{theindex} •
{theorem}\RIR\end{theorem} •
{titlepage}\RIR\end{titlepage} •
{trivlist}\RIR\end{trivlist} •
{varwidth}\{I\}\R • \R\end{varwidth} •
{verbatim}\RIR\end{verbatim} •
{verse}\RIR\end{verse} •

```

Keywords defined in `tw-latex-pkg.txt` (defined by various packages):

	<code>\addbibresource{I}\R</code>
<code>botr</code>	<code>\bottomrule\R</code>

	<code>\citep{I}</code>
	<code>\citet{I}</code>
<code>cmidr</code>	<code>\cmidrule(I){•}</code>
<code>cmidro</code>	<code>\cmidrule[I](•){•}</code>
	<code>\enquote{I}</code>
<code>geometry</code>	<code>\geometry{•}</code>
<code>href</code>	<code>\href{I}{•}</code>
	<code>\midruleR</code>
	<code>\printbibliographyR</code>
	<code>\printbibliography[I]R</code>
<code>topr</code>	<code>\topruleR</code>
	<code>{compactenum}R\itemRIR\end{compactenum}•</code>
	<code>{compactenum}[I]R\itemR•R\end{compactenum}•</code>
	<code>{compactitem}R\itemRIR\end{compactitem}•</code>
	<code>{compactitem}[I]R\itemR•R\end{compactitem}•</code>
	<code>{tikzpicture}RIR\end{tikzpicture}•</code>
	<code>{tikzpicture}[I]R•R\end{tikzpicture}•</code>

Keywords defined in `tw-beamer.txt` (defined by the beamer package):

	<code>\action<I>{•}</code>
	<code>\againframe<I>[•]{•}R</code>
	<code>\againframe<I>{•}R</code>
	<code>\againframe{I}R</code>
	<code>\alert<I>{•}</code>
	<code>\alert{I}</code>
	<code>\alt<I>{•}{•}</code>
<code>bfrm</code>	<code>\begin{frame}RIR\end{frame}•</code>
	<code>\framesubtitle<I>{•}R</code>
	<code>\framesubtitle{I}R</code>
	<code>\frametitle<I>[•]{•}R</code>
	<code>\frametitle{I}R</code>
	<code>\framezoom<I><•>(•,•)(•,•)</code>
	<code>\framezoom<I><•>[•](•,•)(•,•)</code>
	<code>\includeonlyframes{I}R</code>
	<code>\includeonlylecture{I}R</code>
	<code>\invisible<I>{•}</code>
	<code>\lecture{I}{•}R</code>
	<code>\note<I>[•]{•}R</code>
	<code>\note<I>{•}R</code>
	<code>\note{I}R</code>

```

\only<I>\{ • }
\onslide<I>\{ • }
\pause\mathcal{R}
\pause[I]\mathcal{R}
\structure\{I\}\mathcal{R}
\temporal<I>\{ • \}\{ • \}\{ • \}
\uncover<I>\{ • }
\usebeamercolor\{I\}\mathcal{R}
\visible<I>\{ • }
\actionenv<I>\mathcal{R}\mathcal{R}\end\actionenv •
\alertenv<I>\mathcal{R}\mathcal{R}\end\alertenv •
\altenv<I>\{ • \}\{ • \}\{ • \}\{ • \}\mathcal{R}\mathcal{R}\end\altenv •
\beamercolorbox[I]\{ • \}\mathcal{R}\mathcal{R}\end\beamercolorbox •
\beamercolorbox\{I\}\mathcal{R}\mathcal{R}\end\beamercolorbox •
\block<I>\{ • \}\mathcal{R}\mathcal{R}\end\block •
\block\{I\}\mathcal{R}\mathcal{R}\end\block •
\columns\mathcal{R}\mathcal{I}\mathcal{R}\end\columns •
\columns[I]\mathcal{R} • \mathcal{R}\end\columns •
\column[I]\{ • \}\mathcal{R}\mathcal{R}\end\column •
\column\{I\}\mathcal{R}\mathcal{R}\end\column •
\frame\mathcal{R}\mathcal{I}\mathcal{R}\end\frame •
\frame[I]\{ • \}\mathcal{R}\mathcal{R}\end\frame •
\frame\{I\}\mathcal{R}\mathcal{R}\end\frame •
\onlyenv<I>\mathcal{R}\mathcal{R}\end\onlyenv •
\overlayarea\{I\}\{ • \}\mathcal{R}\mathcal{R}\end\overlayarea •
\overprint\mathcal{R}\mathcal{I}\mathcal{R}\end\overprint •
\structureenv\mathcal{R}\mathcal{I}\mathcal{R}\end\structureenv •


```

Keywords defined in `tw-context.txt` (defined in ConT_EXt):

```

\starttext
\stoptext

```

There are also environment codes (above) without `\begin{}` (which is itself a keyword); this allows to finish the environment name alone by  if one started to input it manually.



Regular expressions

As T_EXworks is built on Qt4, the available regular expressions—which are often referred to as **regex**¹—are a subset of those found in Qt4. See the site of Qt4¹ for more information. It is possible to find other information about regexps on the net² or from books. But pay attention that not all systems (programming languages, editors, . . .) use the same set of instructions; there is no “standard set”, unfortunately.

B.1 Introduction

When searching and replacing, one has to define the text to be found. This can be the text itself (e.g., “Abracadabra”), but often it is necessary to define the strings in a more generic and powerful way to avoid repeating the same operation many times with only small changes from one time to the next; if, for example, one wants to replace sequences of the letter **a** by ones of the letter **o**, but only those sequences of 3, 4, 5, 6 or 7 **a**; this would require repeating (and slightly adjusting) the find and replace procedure 5 times. Another example: replace all vowels by **§**—again, this would take 5 replace operations. Here come the regular expressions!

A simple character (a or 9) represents itself. But a set of characters can be defined: **[aeiou]** will match any vowel, **[abcdef]** the letters **a**, **b**, **c**, **d**, **e**, and **f**; this last set can be shortened as **[a-f]** using “-” between the two ends of the range. This can even be combined: **[a-zA-Z0-9]** will match all letters and all numbers.

¹<http://doc.trolltech.com/4.4/qregex.html#details>—this section is based on the information provided there

²see, for example, Wikipedia

To define a complementary set³, one uses “^”: the caret negates the character set if it occurs at the beginning, i.e., immediately after the opening square bracket. `[^abc]` matches anything except **a**, **b**, **c**.

B.2 Codes to represent special sets

When using regexps, one very often has to create a search expression which represents other strings in a generic way. If you are looking for a string that matches email addresses, for example, the letters and symbols will vary; still, you could search for any string which corresponds to the structure of an email address (`<text>@<text>.<text>`, roughly). To facilitate this, there are abbreviations to represent letters, figures, symbols, ...

These codes replace and facilitate the definition of sets; for example, to instead of manually defining the set of digits `[0-9]`, one can use “`\d`”. The following table lists the replacement codes.⁴

³A set of characters that are not allowed to occur for this regular expression to match the text

⁴simplified from Qt4 at trolltech, see note 1

Element	Meaning
<code>c</code>	Any character represents itself unless it has a special regexp meaning. Thus <code>c</code> matches the character <code>c</code> .
<code>\c</code>	A special character that follows a backslash matches the character itself except where mentioned below. For example, if you wished to match a literal caret at the beginning of a string you would write “ <code>\^</code> ”.
<code>\n</code>	This matches the ASCII line feed character (LF, Unix newline, used in TeXworks).
<code>\r</code>	This matches the ASCII carriage return character (CR).
<code>\t</code>	This matches the ASCII horizontal tab character (HT).
<code>\v</code>	This matches the ASCII vertical tab character (VT; almost never used).
<code>\xhhhh</code>	This matches the Unicode character corresponding to the hexadecimal number <code>hhhh</code> (between <code>0x0000</code> and <code>0xFFFF</code>). <code>\0ooo</code> (i.e., zero-ooo) matches the ASCII/Latin-1 character corresponding to the octal number <code>ooo</code> (between <code>0</code> and <code>0377</code>).
<code>.</code> (dot)	This matches any character (including newline). So if you want to match the dot character itself, you have to escape it with “ <code>\.</code> ”.
<code>\d</code>	This matches a digit.
<code>\D</code>	This matches a non-digit.
<code>\s</code>	This matches a white space.
<code>\S</code>	This matches a non-white space.
<code>\w</code>	This matches a word character or “ <code>_</code> ”.
<code>\W</code>	This matches a non-word character.
<code>\1, ...</code>	The <i>n</i> -th back-reference, e.g. <code>\1</code> , <code>\2</code> , etc.; used in the replacement string with capturing patterns—see below

Using these abbreviations is better than describing the set, because the abbreviations remain valid in different alphabets.

Pay attention that the end of line is often taken as a white space. Under TeXworks the end of line is referred to by “`\n`”.

B.3 Repetition

One doesn’t work only on single letters, digits, symbols; most of the time, these are repeated (e.g., a number is a repetition of digits and symbols—in the right order).

To show the number of repetitions, one uses a so called “quantifier”: $\mathbf{a}\{1,1\}$ means at least one and only one \mathbf{a} , $\mathbf{a}\{3,7\}$ means between at least 3 and at most 7 \mathbf{a} ; $\{1,1\}$ is redundant, of course, so $\mathbf{a}\{1,1\} = \mathbf{a}$.

This can be combined with the set notation: $[0-9]\{1,2\}$ will correspond to at least one digit and at most two, the integer numbers between 0 and 99. But this will match any group of 1 or 2 digits within any arbitrary string (which may have a lot of text before and after the integer); if we want this to match only if the whole string consists *entirely* of 1 or 2 digits (without any other characters preceding or following them), we can rewrite the regular expression to read $^[0-9]\{1,2\}$; here, the $^$ specifies that any match must start at the first character of the string, while the $\$$ says that any matching substring must end at the last character of the string, so the string can only be comprised of one or two digits ($^$ and $\$$ are so-called “assertions”—more on them later).

Here is a table of quantifiers.⁵ E represents an arbitrary expression (letter, abbreviation, set).

$E\{n,m\}$	Matches at least n occurrences of the expression and at most m occurrences of the expression.
$E\{n\}$	Matches exactly n occurrences of the expression. This is the same $E\{n,n\}$ or as repeating the expression n times.
$E\{n,\}$	Matches at least n occurrences of the expression.
$E\{,m\}$	Matches at most m occurrences of the expression.
$E?$	Matches zero or one occurrence of E . This quantifier effectively means <i>the expression is optional</i> (it may be present, but doesn't have to). It is the same as $E\{0,1\}$.
$E+$	Matches one or more occurrences of E . This is the same as $E\{1,\}$.
E^*	Matches zero or more occurrences of E . This is the same as $E\{0,\}$. Beware, the $*$ quantifier is often used by mistake instead of the $+$ quantifier. Since it matches zero or more occurrences, it will match even if the expression is not present in the string.

B.4 Alternatives and assertions

When searching, it is often necessary to search for alternatives, e.g., apple, pear, or cherry, but not pineapple. To separate the alternatives, one uses $|$: apple|pear|cherry . But this will not prevent to find pineapple, so we have to

⁵simplified from Qt4 at trolltech, see note 1

specify that apple should be standalone, a whole word (as is often called in the search dialog boxes).

To specify that a string should be considered standalone, we specify that it is surrounded by word separators/boundaries (begin/end of sentence, space), like `\bapple\b`. For our alternatives example we will **group** them by parentheses and add the boundaries `\b(apple|pear|cherry)\b`. Apart from `\b` we have already seen `^` and `$` which mark the boundaries of the whole string.

Here a table of the “assertions” which do not correspond to actual characters and will never be part of the result of a search. ⁶

<code>^</code>	The caret signifies the beginning of the string. If you wish to match a literal <code>^</code> , you must escape it by writing <code>\^</code>
<code>\$</code>	The dollar signifies the end of the string. If you wish to match a literal <code>\$</code> , you must escape it by writing <code>\\$</code>
<code>\b</code>	A word boundary.
<code>\B</code>	A non-word boundary. This assertion is true wherever <code>\b</code> is false.
<code>(?=E)</code>	Positive lookahead. This assertion is true if the expression E matches at this point.
<code>(?!E)</code>	Negative lookahead. This assertion is true if the expression E does not match at this point.

Notice the different meanings of `^` as assertion and as negation inside a character set!

B.5 Final notes

Using regexp is very powerful, but also quite dangerous; you could change your text at unseen places and sometimes reverting to the previous situation is not possible entirely. If you immediately see the error, you can try `Ctrl Z`.

Showing how to exploit the full power of regexp would require much more than this extremely short summary; in fact it would require a full manual on its own.

Also note that there are some limits in the implementation of regexps in `TeXworks`; in particular, the assertions (`^` and `$`) only consider the whole file, and there are no look-behind assertions.

Finally, do not forget to “tick” the regexp option when using them in the *Find* and *Replace* dialogs and to un-tick the option when not using regexps.

⁶simplified from Qt4 at trolltech, see note 1



Compiling T_EXworks

A complete guide how to compile T_EXworks is far beyond the scope of this manual. However, most users should find precompiled versions suitable for their system come either with their T_EX distribution or their operating system. If this is not the case, several precompiled versions can also be downloaded from <http://www.tug.org/texworks/>.

Compiling T_EXworks yourself is only necessary if your system is not (yet) supported, if you want to always have the latest features (and bugs), or generally want to help in improving T_EXworks further. To this end, there are some documents giving detailed instructions to compile T_EXworks on different machines.

***nix** <https://github.com/TeXworks/texworks/wiki/Building>

Mac [https://github.com/TeXworks/texworks/wiki/Building-on-Mac-OS-X-\(Homebrew\)](https://github.com/TeXworks/texworks/wiki/Building-on-Mac-OS-X-(Homebrew))

Win [https://github.com/TeXworks/texworks/wiki/Building-on-Windows-\(MinGW\)](https://github.com/TeXworks/texworks/wiki/Building-on-Windows-(MinGW))

Acknowledgements

Microsoft, Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States and other countries.

Apple, Mac, and macOS are trademarks of Apple Inc., registered in the U.S. and other countries.

GitHub is a trademark of GitHub Inc.

Unless noted otherwise, all icons are either part of T_EXworks or part of the Tango Icon Library (http://tango.freedesktop.org/Tango_Icon_Library).

Bibliography

- [1] **D. Knuth**, *The T_EXbook*, Addison Wesley, 1986-1992
- [2] **D. Knuth**, *The METAFont book*, Addison Wesley, 1986-1992
- [3] **L. Lamport**, *L^AT_EX: A Document Preparation System*, Addison Wesley, 1985 (L^AT_EX 2.09), 1994 (L^AT_EX 2_ε)
- [4] **M. Goossens, F. Mittelbach & A. Samarin**, *The L^AT_EX Companion*, Addison Wesley, 1994
- [5] **M. Goossens**, *The XeTeX Companion*, July 2009, <http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf>
- [6] **D. J. Perry**, *Creating Scholarly Multilingual Documents Using Unicode, OpenType, and XeTeX*, June 2009, <http://scholarsfonts.net/xetextt.pdf>

Index

- % !TeX
 - encoding, 26
 - program, 26
 - root, 17
 - spellcheck, 27
- actions
 - alphabetically, 41
 - by menu, 42
- auto-completion, 22
 - roots, 45
 - tw-basic.txt, 45
 - tw-beamer.txt, 55
 - tw-context.txt, 56
 - tw-latex-pkg.txt, 54
 - tw-latex.txt, 47
- cleaning folder, 28
 - aux files, 29
- comments, 11
- compiling T_EXworks, 62
- completion, *see* auto-completion
- configuration, 29
 - defaultbinpaths, 30
 - inipath, 29
 - libpath, 29
 - texworks-config.txt, 29
 - texworks-setup.ini, 29
 - texworks.ini, 29
- console bar, 13
- CTAN, 11
- Ctrl+', 11
- Ctrl+T, 10
- document
 - creation, 10
 - previewing, 10
 - source, 2
 - typesetting, 10
- editing
 - change case, 21
 - comment, 21
 - indentation, 27
 - line numbers, 21
 - redo, 21
 - search/replace, 18
 - select a block, 22
 - tools, 21
 - uncomment, 21
 - undo, 21
- editor, 2

- font, 15
- encoding, 26
 - latin1, 25
 - utf-8, 11, 25
- errors, 12
- extension .tex, 10
- files
 - format, 1
- folder
 - auto-completion, 6
 - configuration, 6
 - dictionaries, 6
 - resource, 6, 7
 - templates, 6
 - translations, 6
- installation, 5
 - Linux, 6
 - Mac, 7
 - Portable Usage, 29
 - Windows, 5
- interface, 8
- Kew, Jonathan, 3
- keyboard shortcuts, 38
 - actions, 38
 - predefined, 39
 - shortcuts.ini, 38
- Knuth, Donald E., 1
- Lamport, Leslie, 1
- log, 10
- METAFONT, 1
- METAPOST, 2
- output panel, 10
 - hide, 14
- packages, 1, 11
- PDF, 2
- Portable Usage, 29
- PostScript, 2
- preamble, 11
- preferences, 15
- preview, 8
- preview window, 10
- programme
 - default, 26
- project, 16
- regular expressions, 19, 57
 - alternatives/assertions, 60
 - introduction, 57
 - repetition, 59
 - sets, 58
- scripts, 31
 - installing, 32
 - managing, 33
 - using, 33
 - writing, 31
- search/replace, *see* editing
- shortcuts, *see* keyboard shortcuts
- source/editor, 8
- spell-checking, 17
 - .aff files, 18
 - .dic files, 18
- SyncTeX, 25
- syntax highlighting, 36
- tags, 28
 - structure, 28
 - table of contents, 28
- template, 16
- T_EX, 1
 - $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX, 1
 - ConT_EXt, 1
 - L^AT_EX, 1, 9
 - distribution, *see* T_EX distribution
 - dvips, 2
 - LuaT_EX, 2
 - pdftex, 2

- XeTeX, 2
- TeX distribution, 5
 - Linux, 5
 - Mac, 5
 - MacTeX, 5
 - TeX Live, 5
 - Windows, 5
 - MikTeX, 5
- TeXworks, 2
 - parameters, 15
- toolbar, 8
- typeset, 8
- typing cursor, 13
- utf-8, *see* encoding, utf-8
- weird characters, *see* encoding
- windows, 28
- wrap lines
 - automatic, 27
 - hard, 27
- WYSIWYG, 8
- zoom, 28