NAME

text2troff – converts a flat text-file to TROFF text format by inserting requests and ‘ms’ macro commands.

SYNOPSIS

text2troff [*options*] [*input-file*]

DESCRIPTION

text2troff is a bash script that uses a combination of *awk* and *sed* filters to convert a flat text-file into TROFF text format, by inserting TROFF request and ‘ms’ macro commands. The result is sent to standard output, and can be processed further with available ‘troff’ tools with ‘ms’ macro package, preferrably GNU groff_ms(7), to produce typeset PostScript-, PDF-, HTML- or terminal (‘nroff’) output. As an alternative to processing a text-file, text2troff can also read (text) input from a pipe.

The functionality includes interpretation of chapter headers, paragraphs, emphasized text, bullet item lists (whether or not nested), tables, footnotes and ‘code blocks’ (definition see below). It offers various options to set general text font family, style, size and color, the number of text columns per page, representation of chapter headers, paragraphs and tables, as well as end-of-sentence behavior. An option to automatically generate and include a Table of Contents (TOC) is present as well.

text2troff offers an alternative to pandoc(1) for producing TROFF text output, by attempting to minimize as much as possible the degree of formatting required in the input text, yet enabling still highly acceptable text typesetting based on a very limited set of simple conventions. These are used by text2troff as criteria by which to ‘best-guess’ the intended layout of the text.

The output produced by this program shouldn’t be considered as more than an approximation of any full-blown end result. Further polishing can be done by any additional manual placement of TROFF-requests and/or ms macros if chosen for. Main intent for developing this program was the wish to be able to turn a quick-and-dirty piece of flat text into a decently sophisticated document ‘at the push of a button’, with enough options to vary its appearance.

INPUT TEXT REQUIREMENTS

The following formatting conventions apply for flat input text in order to be interpreted properly:

Chapter headers

shall be single text lines that are no longer than 90 characters, start with a capital letter or a number, do not end with a period, comma, semicolon or exclamation mark, and are both preceded and followed by an empty line (*space*s and *tab*s allowed).

Paragraphs

shall be preceded by an empty line, which may contain *tabs* or *space*s.

Emphasized text

shall be surrounded by emphasis markers, consisting of one or more *asterisk*s, at both sides, e.g: `**text**`. No*tab*s nor *space*s are allo wed between the markers and the text. Each opening emphasis marker must be paired by a closing marker before a *tab*, an end-of-line or a footnote marker occurs.

Bullet list items

may be nested, are not a part of a table or code-block, and shall start with a line with either of the following elements at the beginning (after optional *space*s or *tab*s):

- one of the characters `* + - #` or `->` followed by a *space* or *tab*, or
- one of the following (combinations of) characters, followed by `) : . .) .]` or `]` and by a *space* or *tab*:
  - Roman numerals composed of up to seven (7) lower case *letter*s `i v x l c` optionally preceded by `[` or `(` as well
  - a string consisting of up to eight (8) *digit*s, *space*s and/or *period*s
  - an integer of up to four (4) *digit*s, followed by one (1) lower or upper case *letter*

- one (1) single upper or lower case *letter*
  or
- a string of any non-*tab* characters, followed by a *tab* only and (also in deviation of all previous possibilities) only one input text line

Code-blocks
are any (series of) lines that are supposed to have their literal flat text layout preserved, including monospace characters, line indents, spacing and breaks, and without any additional formatting. To distinguish a codeblock from preceding and succeeding text, following tags shall be present, separated by at least a *space*, a *tab* or a *newline* character from any other text:

`[code]` or `[code=`*name*`]`
with *name* = any set of words (e.g. a programming language), marks the beginning of a code-block, and is placed either at the beginning of the first code-block line, or at the end of the preceding line, which may be the last previous non-code-block text line.

`[/code]`
marks the end of a code-block, and is placed either at the end of the last code-block line, or at the beginning of the following line, which may be first next non-code-block text line. `[/code]` may be placed at the end of the same line as where `[code]` is placed at the beginning.

A code-block can't be placed within a table.

Tables are any contiguous series of two (2) or more text lines, each containing at least two (2) non-adjacent *tab* characters, excluding any leading or trailing *tab*s per line. This means that a minimum of three (3) text columns is required to be interpreted as a table. Any contiguous series of one or more *tab*s within a text line is treated as one delimiter between adjacent table-'cells' c.q. columns. This enables the use of multiple *tab*s in the input text, for instance to allign text belonging to the same column for the sake of readibility. As a consequence, table cells supposed to be 'empty' shall be filled by at least a *space* or e.g. a *dash* (`-`) or any other chosen character. Another consequence is that a *tab* character itself can – by definition – never be part of the contents of a table cell. All text intended to be in a single table row shall be placed on the same text line.

Footnotes
are text blocks starting with one or more *asterisk*s between *parentheses* e.g.: `(**)` at the beginning of the line. A footnote ends with an empty line or if a next footnote starts. The corresponding marking point is placed after a word in the main text, and is a *space* or *tab* followed by one or more *asterisk*s and a closing *parenthesis*, e.g.: `**)` The number of *asterisk*s in marking point and corresponding footnote can be chosen freely and needn't be equal. A footnote must start at the beginning of a line, may be placed anywhere in the text but is not interpreted if placed within a code-block or table. It may however contain tables, bullet lists and code-blocks by itself. It doesn't need to be co-located near a marking point, so might be placed as far away as e.g. at the very beginning or end of the file, as the program will automatically associate footnotes to (the locations of) marking points, based on order of appearance. Marking points may appear in any text, including chapter headers, tables, bullet lists, and in any quantity per textline. They are not interpreted however if placed within a codeblock or footnote.

OPTIONS

text2troff takes the following options:

-h Print 'usage' text for help.

-B Revert any 'solitary' bullet into main text. Solitary is defined as: the bullet is placed at a distance of more than 17 TROFF output lines from any other bullet item. (This option is not fully tested yet.)

-b Optimize output for 'browser' only, meaning 'troff' processing to HTML. By this option, text lines in a code-block are never broken, the number of text columns can be one (1) only, and font size can be 10 pts only. This option overrides options **-c**, **-d**, **-m**, **-t** and **-z**. Less suitable for 'troff'

processing to PDF or ‘nroff’ processing within terminal.

-c Optimize output for ‘console’ only, meaning ‘nroff’ processing within terminal. By this option, text lines in code-blocks are broken by terminal window boundaries only, unless option **-b** or **-m** is given as well. Less suitable for ‘troff’-processing to PDF/HTML. This option overrides option **-z**

-d Format the output into double-column (2) layout instead of one (1) column, unless option **-b** is given as well. This option overrides option **-m**

-m Format the output into multi-column (≥ 3) layout instead of one (1) column, unless option **-b** or **-d** is given as well.

-i Have every new paragraph start with an indented line. Without this option no indents are made.

-D Set TROFF creation date as the (fixed) document date. Without this option, groff_ms generates the (actual) document date while processing the TROFF-text.

-N Prefix all chapter headers by an automatically incremented number (to be generated by groff_ms), the first of which being 1.

-n Have each new line in the source text start on a new line in the typeset layout as well.

-p Have the typeset layout continue on a new line after each end-of-sentence period (.) encountered in the source text. Without this option, all running text continues after a period and is filled out to the right margin.

-e Expand tables to full width of the running text margins, Without this option, tables are centered with respect to the running text margins and not wider than strictly required by content.

-s Force (‘squeeze’) tables containing too many columns and/or unbreakable text to fit within running text margins, by adding invisible break characters to the contained text.

-t Automatically generate and include a Table of Contents (TOC), unless option **-b** is given as well.

-u Convert chapter header text from lower case to upper case.

-a *TYPE*

Convert characters with accent marks into escape-sequences.
This option enables reproduction by groff_ms of such characters in case an UTF-8 locale is not available on the system. Allowed values for *TYPE* are:

g GNU groff_char(7) escape-sequences

a AT&T ms legacy escape-sequences

b Berkeley ms legacy escape-sequences

-f *FONT*

Set general character font family as a replacement for ‘Times’. Allowed values for *FONT* are:

H Helvetica

h Helvetica Narrow

a Avant Garde

b Bookman

c Courier

n New Century Schoolbook

p Palatino

-S *STYLE*

Set general character style as a replacement for ‘Regular’. Allowed values for *STYLE* are:

i Italic

b Bolt

bi Bolt Italic

-z *SIZE* Set general character size as a replacement for 10 pts, unless option **-b** or **-c** is given as well. Also accepts values with decimal point.

-k *COLOR*
Set general text color as a replacement for the default. Allowed values for *COLOR* are:

a Aquamarine

b Blue

gn Green

gr Gray

m Magenta

o Olive

r Red

TEXT TRANSFORMATION TO TROFF FORMAT

Following transformations are made by text2troff to the input text to generate the TROFF text format:

Lines added at the top of the input text

`.RP`	Activated ms macro for ‘Released Paper’ format with cover sheet
`...TR`	Deactivated authentic AT&T ‘Internal Memorandum’ ms macro
`...IM`	same for ‘Computing Science Technical Report’
`...TM 76-1273-10 39199 39199-11`	same for ‘Technical Memorandum’, with preset text string
`...MF`	same for ‘Memorandum for File’
`...MR`	same for ‘Memorandum for Record’
`...EG`	same for ‘Engineer’s Notes’
`.ND "`*TROFF creation date*`"`	Only with option **-D**: set TROFF creation date as document date
`.fam` *font_family*	Only with option **-f**: set *font_family*
`.fp` *font_position style*	Only with option **-S**: set *font_positions* & *styles* (max 3x)
`.nr PS` *point_size*	Set *point_size* unless with options **-b** and **-c**
`.defcolor` *color* `rgb` *#rgb_value*	Define *color* w. *rgb_value* per **-k** option, empty otherwise
`\m[`*color*`]`	Set *color* per **-kf** option, *color* empty otherwise
`.nr HORPHANS 5`	ms macro to set header orphans
`.nr PORPHANS 3`	ms macro to set paragraph orphans
`.nr chapt_nr 0 1`	Set new number register ‘chapt_nr’ at 0 with increment 1
`.nr GROWPS 2`	Set number of character size levels including headers
`.nr PSINCR 1.5p`	Set character size increment at 1.5p
`.AM`	Only with option **-a b**: ms macro for Berkeley accent escapes
`.TL`	ms macro ‘document title’
`\f3`*filename*`\fP`	Name of the input text file as the document title
`.AU "`*location*`"` *contact*	ms macro ‘author’ with preset *location* and *contact*
author’s name	Preset *author’s name*
`.AI`	ms macro ’author’s institution’
author’s institution	Preset *author’s institution*
`.AB`	ms macro ‘begin abstract’
abstract text	Preset *abstract text*
`.AE`	ms macro ‘end abstract’
`...CS 12 1 13 0 0 10`	Deactivated special AT&T cover sheet macro with preset string
`.1C`	‘columns’: `.2C` (optn **-d** unless **-b**), `.MC` *width* (optns **-mcz** unless **-b**)
`.nr table_nr 0`	Initialize new number register ‘table_nr’ at 0
`.nr ps_decr \n[.s]*2/10`	Set new number register ‘ps_decr’ at 20% of point size

The macros at the beginning of above header allow variation in the appearance of the document, including a set of special macros by which to evoke some authentic AT&T document formats. In order to activate any

of the latter, a separate program called ‘trofform(1)’ is available. Additionally needed for this purpose is the original AT&T ‘tmac.s’ macro-file, which is in the public domain and can be found on the web.

General text treatment

text2troff inserts a **.br** ‘break line’ request underneath each line of input text that contains one or more *tab*s, or that contains a single string without a *space* or *tab*. If option-**n** is gi ven, **.br** is inserted underneath all other lines as well. If option **-p** is given, lines are broken after each period that ends a sentence, and a **.br** request is inserted on a new line in between.

In all above cases an exception is made for lines within tables and code blocks, which are never provided with line break requests. The same applies for lines above a chapter header, or where text2troff also places a **.LP** or **.PP** or **.IP** macro on the previous or following line.

Removed from the text, except within code blocks, are:

- *tabs* at the beginning or end of a line
- *space*s at the beginning or end of a line, except within tables
- any lines only consisting of minimally three (3) equal - + = or # characters

Character conversions

- All ISO-8859 encoded characters are re-encoded to UTF-8, and return characters (\r) are removed.
- Character combinations -> <- >= <= - and +/- are replaced by native TROFF **\(***xy* escape sequences, except if used in a code block. The same applies for other mathematical symbols and for Greek characters, but only if the locale is not set at UTF-8.
- IJ and ij are replaced by **\[IJ]** and **\[ij]** to make both characters an unbreakable unity, except if used in a code block.
- Characters with accent marks are copied to standard output or, if option **-a** is given, replaced by one the following escape sequence types, to have them represented correctly by groff_ms if the locale is not set at UTF-8:

*[*accent*]*character*	AT&T ms escape sequences (option **-a a**)
*character**[*accent*]	Berkeley ms escape sequences (option **-a b**)
\[*accent character*]	GNU escape sequences (option **-a g**)

- Each period at the beginning of a line (in codeblocks) or word (elsewere) is prefixed by \&
- Each backslash \ is prefixed by another backslash.
- Following characters are postfixed by \: to have groff_ms break strings not otherwise breakable, except if used in a code block:

  @ # % + = and _ not after a *tab* or *space*

  / not after a *tab* or *space* or after +

  ! ? and - not after a *letter* or *tab* or after any of the characters]) ! ? " ’ or -
- *Quote*s ’ ’ are converted to ‘ ’ to have groff_ms produce neat typeset opening and closing quotes, except if used within a code block.
- *Quote*s " " are converted to ‘‘ ’’ for same purpose, also except if used within a code block.

Character fonts, styles, sizes and text colors

Default general character font is Times Roman 10 pts, colour is standard (not specified). Both general font, style, size and text color can optionally be replaced by different values. Following character styles and sizes are set for the text categories listed:

Chapter title headers	Bolt 11.5 pts (default), or bolt version of chosen style and point size plus 1.5
Running text	Regular 10 pts (default), or chosen style and point size
Text in tables	Regular 8 pts (default), or chosen style and 80% of point size
Text in footnotes	Regular 8 pts (default), or chosen style and point size minus 2
Text within code-blocks	Courier monospace font regular 10 pts (default), or chosen point size If placed within a footnote: point size minus 2
Emphasized text	Bolt italic 10 pts (default), or chosen point size
URLS (`http(s)://...`)	Italic 10 pts (default), or italic version of chosen style and point size

Chapter headers and paragraphs:

If text2troff interprets a text line that is both preceded and succeeded by an empty line as a chapter header, then it replaces the preceding empty line by three (3) separate dots followed by ms macro **`.SH 1`** ('sub-header with header level 1').

The header line itself is prefixed by following series of escape sequences:

`\f[3]\n+[chapt_nr]\`*space*`\`*space*`\`*space*

meaning bolt version of used font style, auto-incremented chapter number (if **-N** option is given), and three (3) unpaddable *space*s respectively.

and, to return to previous font style, it is postfixed by:

`\f[P]`

text2troff treats any group of remaining empty lines not appearing within code blocks nor following a footnote as a paragraph beginning, and replaces it by ms macro **`.LP`** ('non-indented paragraph') or – if option **-i** is given – by ms macro **`.PP`** ('indented paragraph').

For example, the original text:

```
This is the last line of the previous chapter.

This is a new chapter

First paragraph line starts here ...
```

is converted by text2troff into:

```
This is the last line of the previous chapter.
.
.
.
.SH 1
\f3\n+[chapt_nr]\ \ \ This is a new chapter\fP
.LP
First paragraph line starts here ...
```

Table of Contents (TOC)

If option **-t** is given, but unless option **-b** is given as well, a **`.TC`** ('Table of Contents') ms macro is inserted on a new line at the end of the text, and underneath each chapter header the following lines are inserted:

`.XS`	ms macro 'start of this Table of Contents line'
`\m[`*color*`]`*Chapter_header_text*	Header repeated, no footnote markers, color request prefixed
`.XE`	ms macro 'end of this Table of Contents line'

If **-N** option is given, a chapter number register call `\n[chapt_nr].` is added between the color request and the chapter header text. groff_ms will now place a Table of Contents at the end of the typeset document. Using pdfroff -spdf instead of groff_ms, the table will be placed at the beginning of the document.

Emphasized word combinations

The strings of one or more `*` placed on both ends of emphasized text, as in following example:

```
Line with an **emphasized text** fragment.
```

are converted by text2troff as follows, to have groff_ms produce bolt italic font style:

```
Line with an \f[BI]emphasized text\f[P] fragment.
```

Bullets representation, nesting en indenting

Each character-string interpreted by text2troff as a bullet is placed between *double quote*s and is prefixed on the same line by ms macro **`.IP`** ('item pointer') and a *space*. Additionally, a **`\m[`***color***`]`** request is inserted between the opening *quote* and the bullet, with *color* as chosen with **-k** option or empty otherwise.
The text part to the right of the quoted bullet string is moved to a new line.

Following bullet types are converted to escape sequences representing a symbol:

- Dash bullets `-` are changed to **`\(en`** to have groff_ms produce an elongated dash: –
- Asterisk bullets `*` are changed to **`\(bu`** to have groff_ms produce a filled circle: •
- Right-arrow bullets `->` are changed to **`\(->`** to have groff_ms produce a right arrow: →

Any indentation in the input text is removed.

Nested bullet lists are treated by text2troff such to make groff_ms horizontally shift a bullet if this is provided with a different bullet type than its predecessor in the same list. In determining whether or not a bullet type change applies, text2troff treats any leading `[ (` and trailing `: . ) ]` characters and string length differences as non-destinctive. Furthermore, any remaining non-alphanumeric characters within strings are treated as (equal to) letters. Bullet differences are for instance:

– Difference in used single bullet symbol: `- + * ->` or `#`

– Any above symbol versus (strings of) alphanumerical characters

– Strings with leading letter(s) versus strings with leading digit(s):

– Digit strings without intermittent dots versus those with dots on distinct positions

If any difference remains, text2troff inserts one or more ms macros **`.RS`** ('right shift') or **`.RE`** ('left shift') on new lines on top of the item's **`.IP`** line. The program keeps track of shift position per bullet type and calculates the number of right shift and left shift macros needed to result in appropriate indent position.

The bullet list is considered terminated if an empty line is followed by a line without a bullet. The empty line is then replaced by a **`.LP`** macro, or **`.PP`** if option **-i** is given. On top of the **`.LP`** or **`.PP`** macro, an appropriate number of **`.RE`** macros is inserted in order to reset any remaining right shift.

For example, the original text:

```
After this line of main text, a bullet items list starts:
1.  Text of 1st main bullet item
2)  Text of 2nd main bullet item
*   Text of 1st sub bullet item
*   Text of 2nd sub bullet item
-   Text of 1st sub sub bullet item
-   Text of 2nd sub sub bullet item

After an empty line, the main text continues ...
```

is converted by text2troff, if option **-kr** is given, into:

```
After this line of main text, a bullet items list starts:
.IP "\m[red]1."
Text of 1st main bullet item
```

```
.IP "\m[red]2)"
Text of 2nd main bullet item
.RS
.IP "\m[red]\(bu" 3
Text of 1st sub bullet item
.IP "\m[red]\(bu" 3
Text of 2nd sub bullet item
.RS
.IP "\m[red]\(en"
Text of 1st sub sub bullet item
.IP "\m[red]\(en"
Text of 2nd sub sub bullet item
.RE
.RE
.LP
After an empty line, the main text continues ...
```

Treatment of text within code-blocks

The `[code]` tag is replaced by the same number of *space*s, and the `[/code]` tag is removed.

On top of the code block, ms macros **`.DS I 3`** ('display start' with indentation 3) and **`.CW`** ('constant width font') are inserted. Underneath the code block, ms macro **`.DE`** ('display end') is inserted.

Tabs within the code block are replaced by a number of *space*s consistent with original positioning and aligning, with the assumption that the *tab* is set to equal four (4) character positions. Removed are any empty lines (including *tab*s and/or *space*s) at the beginning and end of code-blocks, as well as any amount of common indent, leaving only the differences in indent intact.

For example, the original input text:

```
After this main text, a code-clock starts:
[code]

         some code text
            some more indented code text
      code text less indented than first line

[/code]
Main text continues here ...
```

is converted by text2troff into:

```
After this main text, a code-clock starts:
.DS I 3
.CW
   some code text
      some more indented code text
code text less indented than first line
.DE
Main text continues here ...
```

If option **-b** is given, a **`\f6`** request for Courier fond is inserted instead of the **`.CW`** macro, for better representation in an HTML browser.

text2troff breaks up code block lines that exceed the applicable text column margins into shorter units as to fit these margins as good as possible. The resulting line lengths depend on whether or not any of the options **-d** , **-m** and/or **-c** have been given. If option**-b** is gi ven though, code lines are not broken.

groff_ms will print each code block in courier monospace font at an indentation level of 3 characters with

respect to the left margin of the previous text.

Tables representation

text2troff calls the separate program text2troff_table(1) to convert series of tabulated lines into tables, by inserting a number of macros and requests. The definition for a table is: two (2) or more subsequent lines each containing at least three (3) 'table cells' delimited by *tab* characters.

Per line and except for the table header, each individual table cell as delimited by adjacent *tab*(s) is:

- prefixed by **`T{`** and a *newline* character, and
- postfixed by a *newline* character and **`T}`**

Any uninterrupted series of adjacent *tab*s is interpreted as **one** delimiter between neighboring cells, and is reduced to one *tab*.

For example, the original text:

```
After this line of main text, next lines represent a table:
first cell  tab   second cell: long text  tab  third cell
fourth cell tab   fifth cell    tab       tab  sixth cell
Here the main text continues after the table ...
```

is converted by text2troff the following way:

```
After this line of main text, next lines represent a table:
.LP                         Macro 'paragraph starts'
.nr table_nr +1             Number register 'table_nr' is incremented by 1
.ps -\n[ps_decr]            Point size minus decrement value being 20% of point size
.TS H                       Request 'table with header starts'
Tabel \n[table_nr]          Header text and incremented number register 'table_nr'
table_type, allbox;         Requests 'table_type' and 'box around every table cell'
c s s                       Header cell centered (c), spans (s) each other column
l l l .                     Each cell of next rows left (l) alligned
T{
first cell                  First row of table cells
T} tab T{
second cell: long text
T} tab T{
third cell
T}
T{
fourth cell                 Second row of table cells
T} tab T{
fifth cell
T} tab T{
sixth cell
T}
.TE                         Request 'table ends'
.LP                         ms macro 'paragraph starts'
Here the main text continues after the table ...
```

The *table_type* can be:

`center`

In this default case, groff_ms will center the table between the text margins, at a width no more than strictly required by table contents;

`expand`
: if option **-e** is given, or if option **-s** is given and table exceeds text margins, groff_ms will expand the table to full width between the text margins.

if option **-s** is given, invisible break characters **\:** are inserted in the table cell text after every *n* characters – with *n* depending on on table width, chosen number of columns and character type and size – as to have groff_ms break the lines and 'squeeze' the table between text margins.

Treatment of footnotes and marking points

Footnote marking strings of one or more * followed by) are replaced by a ****** escape sequence.
Footnote prefix strings of one or more * between ( ) are removed.
Footnote marking strings appearing within a footnoot or code-block are however left unchanged and literally copied to standard output. The same applies for a footnote prefix string within a table or code-block.

At each marking point found, text2troff moves the next footnote block to a line as close as possible under the marking point, in the same order of appearance.

On top of the footnote block, a **.FS** ('footnote start') ms macro is inserted. Underneath the footnote block a **.FE** ('footnote end') ms macro is inserted.

For example, the original text:

```
Next marking point *) in the main text refers to a footnote,
(*) First footnote goes here

next marking points **) and ***) refer to footnotes as well,
as the main text continues here
(**)  Here is the second footnote
(***) Followed by the third one

and continues further here ...
```

is converted by text2troff into:

```
Next marking point\** in the main text refers to a footnote,
.FS
First footnote goes here
.FE
next marking points\** and\** refer to footnotes as well,
.FS
Here is the second footnote
.FE
.FS
Followed by the third one
.FE
as the main text continues here
and continues further here ...
```

groff_ms will pair each footnote to a marking point appearing in the same order, by an automatically incremented superscript indentifier number. It will attempt to place each footnote at the bottom of the page that contains its associated marking point, or otherwise of the closest next page where it fits, keeping order of appearance intact. In case of more footnotes than marking points, the surplus amount of final footnoots is omitted by groff_ms. In the contrary case, empty footnotes are created near the surplus final marking points.

COMPATIBILITY

This program has been tested to run on both GNU/Linux (Ubuntu 22.04, Tiny Core Linux 10, Alpine Linux 3.12 within iSH app on iOS smartphone), as well as on MacOS X. In order to convert series of tabulated lines into tables, text2troff calls the separate program text2troff_table(1), which as a consequence must be

installed as well. text2troff_table can also be called stand-alone for tables exclusively.

BUGS

The program doesn’t support the code-blocks markers `[code]` and `[/code]` to appear as *literal code text themselves* within code-blocks.
Option **-B** to remove any ‘solitary’ bullets is not fully tested yet.
Interpretation of apostrophs isn’t flawless and may result into an opening quote depending on placement.
Page numbers and first footnote number don’t adopt the general text color setting and thus remain default.
Speed could be further improved.

AUTHOR

Written by Rob Toscani (rob_toscani@yahoo.com)

REPORTING BUGS

Please report any bugs to the author by e-mail or via https://github.com/jazzfan2/text2troff/issues