THE macro contributions from users

Abstract:

You will find here an overview of the collection of useful macros for THE (The Hessling Editor, by Mark Hessling). Additionally you will find some other general information concerning the writing of THE macros.

The macros listed here have been contributed by users to share them with other users. Feel free to check these out. Interesting concepts and valuable examples can be seen here.

In general the macros are free to use, some are even in public domain.

Mostly the contributions came from the mailing list THELIST. This mailing list is intended to discuss all things related to THE. You can subscribe to that list via the CITES LISTSERV web frontend of the *University of Illinois at Urbana-Champaign* (UIUC).

Table of Contents

Author List

The collected macros have been written by various authors.

- *Jonathan Beach*
- *Lucio Chiappetti*
- *Ian Collier*
- *Carl F. Cotner*
- *Th. DeBoo*
- *Roger Deschner*
- *Agustin Martin Domingo*
- *Pablo García-Abia*
- *Florian Große-Coosmann*
- *Shintaroh Hori*
- *Vance Johnson*
- *Sean King*
- *Scott Mattes*
- *Wesley J. Metzger*
- *Klaus H. Ringhofer* (+26.12.2002)
- *Mark J. Short*
- *Sérgio Augusto Simão*
- *Homer Wilson Smith*
- *Franz-Josef Wirtz*

Details on Macros

Back to Table of Content

AM-LaTeX extension

Author:
: Agustin Martin Domingo

Download:
: theamlatex25b.tgz (14129 bytes)
: theamlatex25b.zip (35233 bytes)

Restriction:
: THE 2.5 (and probably versions down to 2.2) or KEDIT

Description:
: These macros allow you to insert LaTeX-phrases in an intelligent way. Either single commands, begin-end-phrases or complicated environment like things are covered. The macros are used as a set of key definitions, each offering a thematic menu for further selection. So, with two key strokes you are where you want to be.

  These macros can be used standalone or interfaced with FW-TeX extension.

License:
: GNU General Public License

Back to Table of Content

all extended

Author:
: Franz-Josef Wirtz

Download:
: allx.the (1601 bytes)

Restriction:
: needs GNU grep utility

Description:
: You can use `allx` when you need more power for selecting the wanted lines. You can use all regular expressions available via the `grep` utility. The lines selected by grep occur in THE in the same manner as a selection with all would have selected them.

  This macro can be found within FW-util package or FW-macros too.

License:
: GNU General Public License

Back to Table of Content

column macros

Author:
: Vance Johnson

Download:
: col.zip (4487 bytes)

Restriction:
: AIX/THE 2.4, XEDIT

Description:
: Several utilities to handle files organised in columns. See col.readme

License:
: GNU Public Software License

Back to Table of Content

count lines

Author:
Scott Mattes
Download:
count.the (1587 bytes)
Restriction:
unknown
Description:
This macro emulates the XEDIT count command. It counts the number of lines where a specified text occurs within a specified region.
License:
public domain

Back to Table of Content

CP-Macros

Author:
Carlos R.A. Pfitzner
Download:
from author's site: http://roberto.dnsalias.com:8880/downloads/the/
from contrib site: pfitzner_the_macros.zip (18.4 kB)
Restriction:
unknown
Description:

- `count.the`: count /string/ bet zones
- `db.the`: del blank lines
- `end.the`: file/quit one/all files
- `ex.the`: extract anything
- `icaps.the`: Initials Capital Case
- `kk.the`: repeat mods on curline
- `lp.the`: place arg on prefix area
- `nodup.the`: del duplicate lines
- `prefutil.the`: prefix UP/low/Cap-Case
- `wdel.the`: del word/end-of-word

License:
GNU general public license

Back to Table of Content

filelist

Author:
Shintaroh Hori
Download:
kl.zip (60701 bytes, version 3.0) and kl.txt (short info)
Restriction:
none
Description:
KL macro is a kind of 'File Lister'. It displays a list of files and directories that meet a given file specification like DIR subcommand of KEDIT and THE editors.
The list is displayed in a KEDIT(or THE) file called "`KL list`" from where you can issue OS/2 commands and KEDIT(or THE) commands against listed files.

There's plenty of documentation included. The zip contains also some other helpful tools (klring, kldir)

License:
IBM License Agreement for OS/2 Tools (free use)

Other references:
Finally, you could also interface to an external program like `fulist` by ***Ian Collier*** or `fl` from OS2FL.

Back to Table of Content

filename completion

Author:
Pablo García-Abia

Download:
complete.the (20028 bytes, version 1.8)

Restriction:
needs `ls` utility (e.g. from GNU-filetools)

Description:
Macro to complete filenames in the command line with a single keystroke, whenever the first command typed is one of the following (abbreviations allowed):

```
XEDIT    FFILE    FILE    MACRO      LS
THE      SSAVE    SAVE    OSREDIR    DIRECTORY
EDIT     GET      PUTD    STATUS
```

Otherwise, an alternative command is executed. If *alternative_command* is `HELP`, this help is shown.

To use it, define a key in the following way:
`DEFINE` *`keyname`* `MACRO complete` *`alternative_command`*

For example:
`define TAB macro complete sos tabfieldf`

In this example, the TAB key will complete filenames when typed IN the CMDline (after XEDIT or THE commands). Otherwise it will execute `sos tabfieldf`.

Filename completion:

Let's assume we are typing in the CMDline anything after one of the allowed commands. The following substitutions are made before proceeding to the filename completion itself.

A double equal sign (`==`) is substituted by the whole filename (without path) of the file being edited.

A single equal sign (`=`) in a given field (delimitied by the selected separator) is substituted by the corresponding field of the filename of the file being edited.

The separator is taken as the character preceding the equal sign unless it is the first in the string. In that case, the character following the '`=` ' is taken as separator. The dot '.' is used as a separator in ill cases.

The characters `=/` or `=\` (depending on the OS) are substituted by the path of the file being edited.

More sophisticated completion may be implemented in the future for other commands, eventually.

License:
unknown

Back to Table of Content

filter

Author:
: Ian Collier

Download:
: filter.the (1648 bytes)

Restriction:
: obviously needs a utility which can act as filter

Description:
: Filters the region from the current line to the target.

  example: `filter 5 fmt -72`

  format the next 5 lines of text with Unix `fmt` utility

License:
: unknown

Back to Table of Content

flow

Author:
: Roger Deschner

Download:
: flow.the (7619 bytes)

Restriction:
: requires at least THE2.7, works with THE 3.2b1

Description:
: This THE macro aligns two or more lines of a text-type file being edited (such as a NOTE). It tries to place as many words as possible on a line, within the right margin as defined by `SET MARGIN`.

  Paragraph breaks are respected, and can be either style - indented with space or tab characters, as this paragraph itself is, or a completely empty line, as the following paragraph is. A side effect of this is that a set of lines which are indented are protected from flowing; this is useful for examples.

  This paragraph is the other style. Note that text containing tab characters for any purpose other than to start a paragraph or indent an example, may not appear as you intended after flowing.

  If a word is encountered that is longer than the margin, it is left by itself on a line, without truncation. This is most frequently encountered with URLs.

License:
: GNU Public Software License

Back to Table of Content

formate

Author:
: Klaus H. Ringhofer

Download:
: formate.the (3230 bytes)

Restriction:
: might have problems on HPUX (check it out)

Description:
: formats the current or next paragraph

A paragraph is defined by consecutive text separated by empty lines. The macro formats the whole paragraph, irrespective the position of cursor within the paragraph. The paragraph indentation is not recognized. The macro removes also subsequent spaces. The text is broken at word ends. Word delimiters are always REXX's word delimiters, not THE delimiters (THE can toggle the way words are delimited). Filters the region from the current line to the target.

License:
Donated to the public without any warranty of any kind.

Back to Table of Content

FW-base system

Author:
Franz-Josef Wirtz
Download:
thefwbase27a.zip (163kB)
Restriction:
THE 2.7
Description:
This is the core of the FW-macros. The main concept is to

- add online help for keywords, THE commands and for context,
- allow file type specific differentiation,
- implement a configurable, extensible, simple yet powerful menu system,
- give improved support when dealing with lots of files in the ring (only when fwutil-package is installed too).

One key is, that it is possible, to have files of different style in the ring and each file can be handled in a different way (e.g. key definitions, main help file for keyword lookup, word wrap...). Since several of the extensions are beased on files also present in the ring theses files are *hidden* when switching through the ring.

To see, how it works on a distinct file type you might want to check for FW-tex extension. Even when you don't use TeX, you probably can learn by example, how you could setup a system for your preferred file type.

License:
GNU General Public License

Back to Table of Content

FW-client/server

Author:
Franz-Josef Wirtz
Download:
thefwcs27a (14kB)
Restriction:
Only on OS/2 (needs `rxqueue` for named queues). Should run also woth earlier THE-versions.
Description:
By establishing a named queue before starting THE the macros running in THE and client command batches can utilise this queue for communication. So, you have one command batch (`thes`) to initiate the queue and start THE. This batch acts as a server. Another batch (`thec`) now can be used to act as a client. By passing a filename to this client it sends a command to load this file to the queue. Within the server the queue can be processed with a THE-macro (`pq` - process queue) and thus the file can be loaded into the editor. In general, arbitrary commands could be sent to the server. Currently the client only sends the `edit` command.

The process is not fully automated to avoid slowing down of THE by infinitly waiting for commands in the queue. Instead, when you have dropped a file on the client, you process the queue with a THE macro manually. Thus, you have the control over what happens and you don't get interrupted when you won't like to. When you started several clients, you can load all files passed with a single `pq` -call.

One simplification has been made: when you drop a file onto the client icon without having the server started, the client makes up for this. This way, you only worry about the client.

It's obvious that you can use this concept via simple drag and drop as well es via the command line in a shell window.

License:
GNU General Public License

Back to Table of Content

FW-delete/undelete

Author:
Franz-Josef Wirtz

Download:
thefwdel27a.zip (20kB)

Restriction:
THE 2.7, long filenames

Description:
This portion of the FW-macros handles the deletion and recovering of text in THE. The way how cursor positioning is handled is different to the way in plain THE. It's more close to WordStar-compatible editors. The undelete feature of block marks maybe useful even when other parts are not to your taste.

The main difference to THE is, that deletion at eol or backspace at bol continues to the next/previous line. The macro 'recoverline' just uses THE's feature of recover. The macro 'undelline' uses an undelete buffer for only one line, when a line is deleted with 'deleteeol' or 'deleteline'. Thus, you can undelete the same line several times.

The 'deletechar' macro works different when used at the command line of a dir file. If the command line is empty, it places the operating system command to delete the file at current line to the command line - without execution. You have to do it yourself.

Normally, these macros should be assigned to keys and used within text area (some work also in command line;-). Within FW-macros there are two ways, depending on wether you use FW-macros in their plain form (WordStar-like) or in compatibility mode (THE/XEDIT/KEDIT-like). In general you are completely free to use your own definitions.

This portion of FW-macros should work on WIN32, OS/2 and X11. It is possible that it works on other systems too.

License:
GNU General Public License

Back to Table of Content

FW-folding

Author:
Franz-Josef Wirtz

Download:
thefwfold27a.zip (23kB)

Restriction:

none (should run on THE 2.4 too)

Description:

This is a package, which tries to implement a way for folding text in a general way, demonstrated at a special example of the `the.man` document file, coming with THE. There are three macros, one which can prepare the text for folding (and folds distinct levels) and two other, to show either more or less details.

So, what you will see after applying the first macro (`fold`) is a list of the chapter headings. After selecting one line, the macro `more` gives you more details. This maybe either the text of that chapter, or, when the chapter has lots of sections, a list of section headings. So, a second `more` may be necessary to give the details for say a special section. The macro `less` will take you back to what you've seen before.

The macros `less` and `more` should be general, while the macro `fold` is specific for a type of text. One has to describe, how a text can be separated into chapters and sections and what should go into the index lines.

The package contains also a document, which describes my concept, the pluses and minuses. So, I hope, I will read some suggestions or critics or own experiences or other approaches to the idea of folding texts. Any discussion is very appreciated.

The macros are tested on OS/2 warp 4 with THE 2.4 and THE 2.5 (beta). They should run on other OSes too.

License:

GNU General Public License

Back to Table of Content

FW-locale

Author:

Franz-Josef Wirtz

Download:

thefwlocale27a.zip (13kB)

Restriction:

implementation for X11, usable on UNIX too.

Description:

This macro can be assigned to several keys to allow for inserting accented characters. You can define either one key to act as a compose character or you can define a group of keys (usually the accents) to act as dead keys. A third possibility is simply define one key to launch a menu for further selection.

The way it is realised by now works on unix with the iso-latin-1 codepage. Other operating systems and codepages need different macros with different chars. The macro itself contains the chars to be inserted. So, if you want to change the macro you need an editor which can insert the correct chars for you.

Just note, that on X11 you have many ways to influence the way characters are inserted in your text (see documentation of THE). This macro supplies a simple way when other ways seem to be too complicated or when you don't have X11.

These macros can be found within FW-util package or FW-macros too.

License:

GNU General Public License

Back to Table of Content

FW-macros (complete)

Author:
Franz-Josef Wirtz, part of TeX-stuff Agustin Martin Domingo
Download:
thefw27a.zip (340kB)
Restriction:
THE 2.7, OS/2, WIN32, X11, LaTeX-stuff currently only for OS/2
Description:

FW-macros are (-:

- a demonstration of the versatility of THE
- a good test suite for THE ;-) and Regina
- a huge sample collection
- a sportive challenge in writing more code in macros than the exe has bytes
- a miracle for all who tried to use them and stumbled at the first step
- an invitation to others who write macro to share them with others
- a never ending try to get more discussion into THELIST
- a practical way to learn REXX for me
- a recreational task for me
- an useful configuration of THE for me when editing LaTeX-files.

Ok. Basically FW-macros are a **concept**. That concept turns THE into a completly different behaving editor (more or less similar to WordStar). Additionally it introduces a distinct way of free configurable menus, help at your fingertips and some other useful features, especially when handling multiple files in the ring. It can make every day usage of THE more simpler, introduces more visual action (e.g. with blocks) into the editing. Since I use THE I never used the prefix area and only sparingly the command line for editing.

Since I'm told that most users of THE want THE to behave as close to XEDIT as possible I once introduced a compatibility profile instead of the standard FW-profile, which still leaves lots of things (e.g. keyboard mapping) like it is, but changes the way things work and: enables all the good things in FW-macros. But I never heard any comment on this... ;-)

Why would you use FW-macros?

- If you like WordStar-keymapping and block marking, it's your lucky day.
- If you don't like WordStar-keymapping you may be fine too, when using the compatibility profile instead of the FW-profile. **That's your challange** just tell me, what's going wrong with my macros ;-)
- If you want to maintain your own keymapping/profile you can be fine too by simply picking up what you need and adding it to your profile. (ok, not that easy, but lots of macros don't rely on each other, but some do). You can stay with your habits, and just use the added functionality. (Ok, maybe some habits have to get sacrificed).
- If you have lot's of unbound keys, with FW-macros that time has gone. Nearly every combination does something.
- If you're getting old and forgetting things you may profit from menu selections where you see what you can do.
- If you're getting tired on switching between command line, file area and prefix area you will win with FW-macros. You can do almost everything with simple keystrokes. So only tasks like locating targets or so need to be done on the command line.

Some highlights:

- If a command went wrong, just hit the help key to get help on the command.
- If you don't know the syntax of a THE-command, just type the command at the command line and hit the help key. The help key works different depending on context.
- When you edit source files like rexx/the/c/tex or whatever else you can use or setup a keyword lookup in the available docs for the phrase beneath cursor.
- You can setup/use a menu system to make selections of actions or configurations. The menu system can vary depending on the file type.
- When you edit several files (like a c-code project) you can save the status of your session (cursor positions, files loaded, bookmarks set) with a single keystroke - and reload it automatically on next start of the. You don't have to type any more long filenames.
- When you have several files in the ring you can get an overview of all files with there specific settings and simply pick a file by selecting it with cursor up/down and finally hitting enter.
- You can make more use of the dir file, have all directories clustered together, sort it in different order or just hit enter to edit a file or enter a directory or start a command with the file.
- Block operations act visual. Use line, box column or word blocks by simply marking the end points in the text. You can move blocks around and use simple indent or exdent of blocks, even for box blocks!
- With the am-macros included you can make intelligent inserts of TeX-phrases into your code. For other source languages you use on a regular basis you could use the same technique.
- Move around more easily. Have scrolling left/right/up/down at hand, jump from paragraph to paragraph or find the block somewhere in your many files in the ring.
- When you edit several files at once you can locate a target in all files in the ring.

Ok, that's for now. Just pick up the macro set and see wether it fits into some of your needs. I would appreciate any comments. Especially from those, who (want) use the compatibility profile.

One known bug: the retrieve doesn't work any more. Any ideas...?

This package contains all the other subsets. This complete package is split up into subsets to ease the selection of special topics if you don't want use the confusing heap of the monolith FW-macros. These subsets can be used without using FW-macros in general.

Anyway, as FW-macros is not a collection of macros but a different approach to use THE some of the macros show their full power only embedded within FW-macros, and lots of them (not distributed as subsets) are only useful, when used as the complete system.

License:
GNU General Public License

Back to **Table of Content**

FW-mark/clipboard

Author:
Franz-Josef Wirtz
Download:
thefwmark27a.zip (29kB)
Restriction:
THE 2.7, file system with long filenames
Description:
This portion of the FW-macros handles the usage of marked regions, undelete buffers and a clipboard in THE. The way how blocks are marked is slighly different to the way of marking in plain THE. It's more like marking in WordStar-compatible editors. Normally, these macros should

be assigned to keys. Within FW-macros there are two ways, depending wether you use FW-macros in their plain form (WordStar-like) or in compatibility mode (THE/XEDIT/KEDIT-like). In general you are completely free to use your own definitions.

The clipboard is a paste buffer, which can be used to collect and hold text for later use. It's a distinct file managed by these macros. Thus, it can't interface to the operating system clipboard. The clipboard persists the session (as long as its directory is not on a ramdisk).

License:

GNU General Public License

Back to Table of Content

FW-move/scroll

Author:

Franz-Josef Wirtz

Download:

thefwmove27a.zip (24kB)

Restriction:

THE 2.7, file system with long filenames

Description:

This portion of the **FW-macros** handles the moving of cursor or text in THE. The way how lineends are handled is different to the way of moving in plain THE. It's more like moving in WordStar-compatible editors - the cursor will wrap at eol/bol to the next/previous line.

License:

GNU General Public License

Back to Table of Content

FW-setup tools

Author:

Franz-Josef Wirtz

Download:

thefwset27a.zip (34kB)

Restriction:

THE 2.7, color version

Description:

This portion of the **FW-macros** handles the configuration of THE. Two macros can handle state variables within THE which can be toggled or cycled through. One macro can toggle the states of environment variables defined by the user. This environment variable can serve as a global extension of THEs variable pool for own purposes in own macros (Note: the actual version of THE has the `editv` command).

Another set of macros describes a visual way in setting up THE. The aspect covered is the color scheme. The color macros serve as an example for intercepting keystrokes to have distinct effects different to the keys in normal context. They also show, how another macro can serve as a state information storage place. When you call these macros from your profile, you will always have the same setup for every session.

Some other macros query distinct aspects of the current settings or files in the ring.

License:

GNU General Public License

Back to Table of Content

FW-TeX extension

Author:
: Franz-Josef Wirtz

Download:
: thefwtex27a.zip (50kB)

Restriction:
: THE 2.7, OS/2

Description:
: This portion of the FW-macros covers the support for writing and processing LaTeX-sources. It is an example for the implementation of a language specific mode to THE. THE behaves different, when a TeX-source is edited. Main features are:

- mode specific menu tree for processing the source,
- keyword lookup in online reference,
- mode specific key definition with lots of intelligent insertions of LaTeX-Phrases (requires AM-LaTeX extension).
- simple conversion of columnar data sources into LaTeX-tables.

Although the current implementation is specific for OS/2, it should be possible to adopt it for situations on other operating systems.

License:
: GNU General Public License

Back to Table of Content

FW-utilities

Author:
: Franz-Josef Wirtz

Download:
: thefwutil27a.zip (48kB)

Restriction:
: THE 2.7, OS/2, WIN32, X11

Description:
: This portion of the FW-macros covers some common utils which should be usable independent from the other FW-macros and useful for 'normal' THE users too.

The utils cover the following topics:

- get information about the current THE session (files in the ring, their states, bookmarks defined),
- ease the use of THE with several files in the ring (locate, save, exit and pick),
- make more out of the dir-file (various sortings different to the sorting within THE)
- format floating text
- see only what you need with an extended all command
- benchmark the REXX-interpreter embedded within THE

With some of these macros you will get even more power over THE when used within FW-macros.

This portion of FW-macros can be used on systems like OS/2, WIN32 and unix and maybe others, since there is no platform specific code used or at least I've tried to handle platform specific situations. It's the beginning of making FW-macros available to other platforms than OS/2.

License:
: GNU General Public License

Back to Table of Content

join_lines

Author:
Sean King

Download:
from author's site: **join_lines.the**
from contrib-site: **join_lines.the** (788 B)

Restriction:
none

Description:
Make one line out of the focus line and the line following. Leading spaces in the following line are ignored. If the cursor is beyond the end of the line the join occurs at the cursor position. Otherwise the lines are joined with a single space between them.

License:
undefined

Back to Table of Content

kdraw

Author:
Shintaroh Hori

Download:
from author's first updated site [IBM OS2EWS ftp]: **kdraw.zip**
from contrib-site: **kdraw.zip** (16kB)

Restriction:
none (but tested only on OS/2, DOS)

Description:
KDRAW.ZIP includes KEDIT and THE macros to draw box characters by cursor move keys just the same manner as DRAW.E macro for EPM editor for OS/2.

Since KDRAW macros are written in KEXX (resp. THE-REXX), they will work under KEDIT of any platforms or under KEDIT-clone (e.g. THE) though I have not tested the macros fully on other environments other than KEDIT for OS/2 and for DOS.

KDRAW macros have better functions than DRAW.E macro as follows;

- Any characters can be entered in a file area while in drawing mode.
- Up and down of a drawing pen is controlled by Ctrl-F1 key which would make it easier to enter characters while you are in drawing mode. (In DRAW.E macro, it is done by INSERT key.)
- You can select either single-line type or double-line type to draw box characters by Ctrl-F2 key while in drawing mode.

License:
see included IBM License Agreement for OS/2 Tools (basically: free with no warranty)

Back to Table of Content

lmatches and mcmatch

Author:
Sean King

Download:
from author's site: **lmatches.the** and **mcmatch.the**

from contrib-site: lmatches.the (1314 B) and mcmatch.the (3271 B)

Restriction:
none

Description:
Find the matching token, or matching bracket. Based on a KEDIT macro by *Michael Chase*. With `[]{}()<>`, works like `CMATCH`. Have the cursor on the first character when matching a multi-character token:

- `do end`
- `#if #endif`
- `if fi`
- etc.

Assign to a key; the key currently used for `CMATCH` is a good choice. The token pairs are made available by executing the LMATCHES macro.

License:
undefined

Back to Table of Content

macs

Author:
Florian Große-Coosmann

Download:
from author's site: macs.zip
from contrib-site: macs.zip (175 kB)

Restriction:
none, used on unix, Windows, OS/2

Description:
My macro package is an extension of *Willi Lange*'s kedit macros of 1985 ff.

Benefits:

- Different behaviour for different source files.
- Better comment handling
- Not coloured extremely. If you need a syntax highlighter you should better change your programming style, not the displaying tool.

Big things you should try on the command line:

- `ant [args]` - invoke ant with optional arguments and redirect the output to the file "err"
- `make [args]` - invoke [n]make with optional arguments and redirect the output to the file "err"
- `class` - create/change a comment for a class. Other comment styles are available, too.

IsCStyle	IsHTMLStyle	IsJStyle	IsPlStyle				
a-enter	a-f2	a-f3	a-tab	a-x	addblock	ant	append
box	boxfill						
c-c	c-f	c-minus	c-plus	c-tab	c_comment_eol	caseword	ccomment
chktabs	class	clearkm	colors	comm	commands	comment$.prf	comment
compile	count	curr_cp					
delbegin	delfield	delword	demo	displayit	dmake	dupblk	
endltabr	enterkey	extrtest					
f6	fgrep	file_tex	fillbox	fkeys	flow	func	

grep	help						
home							
insdate							
join							
keydefs	keys						
l	lcd	lcd_rev	less				
make	match	method	middel	mono	more	myfirstc	myhelp
mysplit							
newline	nofkeys						
origin							
p2	package	pepadd	pref_paste	pretycom	profile		
quitall	quitfile						
reline	reload	repeatlo	reverse	ringlist	rm		
s-f5	savefile	say	seqbox	shiftblock	show	shownum	spell
stript	struct	summe	switchto				
tab	tglfkeys	toansi	toascii	tocp437	tohtml	total	totex
tryedit	typ						
umlaute	uncomm	uniq					
var							
windowex	words						
xpoint							

License:
undefined

Back to Table of Content

move to line

Author:
Scott Mattes
Download:
ml.the (1202 bytes)
Restriction:
unknown
Description:
Locates the line the target is in and then place the cursor at the beginning of the target.

If anything is on the command line at the time of execution, then that is used as the target; otherwise the user is prompted for the target.

The search starts at either the line the cursor is on or the focus line, depending.

License:
public domain

Back to Table of Content

pg_edit

Author:

Pablo García-Abia

Download:

pg_edit.the (3626 bytes)

Restriction:

unknown

Description:

Macro to be assigned to a key (mouse or keyboard). When issued, the string under the cursor (delimited by a given set of characters) is taken as a filename and the file, if exists, is edited. Otherwise, no action is taken.

This macro works in any file, not only in a directory list (like the command `sos edit` does).

For example, I have defined:

```
'define 2LB IN FILEAREA cursor MOUSE#macro pg_edit'
'define C-X macro pg_edit'
```

License:

unknown

Back to Table of Content

power input

Author:

Jonathan Beach

Download:

pow.zip (3761 bytes)

Restriction:

unknown

Description:

Its purpose is to provide a means of inputting text in THE in a manner similar to the power typing mode in the VM/CMS version of XEDIT (or at least my memory of how it operated 5 years or so ago.).

After putting the `pow.the` command in your macro path, you just issue the `POW` command from the THE command line and begin typing. As you type, the text is inserted directly into the document as the line is typed indented by the amount of the left margin setting. It stores the current line in a working variable and updates any changes to the line with the `REPLACE` command.

When you get to the end of the line, `POW` performs a word wrap based on the difference of your right margin setting and the left margin setting It does this by truncating the current line at the last space before the margin length, inserting that line into the file (or at the margin if there are no spaces on the line). The macro then inserts a new blank line in the file and puts the leftover text on that line.

If you press the ENTER key before reaching the end of the line, that line will be inserted into the document right away and a new blank line is started. To exit the macro, you just need to hit return on a blank line. To enter a blank line into the text, just precede it with a space. While the macro is running, the backspace, del, left cursor key and ^H (C-H) can all be used to go back one space to the beginning of the current line (mistakes on lines other than the current line must be corrected outside of the `POW` macro).

License:

Copyright to the Author

Back to Table of Content

power input another version

Author:

Homer Wilson Smith

Download:

from author's site (ftp://ftp.lightlink.com/pub/homer/the)
from contrib-site hspow.zip (3kB)

Restriction:

unknown

Description:

I have written three macros that emulate the CMS VM\370 behavior of XEDIT very closely.

They are `i.the` which replaces the `INPUT` command, and `enter.the` which is bound to the enter key via

```
define C-M macro enter
```

in your profile.

Hitting the enter key causes the screen to go into full input mode and it stays there until two empty returns are executed.

Those of you who can't stand (or can't understand :)) the default behavior of THE and KEDIT will love these macros. I have them for KEDIT too.

`i` followed by text will cause the text to be inserted.

`r` followed by text will cause the text to be replaced. `r` alone will replace the existing text and go into input mode. I just noticed the `r` command is missing, I will get it in the directory shortly.

License:

unknown

Back to Table of Content

power input, wrapped and sligthly modified

Author:

Lucio Chiappetti

Download:

mypow.zip (3.3kB)

Restriction:

Wrap for power input another version. Used with THE 3.0 in XEDIT compatibility mode on an Alpha with Digital Unix 3.2 and Regina 0.08h.

Description:

Essentially I like to have a flag on the status line telling whether power input is on or off. The other thing is that I'm not so interested in "real" POWERINPUT with line wrapping and margins (I write more code than text), so I use the [...] "fullmargin.the" macro to set margins at the physical size of my X11 window (which I can resize).

Package contains: mypow.the, fullmargin.the, i.the, r.the, enter.the, .therc (relevant snippet)

License:

unknown

Back to Table of Content

prefix put/del/get

Author:

Wesley J. Metzger

Download:

ppt.zip (2kB)

Restriction:

unknown

Description:

Copy selected lines in a ring of files from one file to a selected position (with g in the prefix region) of an other file.

`prefixpt:` REXX prefix macro to PUT or to PUT DELETE a line or block of lines.
`prefixg:` REXX prefix macro to do a GET after a line.

These commands are intended to be used from prefix area. You should define several *synonyms* to make use of them.

License:

unkown

Back to Table of Content

recordit

Author:

Mark J. Short

Download:

recordit.the (2kB bytes)

Restriction:

THE 2.7

Description:

This macro is used to record the cmdline into a file. What this gives you is the ability to write a macro as you test it. It is invoked by typing `RECORDIT filename` on the command line. This will cause every line command to be recorded in a filed named `filename.rthe` until `end` is typed on the command line. The file will have all the commands surrounded by quotes, and the comment line on line one.
NOTE: I named the file extension rthe instead of the to avoid accidents. From here you can add in loops, conditionals, etc...
Obviously, the macro should work on a single file only.
And: be careful with commands, which move the cursor from command line elsewhere.

License:

posted to news:comp.lang.rexx but copyrighted to the author.

Back to Table of Content

re-edit

Author:

Pablo García-Abia

Download:

redit.the (372 bytes)

Restriction:

THE 2.5

Description:

When editing a file turned into chaos you can throw away all changes and restart anew from the storage medium. The *corrupted* file get's saved anyway.

License:

unknown

Back to Table of Content

run

Author:
Sérgio Augusto Simão
Download:
run.zip (1988 bytes, run.the and profile.the)
Restriction:
at least THE 3.0
Description:
Runs external programm after saving files
License:
unknown

Back to Table of Content

search in file

Author:
Th. DeBoo
Download:
sif.the (1648 bytes)
Note: this is not the complete macro.
Restriction:
unknown (THE 2.3?)
Description:
Years ago, in 1984 to be more precise, I wrote a search-macro, a bit like ALL. This macro worked fine in XEDIT and later (adjusted) in KEDIT and still works fine.
License:
unknown

Back to Table of Content

swap_chars

Author:
Sean King
Download:
from author's site: swap_chars.the
from contrib-site: swap_chars.the (837 B)
Restriction:
none
Description:
transpose two characters
Transpose the cursor character with the character just preceding, and move right:
`teh quick --> the quick`
At the end of a line, transpose the positions of the last two characters on the line
`teh --> the`
License:
undefined

Back to Table of Content

Other Macro Related Informations

Tuning your Macros

Question: How are macros handled? Are they read from disk and interpreted each time they're called, or is there some way to keep them in memory? I have a macro that is assigned to the enter key, and so gets called quite frequently. I'd like to make it as efficient as possible.

Answer from ***Arthur Pool***:
In the OS/2 operating system, there are several different ways a REXX program can call a function coded in REXX, each of which has implications for both performance and maintainability. These options are discussed (briefly) in the WARP REXX on-line information (at least, in WARP Connect V3), which says (inter alia):

1. ... internal labels take first precedence,
2. then built-in functions,
3. and finally external functions.

External functions and subroutines have a system-defined search order. REXX searches for external functions in the following order:

1. Functions that have been loaded into the macrospace for pre-order execution;
2. Functions that are part of a function package;
3. REXX functions in the current directory, with the current extension;
4. REXX functions along environment PATH, with the current extension;
5. REXX functions in the current directory, with the default extension;
6. REXX functions along environment PATH, with the default extension;
7. Functions that have been loaded into the macrospace for post-order execution.

In practice then, the options, and their implications, are:

1. Include the source code for the function in the primary source file. This should provide the best performance. However, if the function is to be called by more than one `primary' REXX program, this approach is undesirable in that it requires duplication of code, with consequent maintenance problems.
2. Load the macro function into a REXX 'MacroSpace'. This can be achieved using the 'RexxAddMacro' call from 'C', or using the 'RxAddMacro' function provided in the 'RXU' utility package (I understand that WARP 4 (Merlin) provides a similar function in the REXXUTIL package but I have no experience with that). This approach is more or less equivalent to the `EXECLOAD' facility in VM/CMS. This approach should provide performance somewhere between the preceding and following approaches. It does however have some management costs:
    1. you have to explicitly load the function into the macrospace - if you don't, you'll simply execute the copy from disk, which will be much slower;
    2. you have to also ensure that the macrospace copy is unloaded and reloaded whenever the function's source file is modified - if you don't you'll be executing an out-of-date copy.

   As noted above, functions can be loaded in either of two ways:
    1. Loaded into the MacroSpace for pre-order execution (executed BEFORE disk-based files) - this should produce better performance than using disk-based functions.
    2. Loaded into the MacroSpace for post-order execution (executed AFTER disk-based files) - this should produce worse performance than using disk-based functions (below).

   However, it is worthwhile to consider the meaning of `current extension' and `default extension'. When a function is loaded into the MacroSpace, it can be loaded with an extension (eg, `.CMD') or without an extension. In these tests, as the primary REXX file had an extension of `.CMD', it appears that the `current extension' is `.CMD'. We therefore measured the performance of functions loaded into the MacroSpace (and called) with an extension of `.CMD' and also without an extension.

   Note however that in general one would prefer to load functions without extension so that they are equally accessible to REXX programs with any extension - whether called from REXX command files (.CMD), THE macros (.THE), or from other environments. It's also cumbersome to have to specify the extension when invoking the macro.

We measured 4 sub-cases:
[2a] MacroSpace function, pre-order, .CMD extension;
[2b] MacroSpace function, pre-order, no extension;
[2c] MacroSpace function, post-order, .CMD extension;
[2d] MacroSpace function, post-order, no extension;

3. Leave the source code as a separate file, which is invoked anew for each call to the function. Although ideal in terms of maintenance, this approach will not produce good performance. In practice, disk caching will reduce the impact of all function calls after the first.
   Also, the performance will be affected by whether the source file is in the current directory, or how far the system has to search along the PATH string before it finds the source file.
   We therefore test these sub-categories:
   [3a] source file in the CURRENT directory;
   [3b] source file in a directory at the START of the PATH string.
   [3c] source file in a directory at the END of the PATH string.
   [3d] source file in a directory at the END of the PATH string, without **EAs**.
   See below for the rationale for test [3d].

Testresults:

Following are some measurements of elapsed time (in seconds) for 255 function calls using these various approaches.

```
[C:\Usr\AFP\SW\Testing]REXX_Function_Call_Performance 255
[1]                              function in the source program:   0.88
[2a]           MacroSpace function, pre-order, .CMD extension:   2.06
[2b]             MacroSpace function, pre-order, no extension:   2.13
[2c]          MacroSpace function, post-order, .CMD extension:  91.09
[2d]            MacroSpace function, post-order, no extension: 180.87
[3a]  function in an external source file - CURRENT directory:  10.28
[3b]      function in an external source file - START of PATH:  12.66
[3c]        function in an external source file - END of PATH:  55.25
[3d] function in an external source file - END of PATH, no EAs:  42.12
[C:\Usr\AFP\SW\Testing]
```

Notes:

1. The function used was:
   ```
   REXX_Function_Call_Performance_1:
   return arg(1)**arg(1)
   ```
2. These measurements were on a 80486-DX4 with OS/2 Warp Connect (Blue Box) with no service pack applied, using (obviously) HPFS.

Conclusions:

1. As expected, including the function in the primary source file was fastest.
2. Loading the function in the MacroSpace for pre-order execution is slower than including the function in the primary program, but much faster than invoking from a disk file.
3. Loading the function in the MacroSpace with an extension of `.CMD' is faster than loading without any extension, most notably when loaded for post-order execution, presumably because the first search is for macros with a `.CMD' extension.
   Note however that to achieve this performance, the extension must be specified both when the file is loaded and when the function is invoked - somewhat cumbersome, and probably of

marginal benefit in the case of pre-order execution except in the most extreme cases.

4. When loading automatically from a disk file, the position of the function's source file in the PATH string can have a significant effect on performance - and the presence of network drives in the PATH string is likely to exacerbate the effect.
5. Loading in the MacroSpace for post-order execution is very slow. This slow performance can be partly explained because the system searches every directory in the search path (twice if the function is loaded without any extension), fails to find the function as an separate external file, and then finally looks in the MacroSpace (where it finds the function).
6. Even allowing for the preceding item, note that the performance of functions loaded in the MacroSpace for post-order execution is much slower in all cases than those loaded from disk (even when the source file's directory is at the END of the PATH string. Why is this?
   I thought that perhaps it was because the version loaded into the MacroSpace does not include the semi-compiled version which REXX normally stores in the Extended Attributes (EAs), but test [3d] shows that this alone does not explain the poor performance of tests [2c] and [2d].
7. Test [3d] (see above) strips the EAs from the function's source file and makes it read-only (which prevents REXX from attaching the semi-compiled form to the source file) to compare the performance of an external source file without the benefit of the semi-compiled form. Surprisingly, this version is faster than test [3c]!
   Unlikely though it seems, the semi-compiled form appears to be of no benefit in this test! Perhaps because the external function is quite small and relatively simple, the additional overhead of accessing and loading the EAs outweighs the benefit of the semi-compiled form?
   In any case, the poor performance of tests [2c] and [2d] remains a puzzle.
8. **Loading in the MacroSpace for pre-order execution therefore appears to be generally a good compromise - fairly good performance, easy to maintain, but has some management overheads.**

Files used:

```
C:\Usr\AFP\SW\Testing\REXX_Function_Call_Performance.cmd
C:\Usr\AFP\SW\Testing\REXX_Function_Call_Performance_2.cmd
```

Download:

Test scripts: afp_rx01.zip (7.503 bytes)
Necessary libraries: rexxutil.dll (comes with OS/2) and rxu.dll, rexxutil.zip (contains further documentation and samples):
from Hobbes at NMSU: rxu19.zip 01 December 1995, rexxutil.zip 31 January 1995 [file:11 July 1993], rxu1a.zip 8 May 1996

History:

1. Sent to THElist 1998-03-16
2. Updated copy sent to THElist 1998-08-03

Additional notes:

by Franz-Josef Wirtz

For REXX scripts (and THE macros) on the OS/2-platform so called *extended attributes (EAs)* store the token image from a REXX-macro after its first execution. The EAs are saved accompanying the original file and managed by the operating system. Thus, subsequent calls can use the tokenized image and eventually load and execute a little bit faster then (see result 3d and conclusion 7). I don't know, wether other REXX implementations use a similar mechanism.

A slight modified test script gave on my machine (P133 with Warp4 and ObjectRexx) the following results (the second figure is the factor compared to the first result):

```
[1]                              function in the source program:   0.12 > 1
[2a]            MacroSpace function, pre-order, .CMD extension:   0.43 > 4
[2b]              MacroSpace function, pre-order, no extension:   0.44 > 4
[2c]           MacroSpace function, post-order, .CMD extension:  4.28 > 36
[2d]             MacroSpace function, post-order, no extension: 10.66 > 89
[3a]   function in an external source file - CURRENT directory:  1.78 > 15
[3b]       function in an external source file - START of PATH:  2.16 > 18
[3c]         function in an external source file - END of PATH:  5.09 > 42
[3d] function in an external source file - END of PATH, no EAs:  6.06 > 51
```

Back to Table of Content

Converting KEDIT-macro-library (.kml)

Author:

Agustin Martin Domingo

Download:

kml2the.zip (2897 bytes)

Restriction:

Requires a c-compiler.

Description:

Since THE does not support macro librarys you have to put each macro into a separate .the-file. Here is a small utility for an automatic conversion. Thus, when you use **KEDIT** besides THE you only have to maintain the KEDIT-macro library. Nonetheless, you can't use REXX-variables across several macros this way, since the separated macro files don't have any relation to each other. They are loaded and executed each time they are called and unloaded immediatly after completion. If you need to communicate between macros, you may use environment variables for this job (with the *value* REXX function) or make use of the new `editv` command.

Another restriction may be, that within KEDIT macros you can't use select-when-otherwise-end constructs as in REXX (still true?), since KEDIT only implements a subset of REXX.

License:

GNU General Public License

Back to Table of Content

Porting XEDIT-macros

By *Mark Hessling*

I have been hearing from a few of you regarding the porting of XEDIT macros to THE, and thought I would suggest a couple of things that may make your life easier in the future.

While most XEDIT users will use THE in full XEDIT compatibility mode, I would strongly suggest that macros be written in THE compatibility mode.

The reason for this is that the behaviour of THE wrt movement of focus and current lines, movement of cursor etc. in various commands in THE compatibility mode is stable and unlikely to change, whereas in XEDIT compatibility mode, it is likely that behaviour will change as I fix it. The end result is that your macros will break :-(

You should add an `EXTRACT` of `COMPAT` at the beginning of the macro, followed by a `SET COMPAT THE THE THE`, and when the macro finishes, restore the settings saved.

Back to Table of Content

Writing OS-portable Macros

Some aspects need to be considerd when macros should be usable on various operating systems.

- REXX-interpreters differ [`stream()`, `directory()`, `index()`],
- command shells differ [handling of environment variables at OS-level or with `value()`, adressing the OS (`address cmd` vs. `address system`), names of external or internal utilities like `dir`-command (OS/2 <> WIN32) vs. `ls` -utility],
- file i/o differs,
- file name constitution differs [drive, dir separator, long filenames, file type, allowed chars],
- available utilities vary [e.g. `grep`, `ls`, `indent`, `fmt`, `ispell`, `rxqueue`, `etags`],
- available keys differ [only few keys with OS=UNIX, e.g. no ALT-keys],
- access to mouse differs [no mouse on OS=OS2 or OS=UNIX].
- some THE-stuff vary [Soft Label Keys, Color...]

Although this seems to make porting awkward, in real life it's not *that* problem. You should try it when the time comes to it and look at the forebears. You're not the first :-), so now it's a bit easier.

Back to **Table of Content**

Finding Errors in Your Macros

Question:
Is there a facility for tracing the operation (or, more aptly, the non-operation) of a THE macro which I am developing?

Answer:
This is a difficult task indeed. ;-)

Solution 1: Use REXX TRACE

You can use the REXX-instruction **`TRACE`** to give various output, but this will either go to a so called *pseudo file* within the ring of THE-files, called `REXX.$$$` or be displayed in an own window. The disadvantage on this is, that you have to wait until your macro is completed, and then analyse the output. When you call your macro via
`MACRO ? somefailingmacro somefancyarguments`
you can debug the macro interactively, if trace is set appropriately. For lengthy output it is recommended to use `SET REXXOUTPUT FILE 1000` so that you can page through the output (the number denotes the maximum size, in case you have infinit loops ;-).

On Linux with XCurses-THE this REXX-TRACE output will go to `stderr` which is much more useful. You can redirect it to a file (`the testfile.tst 2>the.log`) and follow this file synchonously with `tail -f the.log` in another terminal window.

I don't know wether there exists a parser which can display this output in an useful way. Any hints welcome. Maybe, the `RxD`-tool could be misused for that in some way, if the sources would be available (RxD is an interactive OS/2-Presentation Manager source level debugger for REXX programs by Patrick Mueller, available as IBM-EWS for free [**content**, **rxd.zip download 150k at leo** , many hobbes mirrors have it too])?

Solution 2a: Write to Message Line

A quick and dirty method is to to use the THE-command `MSG` to write something to the message line(s). When you're done you have to remove or comment out these lines.

Solution 2b: Write to Logfile

A slightly better way is to use another macro for doing some logging into a separate file. I use the following:

```
/* fwlog -- */

logfile = "fw.log"
parse arg filename msg

 call lineout logfile, date() time() filename":" msg
 call stream logfile, "command", "close"
exit
```

This macro closes the logging stream immediatly, thus you will see the output immediatly, e.g. when you use `tail -f fw.log` .

You would call this like

```
/* some damned macro.the */
wantlog = 1
myname = "some_damned_macro.the" /* you could use 'parse source' either...*/

somefancybuterroneousstuff
mymsg = "somevar="somevar
if (wantlog) then 'macro fwlog' myname mymsg
somemorefancystuff
exit
```

So, you fill up your macro with debugging code. At least, you can turn the output on or off.

Solution 3: Use THE-trace

You can compile THE with some options to produce its own trace output. I never did this and I think it's more for those who want to debug THE itself.

Solution 4: Use REXX SIGNAL

You can trap REXX exceptions. A very helpful one is the `signal on novalue`.

This might serve as a template for using this feature:

```
/* somefancymacro -- */

signal on failure name _FAILURE_EXIT
signal on halt name _HALT_EXIT
signal on syntax name _SYNTAX_EXIT
signal on novalue name _NOVALUE_EXIT
out.0 = 0
trace off
myname = 'somefancymacro'
myfilename = myname'.the'
logfile = "fw.log"

lotsoffancystuff

signal _CLEANUP
exit /*just in case*/

/*********************** signal targets **************************/
_SYNTAX_EXIT:
 syn_rc = rc
 trace off
 os = version.3()
 if (os == "OS2") then
 do
  rcinfo = "REX"||right( syn_RC, 4, '0')
  drop out.
  rc = run_os( 'helpmsg 'rcinfo, , 'out.')
 end

 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = 'Syntaxerror no. 'syn_rc' at line 'SIGL':'

 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = sourceline( SIGL)

 signal _CLEANUP
exit /* just in case */

_FAILURE_EXIT:
 trace off
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = 'General failure at line 'SIGL
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = sourceline( SIGL)
 signal _CLEANUP
exit /* just in case */

_HALT_EXIT:
 trace off
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = 'Halt condition (^C) met at line 'SIGL
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = sourceline( SIGL)
 signal _CLEANUP
exit /* just in case */

_NOVALUE_EXIT:
 trace off
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = 'Used variable "'condition("D")'"without a value at line 'SIGL
 i_line = out.0 + 1
 out.0 = i_line
 out.i_line = sourceline( SIGL)
 signal _CLEANUP
exit /* just in case */

_CLEANUP:
 trace off
 if (out.0 = 0) then
  exit

 'set msgmode on'
 'extract /msgline/'
 'set msgline on -1 'out.0' overlay'

 do i_out = 1 to out.0
  'msg 'out.i_out
 end
 'set msgline 'msgline.1 msgline.2 msgline.3 msgline.4
 if wantlog then
 do
  do i_out = 1 to out.0
   call lineout logfile,date() time() out.i_out
  end
  call stream logfile, "command", "close"
 end
exit
/* the end of somefancymacro.the */
```

When encountering the syntax exception this macros would call the OS/2-utility `helpmsg` for details on the error, and display this too in the msg area and the log file.

Note: instead of terminating your macro with `exit`, you would use `signal _CLEANUP`. Usually, that part is where you can handle other cleanup stuff which should be processed in any case (removing bak files or so). If you don't have something to process there, it would be no problem not using `signal _CLEANUP` but `exit` ;-)

By the way: I use an underscore as the first char of the label, just for convention, to distinguish this from a procedures name. Usually, with signal you won't be able to come back to the place where signal is triggered. You should not misuse the `signal` instruction in place of a `goto` (which does not exist in REXX).

Solution 5: Language Sensitive Editor

You can use a language sensitive editor which at least does not allow to insert unbalanced `do..end` or parentheses within call instructions and alike. Syntax highlighting is also a helpful feature which can enlight the structure and functionality of your macros. (I use ***emacs*** with rexx-mode and an own rexx-highlight-mode for this. :-O)

Solution 6: Macro Analyzing Tools

You can use tools for analyzing REXX-macros. Besides formatting and cross referencing the macro some can check for simple syntax errors. See coding suggestions.

Recommended Further Readings

There are some valuable hints on common problems with REXX:

- REXX Pitfalls
- Common REXX Coding Errors section in REXX-FAQ
- Trouble Shooting section within REXX Tips & Tricks

Typical THE Pitfall

With THE, a valuable source of error conditions is the mismatch of the place of action between what you're thinking and where THE is working. You have to be very careful on beeing sure, what the *current line* is and what the *focus line* is and where the *cursor* is. If cursor is within the *file area*, the focus is at the same place and most commands act there. When cursor is at command line, the focus is where cursor **was**, but action happens at current line. E.g. `locate` moves the current line to the target (if it can be found), but the focus line remains at the place where it was.

Some commands move the cursor from file area to command line implicitly. Note: with `sos makecurr` you can make the focus line the current line (e.g. the file will scroll), and with `sos current` you can move the cursor to the current line.

This becomes even more complicated depending on the compatibility mode you use. You're strongly recommended to use

```
set compat = THE =
```

where = is the mode for *look* and *keys* you like. (See Mark's note.)

Back to Table of Content

Sharing Your Macros

It would be great to see your macros in this list too:-) Thus others could make use of them. Also macros from different people would enlight the versatility and extensibility of THE. Another point is that each macro can show, how a distinct problem can be solved an serve as an example to encourage others to make their first steps.

Don't be afraid to receive annoying flames from people trying to use your macros. The users of THE are a quite friendly community and know of the problems of **writing** and sharing code while dipping into many **pitfalls** . As they use THE, they like to share with others as they know that only cooperation will improve the worlds face:-).

You can send any contributions as a posting to THELIST (in case you are already subscribed to that mailing list). Another way would be to email the macro(s) to **Franz-Josef Wirtz** as the current maintainer of this index file.

Some things you might consider on deploying your code:

- Accompany the macro with enough information to
  - identify yourself (e.g. email address),
  - show your licensing intention (e.g. public domain, GNU Public License, shareware or whatever else),
  - show your maintaining intention (do you want to get asked for making improvements or add improvements others wrote...?),
  - some info on the version,
  - tell the purpose,
  - the usage instruction
  - and any restrictions/ requirements of the macro.

  When it's one file, these infos should be placed in the header, when you pack a little zip, a separate README would be a better choice;
- When you pack several files together, you might worry about the naming of the package. You might think of the following naming convention:
  thefwfold27a.zip where
  - the: for use with THE
  - fw: initials of the author
  - 27: THE version already supported
  - a: your distribution counter
  - zip: standard ZIP-archiv.

You might also consult **KEDIT's guide for user contributions** for further tips.

Back to **Table of Content**

Coding Conventions

There are several styles around for coding REXX according to

- **Mike Cowlishaw** ,
- **Scott Maxwell** or
- Wide.

To follow one of these styles would improve the readability of your macros to others. Eventually you might look at **GNU coding standards** for further guidance. Although it's a bit c orientated, the general concepts are also valuable for REXX.

REXX formatter and cross referencer

There has been a *REXX formatter and cross referencer* available (**REXXREF3**) from ***Stephen Ferg***, but the last officially released version needs Personal REXX for DOS (by Quercus Systems, but see below for a solution without Personal REXX). The accompanying documents say, that intermediate versions of this were KEDIT macros. Maybe someone still owns these macros? This formatter / referencer can do also some error checking.

I wrote some REXX-scripts to simulate Personal REXX functions used within REXXREF3 and adopted it so that it can work on OS/2 and does no longer depend on the sort utility "lsrtos2" which is shareware. Instead it can use THE for this. You can get these scripts free upon request from me (Franz-Josef Wirtz).

Finally, *Stephen Ferg* is working on a more portable version which will work on OS/2 and linux as well. You may contact him to see its progress.

Another similar (non-free, 15$ shareware) tool is REXX Code Formatter/2 (RCF/2) (download 986k from hobbes), running on OS/2 Warp 3.0.

You can map REXX program function and subroutine references with CodeAnalyzer002.zip (OS/2) by *Doug Rickman*.

Casey Bralla wrote a freeware REXX Command Formatter - Indenter, which formats OS/2 REXX command files to indent Do / End structures.

Neil Hancock collected lots of general style guidelines for REXX-scripts.

Some other guidelines might be:

- Avoid abbreviated THE commands so the command name talks.
- Don't omit the `set` or `macro` keyword so it's clearer what is called.
- Avoid multiple statements in one line, so the logic becomes clearer.
- Introduce names for line targets with an underscore to differentiate to procedures.
- Put the macro name followed by at least two dashes into the first comment line, so that the rxtag -Utility can respect macro calls.
- Break and continue long lines with a `","` (comma) at the very end of each line, break long strings with the concatening operator `"||"` or `" "` (blank).
- Make freely use of blanks between variables, operators, arguments...
- More to be added...

There is a tutorial from Toby Thurston (Toby_Thurston@europe.notes.pw.com) on REXX-macros for the X2 editor by Blair W. Thompson (bwt@interlog.com) available at that webpage (ttxmacs.htm). This tutorial also covers some general guidelines for writing macros which are applicable to THE-macros as well. In brief, he pointed out the following:

- respect current word, line, marks or regions up to a passed target,
- respect current cursor position and restore it if appropriate,
- respect users settings (e.g. insert mode, word delimiters),
- restore settings if changed,
- respect national conventions from the OS (e.g. date time style, decimal point indicator and alike),
- allow with variables at the top an easy translation of fixed strings,
- generalize common stuff in procedure macros called by others,
- make use of general available functions (e.g. ispell, grep, sort) with an interface instead of implementing them anew in your macro,
- think of a way of undoing the effect of your macro,
- avoid the assignment of a macro which can do much harm to an easily accessible key to prevent a call by accident,
- think of giving help on a macro call which needs arguments when no one or a question mark is passed,
- try to be flexible in parsing arguments.

Back to Table of Content

Other Sources of Information on Writing Macros

If you are not familiar in writing macros the first you need is some knowledge over the functions and commands brought to you by THE. You can find information on this in the documents accompanying **THE** or it's master **XEDIT/KEDIT**. Just start at the functions suitable for doing the job you wanted to do in an automated way. You can try out the commands from the command line of THE. There's a macro available (**recordit**) to record your entered commands as a macro.

When you know, what you want to do with THE commands, you should then inform yourself about REXX, since THE macros are written in REXX. REXX embeds your THE commands into a script, which allows you to control the flow of commands. You can impose conditions, various kinds of loops and much more. REXX itself offers also a lot of useful functions, especially for dealing with strings. Thus, when THE does not have a string processing command, probably REXX has it. It is very likely, that the combination of REXX's and THE's commands and functions offer you all you need to process any text. THE manages the display and storage and moving around, and REXX manages the manipulation.

The REXX language passes any unknown stuff to the environment. A THE-macro is a REXX-script where it's environment is THE. THE checks all passed stuff against it's own extension (functions, commands, variables) and finally passes still unknown stuff to the environment of THE, which usually is the operating system (command interpreter, shell). This way, you have unlimited power for working on your text, since for example you can reformat text even with tools available on operating systems base like `sort`, `grep`, `ispell` or even `awk` and alike. That way, THE-macros are scripts, which glue together many powerful ways for processing text.

Sources of information and examples for giving your macro the desired structure with REXX can be found at the following places:

- XEDIT Reference
  - **z/VM V3R1.0 XEDIT Command and Macro Reference (pdf)**
- XEDIT Tutorials
  - **z/VM V3R1.0 XEDIT User's Guide (pdf)**
- REXX Reference
  - *Mike Cowlishaw*'s reference book "The Rexx Language" **(book data)** ,
  - REXX Online **Manual (html)** by *Department of Computer Science, University of Regina, California*
  - the extensive **Regina REXX Manual** (pdf)
  - the HTML-manual of free **IBM Object REXX** for Linux, including Classic REXX too)
  - **z/VM V4R2.0 REXX/VM Reference (pdf)** by IBM
- REXX Tutorials
  - **z/VM V3R1.0 REXX/VM User's Guide (pdf)** by IBM
  - ***Ian Collier's*** **REXX-Tutorial**,
  - SHARE Technology Conference-**Tutorial** for REXX with examples,
  - *Jeff Glatt*'s amusing "**Learn REXX programming in 56,479 easy steps**" (OS/2-inf-format 50k),
- REXX Tips
  - **Tips and Techniques** from Quercus hotline (Personal REXX),
  - ***Shmuel (Seymour J.) Metz***: **Safe REXX on the Desktop: or Will They Still Respect My Code in the Morning? (part 1 of 2)** resp. **(part 2 of 2)**
  - ***Bernd Schemmer***'s **Rexx Tips & Tricks** with lots of useful HowTo-code snippets,
  - *Dave Martin*: extensive **RexxFAQ** ,
  - *Dick Goran*: **"The Rexx Column"** in OS/2-magazine,
  - ***Charles Daney*** (by Quercus Systems): **Advanced REXX Programming Topics** (Download about 32k)

  - *Mike Cowlishaw*: REXX-Pages at IBM (many links, book list...),
  - *David Alcock*: Rexx Anywhere on portability issues
- Editor related stuff
  - *Nikolai Bezroukov*: commented collection of very useful links for XEDIT-like editors (called *eastern orthodox editors*). You can either download various types of macros from there or even follow to online references and guides of IBM's XEDIT or *Rex Swain's* comparative reference material for XEDIT and KEDIT and more.
  - ***Toby Thurston***: tutorial on REXX-macros for the **X2 editor** by *Blair W. Thompson*. Many example macros are available there too.
  - KEDIT Macro Library

The extensions of THE which are typical for text manipulation are *documented* in an own extensive HTML or plain text document. This online reference is enclosed in the distribution of THE (or can be generated from the source-distribution). The appendices 5 and 6 are especially useful for macro writers. Also within its distribution is a small set of macros, which enlight some interesting features. You should not miss these.

Note: Check out http://www.gut-wirtz.de/THE/rearranged/index.htm for an alternative structure of the HTML-manual.

Back to Table of Content

Coding THE-macros With Other Editors Than THE

Sometimes it's desired to have an editor at hand, which can correct a macro just changed and disallowing to edit files properly with THE;-). *Roger Nelson* gives an overview over nearly all available editors.

`emacs` there is a powerful REXX-mode available, written in 1994 by ***Scott Maxwell*** (email address probably defunct), archie or ftpsearch for `rexxmd` or look at sites hosting emacs contributions, it's not part of the standard emacs distribution; there's another, though less powerful rexx-mode from ***Anders Lindgren***, written in 1993 and part of XEmacs, waiting for a new maintainer; finally, a third approach on writing a rexx-mode (rx-mode.zip) had been started by *Espen Skoglung* and ***Ralf Grohmann*** around 1993/1994, thus intermediate between Scott's and Anders' mode).
There is also a highlight mode for *Scott Maxwell*'s mode to have the code in fancy colors (from ***Franz-Josef Wirtz***).
The FW-macros contain also a file (`rxtag.cmd`) to utilize `etags` for creating TAG files on THE-macros. With such a TAG file you can quickly look up the called macro or procedure.

Just for your interest, there is another free editor with REXX as it's scripting language available for several platforms: X2 editor by *Blair W. Thompson* (bwt@interlog.com), a programmers text mode / X-windows editor with keyword highlighting, tag, completion etc, derived from *Tim Baldwin*'s XE sample editor, with influencences from XEDIT as well (e.g. `all` command and many others). At it's homepage there are also many macros available. Maybe these might inspire you too. When looking at these macros you will feel yourself quite familiar to what you already know from THE:-)

Realization of this page Franz-Josef Wirtz (nospam-contact@gut-wirtz.de).
Copyright notice:
Parts of this file are taken from the documents available in the packages or from the postings on THELIST and thus the copyright is with each author.
The informational part is copyrighted to F.-J. Wirtz, except where declared otherwise. You are allowed to distribute this information as long as it's for free, since you got it for free too.
Last edited on: October, 18th 2007
