NAME

`.Mx` — reference extension for the mdoc manual language

SYNOPSIS

`.Mx −enable`[*`output-formats`*]

`.Mx −disable`

`.Mx`
`.Mx` *`macro`*
`.Mx` *`macro key`*

`.Mx −ix`[*`category`*] *`key`*
`.Mx −sx`[*`category`*]

`.Mx −toc`[*`output-formats`*] [**`−tree`** *`output-formats`*]

TABLE OF CONTENTS

DESCRIPTION

`mdocmx`(7)[44] extends the `mdoc`(7)[45] UNIX manual page markup language with references, allowing it to create anchors and table of contents (and potentially an index). The resulting documents are valid, compatible and distributable, and will become fully interactive if the local formatter is `mdocmx`(7)[46] aware. A new multiplexer command achieves this: **`.Mx`**.

Because single-pass `troff`(1)[47] implementations cannot create forward references documents which use this extension need to be preprocessed with the `mdocmx`(1)[48] preprocessor, implemented in portable `sh`(1)[49] and `awk`(1)[50]. The mentioned properties also apply to preprocessed manuals. For more please see **COMPATIBILITY[14]**.

The `mdocmx`(7)[51] reference extension augments the standard `mdoc`(7)[52] document prologue – **`.Dd`**, **`.Dt`** and **`.Os`** – with '**`.Mx −enable`**'. The extension can be restricted to specific output formats by specifying the desired ones as further arguments; for convenience all typewriter-like devices can be addressed with *`tty`*. It is not an error to specify a device for which no special `mdocmx`(7)[53] support is available, such requests are silently ignored.

Creating referenceable anchors

When **-enable**d in the document prologue the third group of **.Mx** usage forms enqueue index anchor requests. These form a last–in — first–out stack that is consumed (popped) by later occurrences of (corresponding) mdoc(7)[54] *macro*s which support referenceable index entries. Each supported *macro* has its own namespace, meaning that, for example, **.Mx** Ic sendmail and **.Mx** Va sendmail will create distinct index anchors.

Using the plain macro **.Mx** without arguments creates a stack entry for which both, the name of the *macro* as well as the *key* are taken from the document content (as soon as possible). '**.Mx** *macro*' creates a stack entry that is consumed by the next occurrence of *macro*, then taking the *key* off the document content, whereas '**.Mx** *macro key*' creates an exact stack entry with also *key* defined, it is popped by the exact macro / key match only.

Referenceable anchors for the mdoc(7)[55] section header commands **.Sh** and **.Ss** are created automatically, but, again, with mdocmx(7)[56] any **.Sx** referencing them will become interactive. External cross-references defined via **.Xr** will also be backed with functionality; dependent upon the output format this requires setting mx-xr-url[42], however. The following macros gain support for referenceable anchors via **.Mx**:

.Ar	Command argument.
.Cd	Configuration declaration.
.Cm	Command modifier.
.Dv	Defined variable or preprocessor constant.
.Er	Error constant.
.Ev	Environment variable.
.Fl	Command line option (flag).
.Fn	Function name.
.Fo	Function name (in function block syntax). This is mapped to **.Fn[26]**, **.Fo** has no index by itself.
.Ic	Internal or interactive command.
.In	An include file for, e.g., the C programming language.
.Pa	File system path.
.Va	Variable name.
.Vt	Variable type, type reference.

String cleanup and reference prevention

Before strings get used for anchor creation or reference lookup any surrounding whitespace will be removed, as well as any preceding '\&', '\%' and postposed '\&', '\%', '\/', and '\c' escape characters. Yet, prefixing a command with two zero-width glyphs (after possible whitespace), as in \&\&, prevents reference lookup for the remaining arguments of the command. For example, in the hypothetic .Ic if , elif , else , endifall four commands w ould be linked, but in .Ic \&\&if , elif , else , endif none of them; if that is not desired, a new command needs to be started: .Ic \&\&if , Ic elif , else , endif.

Freely definable anchors and references

Via the '**.Mx -ix** *category key*' and '**.Mx -ix** *key*' usage forms anchors can be defined almost everywhere. For example, '**.Mx -ix**subsubsection "An interesting topic"' defines the anchor An interesting topicfor the "k ey" subsubsection. The form without a specified*category* will use the builtin name ixsx instead.

References to such anchors can be made by activating the **.Sx** search extension via '**.Mx -sx** *category*' (or '**.Mx -sx**' for the builtin **ixsx[33]** *category*) followed by a normal local reference lookup. This complication is owed to compatibility of the resulting document.

```
.Mx -sx subsubsection
.Sx "An interesting topic"
```

Creating table of contents

The final **.Mx** usage form allows creation of a table of contents, which is of special interest when converting into formats such as HTML, XHTML or PDF. To restrict the creation of the table of contents to special output formats, add their names as further arguments to **−toc**; for convenience all typewriter-like devices can be addressed with *tty*.

By default only **.Sh** section headers are a vivid part of the TOC; in order to include **.Ss** subsections add **−tree**. Note that if **−tree** is used in conjunction with output-format restrictions it will only affect those output-formats that appear later on the line.

In the first of the following examples a table of contents will be generated for PDF and typewriter-like devices. In the second example a tree of contents will instead be generated for the output formats PDF and HTML, whereas typewriter-like devices will see a flat table of contents with only section headers.

```
.Mx -toc pdf tty
.Mx -toc tty -tree html pdf
```

Strings that affect mdocmx

Due to deficiencies in some implementations of `troff`(1)[57] strings given on the command line (via **−d**) have to be given an argument to be perceivable on the macro level. Alternatively the shown numbers can be bitmixed via `MDOCMX_FLAGS`[43].

`mx-debug`	(2) If defined `mdocmx`(7)[58] macros will offer some verbosity. In addition not only references will produce visual output, but also anchors.
`mx-anchor-dump`	If this is set to a filename then the list of anchors is dumped to it.
`mx-disable`	(4) Has the same effect as '**.Mx −disable**'.
`mx-toc-disable`	(8) Forcefully turn off any table of contents creation.
`mx-toc-emerged`	(16) Normally compact display is used for the table of contents, but when this string is set an emerged display is used for the first level that lists the headings.
`mx-toc-force`	(32) Defining this can be used to enforce the creation of a table of contents as specified, even if the documents **−toc** configuration would not create one for the targeted output device. A flat table of contents will be generated unless the value is (64) `tree`.
`mx-toc-name`	If defined its content is used as the headline of the table of contents. The default is "TABLE OF CONTENTS". (Note that if the table of contents has instead been generated by the `mdocmx`(1)[59] preprocessor then the resulting document already includes a definition of this string to ensure compatibility with, at least, `mandoc`(1)[60].)
`mx-toc-numbered`	(128) If defined the first level of the table of contents will be numbered.
`mx-xr-url`	The output formats HTML, XHTML and PDF require additional information, a target address, in order to be able to create interactive external **.Xr** references. Two modes are supported: one may assign an URL string or the hyphen-minus '`-`', in which case a macro `mx-xr-url-create` will be called. Thanks to `troff`s late evaluation, the three arguments sequence number, manual section number, and reference name are available in both cases. The macro mode is therefore only needed when filtering is necessary, for example:

```
$ mdocmx mdocmx.7 |
   MDOCMX_FLAGS=64 roff -Tpdf -mdoc \
    -dmx-xr-url='https://XY.org/man/\$3.\$2' \
    > x.pdf

$ cat filter-example.tr
.eo
.de mx-xr-url-create
. if 'html'\*[.T]' \{\
.  ie 'that-manual'\$3' \
.   URL "https://man.YZ.org/\$3-\$2.htm" "[\$1]"
.  el \
.   URL "https://XY.org/man/\$3.\$2" "[\$1]"
. \}
..
```

IMPLEMENTATION NOTES

The **.Mx** request cannot share a line with other macros, neither in the document prologue nor in its content. Whereas that is mostly owed to the necessity of ensuring (backward) compatibility with environments that do not support mdocmx(7)[61], it also simplified implementation of the preprocessor.

Due to the way GNU mdoc(7)[62] is implemented, visual references will be placed after their text, instead of creating the well-known link style (for at least those output formats for which such style makes sense). Due to the same reason section headers which contain mdoc or troff commands alongside their content string are not supported. All this could be overcome by changing the recursive descendent GNU mdoc implementation that changes content during its descend, a howto is thought (commented in mdocmx source).

Internal extended synopsis

In addition to those usage forms that have been described above the **.Mx** multiplexer command also understands further flags and arguments which are of possible interest for formatter and macro implementors. These further flags and arguments are only generated by the mdocmx(1)[63] preprocessor and are solely ment to communicate the preprocessed state of the document to the actual consumers.

For one a **−preprocessed** flag is appended to the single **−enable** command in the document prologue. And then an additional **−anchor-spass** form is introduced, which tak es two or three arguments – the macro (name of the command) for which this defines an anchor as well as its key, possibly followed by a numeric argument that describes the relationship in between section headings: for **.Sh** commands it defines a running one-based index count of section headers, for **.Ss** commands it instead specifies the index of the section header they belong to, therefore creating the possibility to generate TOCs.

ENVIRONMENT

The environment variable MDOCMX_FLAGS may be set to a bitmix of the **Strings that affect mdocmx[9]**. So for example `export MDOCMX_FLAGS=4` disables mdocmx(7)[64], whereas `export MDOCMX_FLAGS=$((64 | 16))` sets mx-toc-force[39] to produce a tree view of the table of contents, and also mx-toc-emerged[38] to make it appear less compact.

EXAMPLES

A complete, but completely fanciful mdoc(7)[65] document that uses the mdocmx(7)[66] extension would for example be:

```
.Dd April 22, 2015
.Dt MDOCMX-EXAMPLE 7
.Os
```

```
.Mx -enable tty
.
.Sh NAME
.Nm mdocmx-example
.Nd An example for the mdocmx mdoc reference extension
.
.Mx -toc
.
.Sh DESCRIPTION
Sors salutis et virtutis michi nunc contraria.
.
.Bl -tag -width ".It Fn _a_e_i_"
.Mx
.It Ic .Ar
This will create an anchor for a macro
.Ql \&Ic ,
key
.Ql .Ar .
.Mx
.It Ic .Cm
Anchor for
.Ql \&Ic ,
key
.Ql .Cm .
.Mx
.It Ic .Dv
And an anchor for
.Ql \&Ic ,
key
.Ql .Dv .
.Mx Ic
.Mx Ic "final anchor"
.Mx Fn _atexit
.It Fn exit
No anchor here.
.It Fn at_quick_exit , Fn _atexit
Not for the first, but for the second
.Ql \&Fn
there will be an anchor with the key
.Ql _atexit .
.It Ic "no anchor here"
.It Ic "final anchor"
Pops the pushed
.Ql \&Ic
/
.Ql final anchor
macro / key pair.
.It Ic ciao
Pops the
.Ql \&Ic
and assigns the key
.Ql Ciao .
```

```
.El
```

COMPATIBILITY

Using the mdocmx(7)[67] extension in mdoc(7)[68] manual pages should not cause any compatibility problems in so far as all tested environments silently ignore the unknown commands by default. Because of this, and due to the nature of this extension, an interesting, backward as well as forward compatible approach to use mdocmx(7)[69] may be to preprocess manuals with mdocmx(1)[70] on developer machines and instead distribute the resulting documents.

SEE ALSO

awk(1)[71], mandoc(1)[72], mdocmx(1)[73], sh(1)[74], troff(1)[75], mdoc(7)[76]

HISTORY

The **.Mx** environment appeared in 2014. The terminal output was rewritten to use OSC 8 control sequences in 2022.

AUTHORS

Steffen Nurpmeso <steffen@sdaoden.eu>.

CAVEATS

Be aware that the content of the **-width** argument to mdoc(7)[77] lists etc. is evaluated as if it were normal document content; for example, in the following example the Fn _atexit will be evaluated and may thus get used by **.Mx**:

```
.Bl -tag -width ".It Fn _atexit"
```