NEdit LaTeX-mode

Or LaTeX-editing made easy

by Jörg Fischer

Version 0.9.1, April 24, 2005

http://nedit.gmxhome.de

Contents

Description

The LaTeX-MODE is a package or set of macros, i.e. small programs, written in the NEdit macro or scripting language. It will turn NEdit, the Nirvana text editor, into an advanced LaTeX editor. NEdit version 5.4 is required! NEdit is a Unix program for the X Window System. It is Free Software in the sense of the GPL. You can download the latest version from NEdit.org.

Notice that the macros need a redefinition of the key bindings. This redefinition of the key bindings assumes that you are using a standard *US keyboard layout*. If you do not use such a keyboard layout, you probably have to adapt the key bindings to your situation in order not to get unexpected results!

Notice that some macros require NEdit to be in server mode, see for example 5.1.4 and 5.1.7. You run NEdit in server mode by starting it with `nc` or `nedit -server`.

Notice that some parts of this manual may not be up to date — please, be patient. For a short note on what's new, go to 9. Notice also that there is a set of *example files* to get you started quickly. It illustrates the basic features of the LaTeX-MODE, containing editing mathematical equations and multi-file documents (references, sectioning, bookmarks). *You should*

have a look at it!

Finally, these macros are only my personal customizations. Since editing styles, needs and tastes are completely different, you should adjust everything to suit your needs, preferences and usage patterns. *Make your text editor work the way you want, rather than the other way round!*

Notational Conventions

I will try to keep to the following notations, in order to ease reading:

- Keys are written in small capital letters, e.g., SHIFT+SPACE means to hit the SPACE key while holding the SHIFT key.
- Names of files and directories are enclosed in '.', e.g., '.nedit'.
- Names of menu entries are enclosed in ".",e.g., "Run $\longrightarrow$ Preview".
- Commands, variables and generally source code are written in type-face, e.g., `nedit -import .nedit`.
- Emphasized words are *italic* and very important points are red.

1. Motivation

Perhaps you are content with the way your text editing tool is working. More probably you are not. Usually however, you don't know what to do about it at first. So you live with it the way it is. However, there are capable text editing tools that can be adapted to your needs, and these tools are freely available. Since nobody can know what you need, what you would like to have, or what your usage patterns are, realistically the only possible way to get a text editor that does what you want is that you configure it in such a way yourself. The only questions are how can this be done, how hard it will be, and how long it will take? This means basically, will it be worth the effort?

My answer is yes, if you choose the right tool for the job. This means not only a tool that can do everything what you would like, but also a tool that you can easily persuade to do so. I have written the LaTeX-MODE mainly for me. It is a collection of macros that evolved over the years and still evolves. Although these macros follow my usage patterns, I believe it might be worthwhile to make them publicly available. You will get the most out of it if you understand it as a kind of practical tutorial of how to adapt NEdit to *your* needs.

2. Introduction

At first, I didn't want to write a manual at all. I don't like having to read long manuals to learn how to use something. The point is that individual editing needs and tastes are so different anyway that it is rather doubtful to try to invent anything that could serve all needs. The important thing is that you should be able to *easily* configure an editing environment to your needs.

I am convinced that useful things have to be easy to use. This is the way NEdit works and it is the way this macro package is written. You can start using it without having to learn some awkward handling. Just write your text, then select it with the mouse and invoke a macro to let it do the rest. Suppose, for example, that you want to give in a small 2×2-matrix as text-formula with round brackets and the entries a and b in the first row and c and d in the second row. It looks like this:

This is a small matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$.

You do it in the following way: Directly after "This is a small matrix" you type once[1] $. Then you type A SPACE B RETURN C SPACE D. Then you

[1] If "smart indent" is on.

select from a to d with the mouse and click the right mouse button. In the so called Window Background menu that shows up, you choose Matrices. A dialog menu pops up, where you see the different types of brackets and below there is a row of buttons labeled “OK”, “small”, “ltx”, “small-ltx” and “Cancel”. Select round brackets and press the button labeled “small”.

In the same way (almost) all the actions are handled, i.e. for equations, lists, environments, format, snippets and so on. That is all you need to know to work with these macros!

I think learning by doing is fastest. That is why there is a small example file named ‘example.tex’, where you can try a few things out for yourself in order to get you started.

Although you don’t need to read a manual to learn how to work with the LaTeX-MODE macros, there are several other reasons to have a manual. One reason is that things may not work and the manual can help to figure out what goes wrong.

Notice further that the example above needs the $\mathcal{AMS}$-LaTeX package. Standardly the macros fill in the $\mathcal{AMS}$-LaTeX commands for matrices and equations. That is because *I* am working with $\mathcal{AMS}$-LaTeX and this is just another reason to have this manual – the macros are suited to my needs and probably these will differ from your needs. So you may wonder, why

these macros are doing things this way and not that way. You may even dislike the way I do editing. Well, please recall that I did not write these macros for you. I have written them for me. These macros are only my individual customizations and what you need are your individual customizations for editing LaTeX-files. Hence there is this manual trying to document everything. Moreover, the source code of the macros is documented, so that you have a better chance to change things.

These macros are released in the hope that they are helpful. On the one hand there are things that you possibly will use unchanged or only slightly adapted, on the other hand you get a tutorial of how to write macros yourself. Of course, in that case you have to learn a bit about NEdit, about its macro language and about regular expressions (see 5.1.11.1, for example). But you can trust in the fact that I am a lazy type of guy. The amount of work to get things done with NEdit is considerably lower compared to other text editing tools.

3. Installation

Recall again that some macros require NEdit version 5.4. If you don't have version 5.4, yet, you can download it. NEdit consists only of a single binary, so there is no need of a special installation. Only execute it.

You are running MS Windows? If you have an X server, you can run NEdit on MS Windows 95, 98, ME, NT, 2000 and XP, together with MiKTEX. So do I, sometimes. I recommend, that you use the easy to install package from my home page. Although NEdit is an X client, it is fully working under MS Windows with any X server.

There are slight differences in the installation of the LATEX-MODE between Unix and Windows. Notice that for the Windows part, I assume that you are using the package from my home page.

3.1. Quick Guide (Unix/Linux)

Unpack the file 'latexmode-0.9.1.tar.gz'. The easiest way to test the LATEX-MODE is to create the environment variable NEDIT_HOME pointing to the 'latexmode-0.9.1' folder. Assuming that you unpacked the LATEX-MODE into your home directory, and that you are using bash, the command would be

simply:

```
export NEDIT_HOME=~/latexmode-0.9.1
```

Moreover, notice that the macros need certain key bindings that are contained in the file '.Xdefaults' in the 'latexmode-0.9.1' folder. So you need to issue the command:

```
xrdb -m latexmode/.Xdefaults
```

If you have done all of this, you can now simply start NEdit, best in server mode, to edit some LaTeX document, e. g.

```
nc latexmode-0.9.1/docs/example.tex
```

Notice that in some Linux distributions, `nc` stand for the `netcat` program. Then the NEdit-client was renamed to e. g. `ncl` or something else.

3.2. Detailed Installation

There is no automated installation process for these macros. As a matter of fact, I consider macros to be an individual thing. I would not even

know, how to do the right thing for everybody automatically. Hence I explain how things work. First, if you should not know NEdit version 5.4 yet, you should read the online help about NEdit's preference files under "Help → Customizing → Preferences" (perhaps you should read a bit about X resources, too), and also at least the start of the macro language section about the file 'autoload.nm'.

In the 'latexmode-0.9.1' directory, there are the three files 'nedit.rc', 'autoload.nm' and '.Xdefaults'. The file 'nedit.rc' contains the menu definitions (i.e., what you see under the "Macro" menu), and the file 'autoload.nm' contains macro code that gets executed when starting up NEdit. This code loads the LaTeX-MODE macros, which are located in files that are in the folder 'latexmode-0.9.1/nedata'. Finally, the file '.Xdefaults' contains important key-bindings for the LaTeX-MODE macros.

Now, for this to work NEdit must first find the files 'nedit.rc' and 'autoload.nm', and then, the macro code in 'autoload.nm' must find the actual LaTeX-MODE macros. Moreover, your X server must know about the settings for NEdit contained in the file '.Xdefaults'. Finally the LaTeX-MODE macros depend on various other data files all located in the directory tree under 'nedata'.

As shown in the previous section – assuming that you are on Unix/Linux

and unpack the LaTeX-MODE into your home directory – it is not hard to get all working. Only make NEdit aware of the preference files by setting an environment variable, make your X server aware about the settings for NEdit in the file '.Xdefaults' by including them into the X database with a single command, and you can start up NEdit and use the LaTeX-MODE macros already. It would be simple to write a script for this, but it is impossible to predict what *your real needs* actually are. So, I let you do these simple steps yourself to get you learning for more. If you can't, or don't want to, do the steps in the previous section yourself, I'm afraid to have to say this LaTeX-MODE is not intended for pure consumers. There are better options for you in this case.

Now, if you would like to include the LaTeX-MODE in a preference file of your own, you can simply import it with

```
nedit -import nedit.rc
```

Then check whether everything worked, and in order to make the import permanent, you have to save the defaults once ("Preferences → Save Defaults"). The NEdit help contains more about the location of the preference files. If you have an 'autoload.nm' file of your own already, you can add the macro code of the provided 'autoload.nm' to it. The important thing

is the global variable `$data` that must always point to the ‘nedata’ folder. There is another global variable `$windows` that decides whether you are running on a Unix/Linux system or on MS Windows.

Moreover, you probably want to add the content of ‘.Xdefaults’ to your real X resource file (depending on your system usually ‘.Xresources’ or ‘.Xdefaults’). Notice again that the key-bindings defined there assume that you are writing on a standard US keyboard layout. You have to adapt these bindings, if you should use a different layout. Notice that the LaTeX-MODE macros provide a way to write German texts on a US keyboard layout, like I do, that is close to writing the text on a German layout, while not losing the US keys for the commands (e. g. backslash and brackets), cf. 7.

3.2.1. Uninstalling

Notice that there is no complete automatic installation of the LaTeX-MODE after you have imported it into your ‘nedit.rc’ file. But there is the macro “Macro → NMacro → uninstall” that can remove a list of menu entries from your preference file. The list of menu entries imported by the LaTeX-MODE is provided in the file ‘latexmode-0.9.1/uninstall.txt’. The uninstall macro will save a backup copy of your preferences in the file ‘nedit.rc.bak’.

So, to uninstall the LaTeX-MODE, you have to run the macro above, and to exit NEdit. After restarting you still have to remove the imported syntax highlighting patterns. This can be done from the Syntax Highlighting Patterns menu with the button Restore Defaults. If you want to remove a whole language mode like BibTeX or DTX, Data or TeX-Log, you have to remove the highlighting patterns first, then exit and restart to delete the mode under "Preferences → Default Settings → Language Modes". (Sorry, this is an NEdit feature.)

Finally, you shouldn't forget to remove the keybindings defined in your X resources file for the of LaTeX-MODE.

3.3. Windows (Cygwin Port)

For this part I assume that you are using the installation package of NEdit for MS Windows provided on my home page, and that you are using the the start-up script 'start-nc.sh' provided in that package.

While the theory of installing the LaTeX-MODE is the same, there are slight differences having to do with the integration of NEdit into MS Windows. (If you should use the full Cygwin/X emulation on top of MS Windows, the installation is the same as explained before for Unix/Linux systems.)

Unpack the file ‘latexmode-0.9.1.tar.gz’. Then copy the files ‘nedit.rc’, ‘autoload.nm’ and ‘.Xdefaults’[2] to the NEdit home directory (cf. the installation of the Cygwin port – if you changed nothing it would be ‘C:\nedit’).

Open the file ‘autoload.nm’ and set the global variable `$windows` to `"true"`. (This is done automatically, if you have installed MiKTEX on your system.) Finally copy the folder ‘nedata’ to the NEdit home directory.

3.3.1. Remark on X Defaults

If you are a pure MS Windows user, you probably wonder, what this X defaults thing is about. NEdit is an application for the X Window system, the standard graphics system under Unix, which is based on the client/server model and is indeed independent of operating systems. There are many X servers for Windows. In order to use NEdit you’ll need some X server.

One of the advantages[3] of the the X Window system is, that you can have complete control over an application, for example what keyboard

[2] Depending on the X server the file name may have to be changed. Since there is a dot at the start, buggy Windows explorer doesn’t allow to change the name. So open the file in an editor and save it under the required name.

[3] There are also some disadvantages.

handling is concerned. Have a look at the '.Xdefaults' file. It would be new to me, if one has the same control over keys in MS Word. For a useful application see 5.1.11.

4. Overview

After installation you will find the macros located at the Macro and Window Background menu (which shows up when clicking the right mouse button). *Notice* that almost all macros are made language dependent, that means you will see them only when editing LaTeX-files.

You can customize the Macro, Background and Shell menu by going to "Preferences → Default Settings → Customize Menus" and then select the type of menu. One of the key definitions in the provided '.Xdefaults' file defines F12 to pop-up the "Macro Commands" dialog.

With the macros contained in the LaTeX-MODE you can

- run the TeX-compiler, a previewer (including forward and inverse search), and other tools (makeindex, bibtex,...),
- control your references and citations,
- display lists of the user-defined commands in, and the sectioning of, your file,
- define named snippets, for inserting in your text,

- define abbreviations for longer words or commands, that you need to write often,
- let partly written words or commands and environments be completed,
- insert matrices and equations in the easy way described in the introduction,
- insert all kinds of environments, formattings, sections, templates, mathematical symbols, . . . ,
- put mathematics automatically at separate lines and let closing brackets automatically be inserted.

The macro package contains also improved syntax highlighting patterns for LaTeX and BibTeX, and a small, not yet fully integrated, mode for Documented LaTeX Source.

Moreover, a Tcl/Tk script called *HelpSystem* together with an on-line help about LaTeX was included, since this could (eventually[4]) be more than only a browsable on-line help.

[4] It isn't so far.

In addition, since you will probably need to adapt the macros to your needs and taste, there are some macros for use with the NEdit macro language included. You can see them located at the Macro menu under “NMacro”, and “Macro → Help → Calltip”, which displays help about the built-in NEdit macro functions and variables.

Notice in this context that some of the LaTeX-MODE macros are of general use. So expanding abbreviations is certainly not only good for LaTeX-editing. Indeed, the same macro is used to fill in constructs like while- and for-loops of the NEdit-macro language while editing ‘*.nm’ files.

Independent from this, editing your files with NEdit will reduce the amount of your errors dramatically, because there is a capable real-time syntax highlighting[5] available.

[5] Not only keyword coloring, please!

5. Description of the Macros

In NEdit any macro needs a menu entry. You can put macros in the Macro menu or in the Window Background menu. However, it is sufficient that the menu entry consists only of a call to a function, where you define the functions in separate files. In version 0.8 almost all LaTeX-mode macros are functions defined in files, and the menu entries are merely calls to these functions.

For the future I will keep this menu structure. Of course, if there are new macros, I will need to add entries, but I won't change the menu structure as such. I describe now first the macros located on the Macro menu and then those located on the Background menu in the order of appearance in the menu.

5.1. Macro Menu

Notice that almost all macros are made language dependent, that means you will see them only when you are in the language mode for which they are defined. Apart from the core of the LaTeX-mode macros, which are of course only active if you are editing LaTeX-files, there are also macros

for DTX, Bib-files, Log-files, and some macros of general use. If NEdit is in LaTeX-mode, you should see the following entries in the Macro menu, where the entry “LTXMode” contains the core part of the LaTeX-MODE. (I’ll describe the macros for DTX in a separate section.)

5.1.1. spell-check (alpha)

The problem when trying to spell-check your LaTeX-document is, of course, that there are many commands in. Some spell-checkers, like Ispell, have a LaTeX-mode to avoid the worst, but so far this doesn’t work to well in my opinion.

I think the natural approach is simply to filter all of mathematics and other commands from your document *before* sending it to the spell-checker, so that the spell-checker only needs to check the real words, which is the spell-checkers native job. So, the main part of the spell-checker handling macro is a set of patterns (regular expressions) that are supposed to strip off (most) of the LaTeX-commands from your file.

Notice that you need either Ispell or Aspell installed on your system to run this macro and that the macro needs to find the spell-checker, cf. the global variable `$spellchecker` which contains the command to invoke

the spell-checker.

The spell-checking macro works for English and German texts (wherby German is chosen by switching with ALT+.). For other languages you have to edit the macro in ‘spellcheck.nm’ appropriately.

After the document was spell-checked, an assumedly mis-spelled word is selected and a dialog with a list of suggestions for correcting the word is shown. Select a suggestion and click on “Correct_all” (or on “Correct” to correct only this occurrence – then the macro jumps to the next occurrence if any). If there is no usable suggestion, simply correct the mistake manually and select the corrected word. Then click on “manual” to correct all other instances of the same mistake.

If a correctly spelled word is selected as mis-spelled (because it isn’t in the dictionary of the spell-checker) you can skip it. (The dialog entry “Skip” will only skip the selected mistake.) Or you can add it to the dictionary with “$\rightarrow$ Dict”. The button “undo” lets you undo the last correction, if you’ve done a mistake, and repeat it.

Finally, you can cancel the macro simply by closing the dialog.

5.1.2. NMacro

These macros are not really part of the LaTeX-MODE. They are there for your convenience. Say you would like to change a macro in the LaTeX-MODE. Then you would open the file 'latexmode.nm' and edit it. You needn't restart NEdit in order to get the new macro working. You can run the "NMacro → Load functions" macro (only visible when you are editing an NEdit macro file) by hitting the accelerator key ALT+0. Of course when the macros are reloaded all variables get reset, so you must define your main LaTeX-document again, for instance, but you needn't end the editing session.

Another useful feature is that you can display all accelerator keys currently defined by running "NMacro → show accelerators".

5.1.3. LTXMode → Main File

This small macro lets you define a main file. So if you are working on larger projects, the labels, commands, sectioning, running TeX and preview macros are more comfortable to use. Notice that for some features you have to define a main file first! There can be only one main file at a time. The main file is defined through the dialog. In the dialog there is

a list of file names displayed that either are currently open or have been bookmarked, cf. 5.1.9.

5.1.4. LTXMode → Run...

While it is easily possible to run TEX from a shell, e. g., under Unix you could set the environment variable

```
export TEXEDIT=nc -noask -line %d %s
```

However, many folks find it more convenient to run TEX from within their editor. All macros for producing and previewing your documents are located under "LTXMode → Run". Notice that these macros run with teTEX as well as with MiKTEX. The previewer is defined either as Xdvi or, for Windows, Yap.

If you have defined a main file, see 5.1.3, invoking "Run → LATEX" will compile your main file, otherwise the current file is compiled. The same holds for the preview macro. Invoking "Run → LATEX(+src specials)" will insert so-called source specials in your Dvi file, see 6. "Run → Next Error" will jump to the next error line and display a call-tip with the error message. By default warnings are not considered. You can change this by turning

debug mode on with the switch “Run → Debug on/off”. In the macro itself you can set a variable for the maximum number of error or warnings to be considered.

If the Dvi file contains source specials, executing the macro “Run → Forward Search” will open up the previewer and switch the display to your current cursor position, so that you needn’t search the part that you would like to preview yourself. For inverse search you must configure your previewer, such that it calls NEdit. Notice that you must use NEdit in server mode, i.e., through the file nc (NEdit client) so that there will not be opened second instances of files you have already open when the previewer tries to switch to them.

If you are missing something, such as running BibTEX, makeindex or a “build” macro, the provided macros have enough information how to do this yourself.

5.1.5. LTXMode → Insert

These macros simply insert environments, miscellaneous templates of LATEX-constructs, symbols, mathematical symbols and BibTEX entries in your text. If you miss more templates you can simply add them to the data

files.

Special note for the mathematical symbols. There are some editors that try to impress you with a kind of graphical user interface, i.e., they show pictures of math symbols that you can click on and the appropriate command is inserted in your text. This is a rather simple thing. Using the HelpSystem, see 5.1.7, for such things is more flexible than "Insert → Greek letters" may look like. (Requires Tcl/Tk!)

Invoking "Insert → quotation marks" by hitting CTRL+' will insert the right quotation marks – left or right, also considering whether you are writing German (ALT+., cf. 7).

"LTXMode → Insert → BG" contains the real function calls of the Background menu. The entries on the Window Background menu are only calls to the menu entries here (via macro_menu_command). The reason for this is that it isn't possible to define accelerator keys for the menu entries of the Window Background menu. So having it this way lets you define additional accelerator keys for these macros, if you wish to do so.

5.1.6. LTXMode → Structures

A common problem with larger files is for example that you need many text references, which are created with the command `\label` (or, if we think of pdf, with `\hypertarget`). Other examples are that you have many user-defined macros or environments in your text (`\newcommand` and `\newenvironment`, respectively).

Even if the file is not really large, you are likely to forget what labels you have already created and where they are. Of course, you can search them manually or let the TEX-compiler tell you that there are double labels and then you must search and change the double ones later on. Better is, we let your computer work. So these macros will search for labels and display an alphabetically ordered list of the names of your labels (hyper-targets, sections, user defined commands or whatever you like).

If a main file is defined the macros will search recursively through all your include files. Notice that the included file names are expected to be kept in the same folder as the main file. (At least I think so.)

Putting the cursor on a reference or a citation, invoking "Structures → Show Ref/Cit" will display a call-tip containing the label or bibliographic entry to which this reference or citation belongs.

Suppose you have a label on a lemma, and througout your text you

reference to this lemma. Later on you decide that the lemma should be a proposition. Then you would have to change all your references like "By Lemma. . . " to "By Proposition. . . ". Putting the cursor on the label after you've changed the environment and invoking "Structures → The/Ref (alpha)" will do this automatically through all your project files (if a main file is defined).

5.1.7. LTXMode → Help

There is an on-line help for LaTeX included. This is based on a Tcl/Tk-script of a Russian author, which is under GPL. For the details invoke this help system from "LTXMode → Help → About Help". You will need Tcl/Tk installed on your system. Of course, it works on MS Windows, too.

Another possibility to invoke this help system is from "LTXMode → Insert → Greek letters". This displays all Greek letters in the help system and you can simply click on them in order to insert them in your document. This insertion is done by an interaction of the Tcl/Tk-script and the NEdit-macro language on the other side. Read the file 'latex.tcl' for details. *Notice that "Smart Indent" has to be turned on and you have to run NEdit in server mode for this to work*!

One of the advantages of the above approach is that it is cross-platform and doesn't depend on a particular editor program. The help system, i.e., the '*.help' files are merely HTML-files with some additional directives. The pictures of the Greek letters are Png-files in base64 encoding (which is not required to do – I only didn't want a large collection of single Png-files. By encoding them you can store them into a single file). You can do such things simply yourself. You could extract whole formulas of you Dvi fils as Png-pictures in a automated way and store them for later use. You can then look through them with a Tcl/Tk script and a simple click on the formula would insert it into your document.

5.1.8. Editing → Complete Word

In order to help you with your writing, there are the word- or code completion macros. You write the start of your word, of a command, or of references or citations, and invoke the macro with F4. The macro looks through your current file *and through external completion files, that you can define*, and completes the word. A list of English and German top words and a LaTeX-version of the German top words is already included. There is also an example of a user-defined word completion file. Moreover, there is a

code completion file for LaTeX-commands included.

Notice that the decision whether a word- or a code completion is attempted is simple — if your word starts with a backslash, a code-completion is done, otherwise it is a word completion. A special case is when you are writing a reference or citation. For instance if you have written

```
\cite[Theorem 2.1]{Sm}
```

and the cursor is behind the Sm and before the closing brace, then hitting F4 tries to complete the Sm with matching entries from your bibliography. (This requires that you have defined a main file, cf. 5.1.3.)

Notice that for a completion file, a whole line is taken as completion, so that you can give in several words, where you need only to write the start of the first word in your file and press F4 to get the whole completion. If there are several possible completions, you can cycle through them by pressing repeatedly F4 or you press F5 which will show a list of all possible completions, where you can choose the wished one. If you have set a main file, user-defined commands will also be completed.

5.1.9. Bookmarks

Especially, when working with multi-file documents or projects, it will be comfortable to bookmark some of the files that you regularly need. This is independent from the ‘Open previous’-list inside NEdit that is constantly changing. It also allows to easily set the main file, see above. You can bookmark the current file simply with CTRL+B. You can see a list of the bookmarked files to open a bookmarked file or to delete an entry from the list with SHIFT+CTRL+B.

5.1.10. Help

This is the basic call-tip feature. Help about functions (parameters, return values and the like) are displayed in a small widget at the cursor position, similar to tooltips. You need tips files for this to work. Provided are tips for the NEdit macro language. More is available from home page.

5.1.11. Expander Macros

The Expander macros are located at the macro menu. At the entry “Expander” you can see the macros “Init”, “On”, “Off” and “Edit dat”. Below

these is a separator and the macros “lists on” and “lists off”.

In order to check if they work initialize[6] them with ALT+U or click on “Init”. Then give in `'e` and hit space. You should get an `\epsilon` now. To see all the already defined abbreviations click on “Edit dat”or hit CTRL+SHIFT+E. You can add your own abbreviations or edit the given ones. If you open the file where the abbreviations are stored with the *macro* (CTRL+SHIFT+E), edit and save it, the newly defined abbreviations will be present without re-initialization. Otherwise, you have to re-initialize.

The macros that you see there only initialize the Expander macro, turn it on or off and edit the data file, where the abbreviations for expansion are stored. The entries “lists on” and “lists off” turn the *lists completion macro* on or off. This is a small macro that automatically inserts `\item` or `\bibitem`, when you are in a list environment such as *enumeration* or *thebibliography*. It is not related to the automatic completion macro and can be turned on or off separately. There is nothing more to say about it.

What you can’t see in the menu are the macros that are doing the work, i.e., you don’t see neither the automatic completion macro nor the list

[6] If you would like to initialize the expander macro automatically, set the global variable `$C_on` at the start of the ‘autoload.nm’ file to one.

macro. This is because you needn't execute them, so I've hidden them.[7]

Execution of the completion macro is bound to the SPACE key and the list macro is bound to SHIFT+RETURN. That means, every time you hit SPACE or SHIFT+RETURN, not a blank or newline is inserted in your document, but one of these macros is run.[8] The binding of SPACE is done by way of key translations[9], whereas SHIFT+RETURN is bound as accelerator key. See 10.1 for the technical details about this. If you believe it is dangerous to bind the completion macro to SPACE you can and should of course change the key bindings.

The automatic completion macro is generally usable to define abbreviations that the editor expands. There are several variations of such macros around, one of them is the so-called *expander*. The expander is described in detail at the NEdit home page.[10] I've included some of its features in this version. To see the abbreviations that will be completed, click on "Edit

---

[7] This is simple. Set the menu entries for the macros to an undefined language mode (I chose "@keys", because the key-bindings have been re-defined for it), so you can never see them in the menu.

[8] Don't worry. These macros will not forget to insert a blank or newline in your document.

[9] For this reason you must add the contents of the 'dot_Xdefaults' file to your '.Xdefaults' file.

[10] Notice, that parts of the expander are external C programs.

dat". This will open the file 'expand_$language_mode.dat' in the 'ne-data' folder. You can simply edit this file, change abbreviations or add your own ones. Then save the file and initialize. Note that to run the macro you must first execute "Init" from the menu or define a short cut for it. Then the file is load to a string in memory, so that the file need not be opened every time you hit the space key.

The expander's features included are that you can recursively[11] expand abbreviations and that you can expand an abbreviation with a selection. Recursive expansion means the following: If you put an already defined abbreviation enclosed between |>...<| in the definition of another abbreviation, then this field will be recursively expanded. Moreover you can set the cursor at any place in the expansion by putting an empty field |><| at the desired position. Also, you can give in fields that aren't defined as abbreviations. After expanding, the first such field will be selected, so that you can type in your text for the field. Afterward you can jump to the next such field with "Next field" or define a short cut such as CTRL+X, for example. Expansion with a selection means that you have the following situation: abbrev␣word, where abbrev is the abbreviation to expand

[11] See the files 'templates_lat.dat' and 'persoenlich.dat' for one possible application of recursive expansion.

and `word` is some word. The cursor is directly after word. When you hit CTRL+SPACE now, then the word is taken as selection and the abbreviation is expanded. For example, you can define the abbreviation `bg` as

```
\begin{|>s<|}
|><|
\end{|>s<|}
```

Then `bg verbatim` will be expanded – well, I think you understand it. The `|>s<|` stands for the selection. But what when you would like to have more than one word as selection? No problem, select all the words you want behind the abbreviation and press CTRL+SPACE.

Notice that there is a small problem, because the expansion macro is bound to the SPACE key. So hitting space directly after `bg` in the above example will expand it before you can type in the rest. One solution would be to have a different key binding and you can change this to your taste. My solution is that with SHIFT+SPACE there will be inserted a blank without the call to the expansion macro.

This solution is also useful, because the automatic completion macros capitalize automatically the first word of a sentence, if you forgot to do so. This will cause no pain for TEX-editing. If there is an abbreviation, e.g.,

e.g., you normally adjust spacing with ~ or \␣, because TEX interprets a dot as end of sentence. Or otherwise, again, you can type SHIFT+SPACE after the abbreviation.

Moreover, you have the option to define abbreviations starting with a @. Then the definition part will not replace directly the abbreviation, but will be interpreted as shell command and the output of the shell command will replace the abbreviation. Example: Define the abbreviation `@date` as `date +%m-%d-%y` and it will expand to the current date.

5.1.11.1. Remark A remark on the correction macro above. This is done with the help of *regular expressions*. What is this? I think this is best understood by that easy example. Suppose you want to correct the first word of a sentence, if you forgot to capitalize it. You could of course easily search for a dot ending a sentence and also for a dot followed by a space, but what then. There are quite a few letters in the alphabet. Of course you could search for `.␣a` and then for `.␣b` and so on. But what you are really searching for is a dot followed by a space and then a letter. This is indeed a simple example of a search pattern. And such search patterns are described with the help of regular expressions. The regular expression describing this search pattern is `\.␣\l`, where the dot is described by `\.`

the space is described by itself and \l stands for some single letter. Now have a look at the source of the above macro!

5.1.12. Key Board

To ease editing TEX files, NEdit's standard key bindings are changed by the LATEX-MODE, for technical details about it see 10.1.

With these changed key bindings, the following will happen when editing a TEX file:

- Maths environments starting with $ are automatically placed at new lines and an additional $ is inserted. The cursor will be placed between the dollar signs.
- Maths environments starting with \[or \(are automatically placed at new lines. The appropriate closing bracket is inserted and the cursor is moved in between. You give in your equation that then is placed at a new line, i.e., you needn't type RETURN, this is done automatically.
- Generally typing in an open bracket produces the appropriate closing bracket and the cursor is placed between the two brackets.

- Typing in two underscores in a row will input sub and super indices.
- Hitting ALT+" inserts US or German quotation marks (depending on language setting, cf. 7).

ALT+4 will insert two dollar signs without a new line. Holding the ALT-key in addition to the normal key will give you the unchanged key behavior, e. g., ALT+SHIFT+[will insert a single open brace {.

Notice that, *if "Smart Indent" is turned on*, you can move through brackets or the dollar sign by just typing in spaces, i.e., typing in two spaces in front of a closing bracket (including those brackets preceded by an backslash) or the dollar sign will move the cursor and the last space behind the bracket or the dollar sign. You should type in punctuation marks, that should come after a bracket or a dollar sign, inside the brackets or the dollar signs – moving through the brackets/dollar will move punctuation marks behind the brackets/dollar automatically.

Moreover, if sub-/superscripts or another bracket follows superfluous spaces are removed. So you needn't move the cursor back, just go on typing!

5.2. Window Background Menu

The following macros are located on the Window Background menu that pops up when clicking the right mouse button. Notice that shortcuts for these macros can only be defined in the Macro menu (not the Window Background menu).

5.2.1. Theorems

Write your theorem, proposition or the like, slect the text thereafter and invoke the macro by clicking the right mouse button and choosing “Theorems” from the Window Background menu. Of course, what you need to get filled in depends on your `\newtheorem`-definitions. So you need the ‘theorem.dat’ data file. Or you could rewrite the macro to look for the definitions in the file you are editing. (You could use the “label”-macro for this, see 5.1.6.)

5.2.2. Equations

Only the $\mathcal{AMS}$-LaTeX versions of equations are supported, because they are clearly better than the (pure) LaTeX ones. The best way to see how

things work is to try the examples in the ‘example.tex’ file.

Here is a only short description. I’ve tried to make as much automation as possible. Normally you fill in only the entries of the equation even without alignment mark (&). If your single equation doesn’t fit on one line (in the TEX-output) you insert a newline (RETURN key) at the place where the equation should be split and write the next line and so on. Thereafter you select the whole equation and invoke, as usual, the “Equations” macro from the Window Background menu. Select single equation and choose “OK” or “No-number”. Then the macro will look first whether there is more than one line of input, otherwise there isn’t much to do. If there is more than one line, the macro knows that the equation must be split and needs alignment. The alignment is done in the following way: Each line processed separately. First the macro looks for a semicolon. If found the first semicolon is changed to a &. If no semicolon is found, the macro checks for a binary relation and inserts a alignment mark (&) in front of the first binary relation that is found in the line. If neither a semicolon nor a binary relation (you probably have to add some more binary relations to the search string) is found, then a & is inserted at the start of the line if it is not the first line. If nothing is selected, a template is inserted at cursor position.

5.2.3. Matrices

Just give in the entries of your matrix, separate columns with a blank space, if this is not possible, define a different separator, e.g., a semicolon.

Tables are supported only with a template under Insert misc-templ. This is because I don't need them usually. If you do you can take the matrices macro as an example how to program a macro inserting tabulars. If nothing is selected, a template is inserted at cursor position.

5.2.4. Lists

This is for quick inserting of various lists such as enumerations or descriptions. You needn't type in `\item`, that is done automatically. Only hit the short-cut SHIFT+RETURN for a newline.

5.2.5. Format and Sections

Various font sizes and also the verbatim environments, footnotes and quotation marks are inserted. This is mainly in case you forgot the appropriate commands and otherwise would need to look them up somewhere.

5.2.6. Snippets

This macro lets you maintain a collection of named snippets. There are several ways to insert often needed commands or constructs or just parts of text that you need to write from time to time.

For regularly used commands that you don't want to write again and again, the automatic completion macro (expander) or the word-/code completion macro are probably more adequate and faster. The snippets macro is more for (longer) parts of text that you only need to write time after time, so that you are more likely to forget about them. That is why you can give names to the snippets (instead of just a short abbreviation) and the beginning and ending parts of the snippets are shown together with the names in the list dialog menu from which you choose them.

5.2.7. Comments

These macros are for quoting or unquoting parts of your file. It is just a slight variation of default macros included with NEdit. The 'docstrip' macro will delete all comments in a selection or in the whole document.

The "ues2tex" macro has nothing to do with that. It is a macro that changes the German umlaute to TEX-style and vice versa. This can serve

as an example for other things, too. I just did not know where else to put it.

6. Source Specials

By *source specials* is meant, that line numbers and the names of your source files are included in the Dvi output. This information allows to relate pieces of output, i.e. paragraphs or formulas, with the place in your source file that created the output. Thus by clicking in the previewer window, you can jump to the appropriate place in your source file. This is called *inverse search*. Moreover, it is also possible to call the previewer from inside your editor to show the output related to the current cursor position. This is called *forward search*. In order to make use of inverse and forward search, you need a Dvi file with source specials and a previewer that supports these. Moreover, for forward search you need at least an editor that can call external applications.

By now any TEX distribution should contain a compiler that can produce Dvi files with source specials and a previewer that supports them. So all you have to do is to tell your previewer (probably either Xdvi or Yap) to

work with NEdit. For Xdvi you should add

```
xdvi.editor: nc +%l %f
```

to your '.Xdefaults' file.

For Yap, still assuming that you are using the installation package for NEdit on Windows provided on my home page, cf. 3.3, there is a small shell script 'yap-nc.sh' included in the LaTeX-mode. This must be copied to the 'cygwin\bin' folder, in order to use inverse search with Yap. Also Yap has to be set up appropriately, i.e., you find the command line to invoke the editor in the file 'yap-com.txt' also included in the LaTeX-mode.

7. Writing German Texts

You can toggle with Alt+. between writing English and German texts. If you hit Alt+. the mode is switched and a calltip is flashed shortly to indicate the current mode. How the special German mode came into being is explained now (for other languages similar things could apply, i.e., you could invent your own Dutch, French, Italian, Spanish,..., whatever mode).

The wish was to write German texts as if using a German keyboard layout, but at the same time I did not want to lose the easy access to the backslash and the various brackets that the standard US keyboard layout offers. As a matter of fact, TEX documents are created most easily with a US keyboard, since the author of TEX was an American. (One should also mention, that the German keyboard is very poor for programming.) So, the idea was to keep using the US keyboard layout for German texts, but changing the keybindings a bit inside NEdit. The approach that comes to mind first is probably to define ALT+; to insert 'ö', SHIFT+ALT+; to insert 'Ö', and so on. However, this does not feel like writing on a German keyboard at all. Thinking about it some more, I remembered that I could bind arbitrary keys to arbitrary macros (not only to the simple macro consisting only of a single insert-string action), and that unless I would going to write some comics, there are no German words that could contain two subsequent umlaute. Hence the rough idea was to define the key that in the US layout is the semicolon (and colon if the SHIFT key is held down) to insert 'ö', and to define SHIFT+; to insert 'Ö', and so on for the other umlaute and the ß, exactly as it is for the German layout (thus writing texts as if using a German layout). But if the preceding character is already a 'ö', then hitting the semicolon again, the 'ö' is replaced with a semicolon.

This was the first approach, and it felt already better than the solution with holding the ALT key additionally. However, it was still too rough. So, the macros had to become smarter. Now you get always the US keys when editing a command (i.e., a string preceded by a backslash), you get always a minus, if the preceding character is not a (non-capital) vocal, SHIFT+; inserts a colon if preceded by non-capital letters, inside inline maths equations you get the US keys (the macro is syntax highlighting aware), and so on – fantasy is not limited. I know these are very personalized configurations, but they possibly give you ideas for *your own* configurations!

8. General Remarks

These macros are only for use with NEdit. So you could say, that you and many others are using different editors and therefore it is bad to have something intended to help with editing LaTeX-files, which can only be used with a special editor. I'm not going to discuss about what is the best editor. Also, I do not try to sell anything nor to convince anybody of using NEdit for text editing.

My view is as follows: I had to choose an appropriate tool for the purpose of making editing LaTeX-files easy. I chose this editor, because that purpose can be achieved with it.[12] Perhaps there are other editors around achieving that purpose, too. But this doesn't make my choice wrong, because NEdit makes editing LaTeX-files easy. If you stick to your editor, because it helps with editing LaTeX-files, too, then just make your contributions.

Notice that I do not consider the set of macros to be complete or perfect. They are perhaps not even good.[13] For example although it could be done the LaTeX-MODE contains no macros for formatting your document (because *I* think this is useless) and there is no automated re-parsing of

[12] And NEdit is GPL, of course.

[13] Although there are people around that try to make money out of something lesser.

your documents or the attempt to implement too many automated things in general (because in my experience you can't get a program to do automatically what you want – it will mostly do automagically what you don't want).

Notice at this point again that I did not program these macros for you. These macros are mostly as I use them, so they have a strong tendency toward mathematics, or better that kind of mathematics that I need to write. There are other things added only to show how to get them. But these things need probably improvements. So if you are missing something or if you think some macros or even all are not good enough, do not hesitate. Add your own macros or improve the existing ones. There is no problem. You can freely distribute it. I've put the whole package under GPL, but the reason for that is only that I think that changing and redistributing shouldn't be done in a totally disordered way.

The intention of releasing such a set of macros is to give you a *starting point*, to give you examples of how to do it yourself. You needn't be afraid. The reason for choosing NEdit is that it is unbeaten for its combination of functionality and ease-of-use. It's macro language has a learning curve of about half an hour (or perhaps one hour, if you have never written programs yourself, as myself until I started with this here), including the

built-in editor library. Please understand these macros more as a tutorial of how to do it yourself than as ready end product designed specifically to your needs – how could anyone know about these, unless you tell him so, *and* pay him for implementing it!

9. Changes – what's new

For the latest, see the file changelog.

Version 0.8 was completely restructured. This means all macro code is contained in files from which the code is loaded. The menu entries contain only calls to functions defined in the files. This simplifies further maintenance a lot. Moreover there is more support for references and citations. Notice that there is no upgrade mechanism from previous versions. The format of the '.dat' files had to be changed, since I had chosen unused (for German) non-ASCII strings as separators. However these strings were just normal characters in the Russian encoding as a user from Russia told me. So now there are unused ASCII strings like 0x1c ('FS') used as separators.

In version 0.6 (was not made publicly available) there are mainly fixes and enhancements. The expander macro has been fixed and the possibility to use abbreviations to execute and get the output of shell commands was added. Notice that for avoiding compatibility problems between different operating systems, all data files have now all lower case names. The TEX-shell, i.e., the macros to run LATEX, has been re-written. If your TEX-compiler and your previewer support source specials, the shell macros

support them, too. The syntax highlighting patterns for LaTeX have been fixed and improved. The key-bindings have been changed to allow to write German umlaute on a US keyboard layout (in order not to lose easy access of the backslash). Perhaps this idea can be modified to be useful for other languages, too. Last more code completions have been added and there are some templates or example files provided.

In version 0.5 there are several improvements. The 'labels'-routine has been simplified and 'sectioning' has been included. The Word-/Code-completion macro has been corrected. The HelpSystem has been updated to version 1.4 and more examples have been included. There is also a corrected version of the LaTeX-highlighting patterns for NEdit. Moreover, the multi-file handling has been improved, see 5.1.3. There are also some smaller things, e.g., hitting the underscore-key two times in a row will insert sub- and super indices, hitting ALT+' inserts German quotation marks (can be changed to US ones). Also, this documentation was updated.

Notice that some key bindings have changed and that there are changes in the expander macro, see 5.1.11.

In version 0.3 there are only minor changes to version 0.2. The spell-checker handling macro is slightly corrected. The assistant system can give now a first vague impression of what it should become. Have a look at “LTXMode → Insert → Greek letters”. There is a technical section in this manual to explain how this is done, see 5.1.7. Moreover, there is an illustration of the use of the recursive expansion feature of the expander macro, see 5.1.11.

In version 0.2 there are some more macros of general use included. For example the word completion macro (that can be used for code completions, too) and an alpha version of a macro that handles an external spell-checker like Aspell or Ispell (which you must have installed on your system). Moreover, the smart indent macros (no good word for what they are) have been changed almost completely. I have changed the key-bindings, so that to every opening bracket there will be the appropriate closing bracket inserted. If you don’t like it, you can hold the ALT-key down, or delete the key-bindings (in the X defaults file).

In addition, there is a macro included, that saves the current cursor position in a file, when you close it, so that you can go on editing at the same position next time. This is also done by changing key-bindings, i.e.,

the macro is executed when hitting CRTL-Q or CTRL-W, before the file is closed.

Finally I decided to include an alpha version of a kind of assistant system (or it could become something like this), although it is far from completed. The reason is that I don't have much time (actually not even enough time to update this manual), so I couldn't bring it to the state that I would like it to be, but it should be clear from the provided macros, how this can be done.

Make sure to check out 5.1.11. This macro has been renamed and the data files that it uses have been renamed, too. It is possible now, to include several data files for abbrevations and corrections by simply adding their names to the arrays at the start of the initialization, see the "Expander → Init" macro.

Notice, that short-cut keys have been changed, too. They are listed in the file 'key_bindings.txt'. See also the X defaults settings for NEdit. Finally this manual isn't up-to-date, thus you may need to read the macros themselves.

In version 0.1, the LaTeX-MODE runs with MiKTeX. For Windows 95, 98 and ME you should use the binaries from my home page. New macros

are for running LaTeX, previewers, showing help files and inserting BibTeX entries. The automatic completion macros have been improved and the ‘expansion with selection’-feature of the so-called expander was added. There were also a few slight improvements of other macros. In order to ease installation, all (proposed) accelerator keys were shifted to the file ‘dot_Xdefaults’, so you have to define key bindings for yourself. I did this in order to avoid interferences with accelerator keys that you may have already defined when importing the NEdit-preference file.

10. Technical Notes about NEdit.

These sections cover technical remarks about key bindings in NEdit and the so-called call-tips. Note again that the LaTeX-MODE macros assume that you are using a US keyboard layout. If you are using a different layout, you will usually get unintended results, so you have to adapt the key-bindings to your needs.

10.1. Key Bindings

There are two different ways to define key bindings in NEdit. One is by way of key translations in your X resource file, the other is by defining accelerator keys in the macro menu and the shell menu. Both ways of defining key bindings are quite different. The key translations cannot be changed on the fly, i.e., in order to change them you have to edit your X resource file, to run `xrdb` and to restart NEdit. So, if in the X resource file keys like SPACE, RETURN or the bracket keys are bound to macros (as is done in the LaTeX-MODE) and these macros shouldn't be in the menu definitions, NEdit tries to invoke them, when these keys are pressed and nothing will happen. On the other hand, the definitions of accelerator keys

in the menus can be changed on the fly and they overrule the key translations. Moreover, these accelerator bindings are only active, when the respective menu is active. This means that you can bind the same accelerator to a lot of macros, if these macros are for different language modes. Thus the decision seems to be clear: Forget about key translations and use the accelerator bindings.

However, why are these bindings called *accelerator* bindings? Can they not be used to bind SPACE, RETURN or the bracket keys? The answer is: Yes, they can, *but* you should not do so, because these bindings are indeed only intended for accelerator keys. The problem is that their definition is global to the window. You can see this, if you bind for example the SPACE key to a macro by an accelerator binding in the macro menu. Then invoke the incremental search bar (CTRL+I) and try to give in a blank. This won't work, because the macro bound to SPACE will be executed. So, binding normal keys to macros will make the incremental search bar useless (if you bind RETURN to a macro by an accelerator binding, you can't even search incrementally). Moreover, key translations are more flexible in the sense that you can bind macros to *X events*. See also the short tutorial about key bindings that I made available at my home page.

These are the reason to use both type of key bindings. All the bindings

of normal keys such as SPACE and the bracket keys to macros are done by way of key translations defined in the X resources file. There is also a definition included in the X resources file to get their original behavior. Simply hold additionally the ALT key down. Only for the SPACE key you have to hold a SHIFT key down.

But, of course, you *can and should* change the key bindings to your taste and needs! You probably even have to if you are using a non US keyboard layout! Notice that I even didn't include accelerator keys for most macros. So, if you don't want to invoke the macros through the menu each time, you have to define accelerator keys for them, which is simply done in the macro and window background commands menus ("Preferences → Default Settings → Customize Menus").

10.2. Call-tips

In order to implement a TEX-shell inside NEdit, we have to display error messages of the TEX-compiler. There is no buffer concept in NEdit. We could open a new editing window only to show an error message or we could show the message in a dialog box. The later choice is bad, because it will block macro execution and we have no control of the position of the

box. It will probably obscure the editing area. The former choice could work, if we open a new window through the NEdit-client in order to control the size and the screen position of the new window. However, then you have to switch between windows (it is not clear where the editing window is and where to open the message window) and worse we run into a timing problem (is the new window already created, before we insert the error message – this would be ensured when opening the window not through the client but with the open or new window action, but then we cannot control the size of the new window, i.e., it will have the default size of an editing window). Hence it is natural to use call-tips to display the error message.