NAME

mdoc - semantic markup language for formatting manual pages

DESCRIPTION

The **mdoc** language supports authoring of manual pages for the man(1) utility by allowing semantic annotations of words, phrases, page sections and complete manual pages. Such annotations are used by formatting tools to achieve a uniform presentation across all manuals written in **mdoc**, and to support hyperlinking if supported by the output medium.

This reference document describes the structure of manual pages and the syntax and usage of the **mdoc** language. The reference implementation of a parsing and formatting tool is mandoc(1); the *COMPATIBILITY* section describes compatibility with other implementations.

In an **mdoc** document, lines beginning with the control character ‘.’ are called "macro lines". The first word is the macro name. It consists of two or three letters. Most macro names begin with a capital letter. For a list of available macros, see *MACRO OVERVIEW*. The words following the macro name are arguments to the macro, optionally including the names of other, callable macros; see *MACRO SYNTAX* for details.

Lines not beginning with the control character are called "text lines". They provide free-form text to be printed; the formatting of the text depends on the respective processing context:

```
.Sh Macro lines change control state.
Text lines are interpreted within the current state.
```

Many aspects of the basic syntax of the **mdoc** language are based on the roff(7) language; see the *LANGUAGE SYNTAX* and *MACRO SYNTAX* sections in the roff(7) manual for details, in particular regarding comments, escape sequences, whitespace, and quoting. However, using roff(7) requests in **mdoc** documents is discouraged; mandoc(1) supports some of them merely for backward compatibility.

MANUAL STRUCTURE

A well-formed **mdoc** document consists of a document prologue followed by one or more sections.

The prologue, which consists of the **Dd**, **Dt**, and **Os** macros in that order, is required for every document.

The first section (sections are denoted by **Sh**) must be the NAME section, consisting of at least one **Nm** followed by **Nd**.

Following that, convention dictates specifying at least the *SYNOPSIS* and *DESCRIPTION* sections, although this varies between manual sections.

The following is a well-formed skeleton **mdoc** file for a utility "progname":

```
.Dd $Mdocdate$
.Dt PROGNAME section
.Os
.Sh NAME
.Nm progname
.Nd one line about what it does
.\" .Sh LIBRARY
.\" For sections 2, 3, and 9 only.
.\" Not used in OpenBSD.
.Sh SYNOPSIS
.Nm progname
.Op Fl options
.Ar
.Sh DESCRIPTION
The
.Nm
utility processes files ...
.\" .Sh CONTEXT
.\" For section 9 functions only.
.\" .Sh IMPLEMENTATION NOTES
.\" Not used in OpenBSD.
.\" .Sh RETURN VALUES
.\" For sections 2, 3, and 9 function return values only.
.\" .Sh ENVIRONMENT
.\" For sections 1, 6, 7, and 8 only.
.\" .Sh FILES
.\" .Sh EXIT STATUS
.\" For sections 1, 6, and 8 only.
.\" .Sh EXAMPLES
.\" .Sh DIAGNOSTICS
.\" For sections 1, 4, 6, 7, 8, and 9 printf/stderr messages only.
.\" .Sh ERRORS
.\" For sections 2, 3, 4, and 9 errno settings only.
.\" .Sh SEE ALSO
.\" .Xr foobar 1
.\" .Sh STANDARDS
.\" .Sh HISTORY
.\" .Sh AUTHORS
```

```
.\" .Sh CAVEATS
.\" .Sh BUGS
.\" .Sh SECURITY CONSIDERATIONS
.\" Not used in OpenBSD.
```

The sections in an **mdoc** document are conventionally ordered as they appear above. Sections should be composed as follows:

NAME
The name(s) and a one line description of the documented material. The syntax for this as follows:

```
.Nm name0 ,
.Nm name1 ,
.Nm name2
.Nd a one line description
```

Multiple ‘Nm’ names should be separated by commas.

The **Nm** macro(s) must precede the **Nd** macro.

See **Nm** and **Nd**.

LIBRARY
The name of the library containing the documented material, which is assumed to be a function in a section 2, 3, or 9 manual. The syntax for this is as follows:

```
.Lb libarm
```

See **Lb**.

SYNOPSIS
Documents the utility invocation syntax, function call syntax, or device configuration.

For the first, utilities (sections 1, 6, and 8), this is generally structured as follows:

```
.Nm bar
.Op Fl v
.Op Fl o Ar file
.Op Ar
```

```
.Nm foo
.Op Fl v
.Op Fl o Ar file
.Op Ar
```

Commands should be ordered alphabetically.

For the second, function calls (sections 2, 3, 9):

```
.In header.h
.Vt extern const char *global;
.Ft "char *"
.Fn foo "const char *src"
.Ft "char *"
.Fn bar "const char *src"
```

Ordering of **In**, **Vt**, **Fn**, and **Fo** macros should follow C header-file conventions.

And for the third, configurations (section 4):

```
.Cd "it* at isa? port 0x2e"
.Cd "it* at isa? port 0x4e"
```

Manuals not in these sections generally don’t need a *SYNOPSIS*.

Some macros are displayed differently in the *SYNOPSIS* section, particularly **Nm**, **Cd**, **Fd**, **Fn**, **Fo**, **In**, **Vt**, and **Ft**. All of these macros are output on their own line. If two such dissimilar macros are pairwise invoked (except for **Ft** before **Fo** or **Fn**), they are separated by a vertical space, unless in the case of **Fo**, **Fn**, and **Ft**, which are always separated by vertical space.

When text and macros following an **Nm** macro starting an input line span multiple output lines, all output lines but the first will be indented to align with the text immediately following the **Nm** macro, up to the next **Nm**, **Sh**, or **Ss** macro or the end of an enclosing block, whichever comes first.

DESCRIPTION
This begins with an expansion of the brief, one line description in *NAME*:

```
The
.Nm
```

utility does this, that, and the other.

It usually follows with a breakdown of the options (if documenting a command), such as:

The options are as follows:
.Bl -tag -width Ds
.It Fl v
Print verbose information.
.El

List the options in alphabetical order, uppercase before lowercase for each letter and with no regard to whether an option takes an argument. Put digits in ascending order before all letter options.

Manuals not documenting a command won't include the above fragment.

Since the *DESCRIPTION* section usually contains most of the text of a manual, longer manuals often use the **Ss** macro to form subsections. In very long manuals, the *DESCRIPTION* may be split into multiple sections, each started by an **Sh** macro followed by a non-standard section name, and each having several subsections, like in the present **mdoc** manual.

CONTEXT
This section lists the contexts in which functions can be called in section 9. The contexts are autoconf, process, or interrupt.

IMPLEMENTATION NOTES
Implementation-specific notes should be kept here. This is useful when implementing standard functions that may have side effects or notable algorithmic implications.

RETURN VALUES
This section documents the return values of functions in sections 2, 3, and 9.

See **Rv**.

ENVIRONMENT
Lists the environment variables used by the utility, and explains the syntax and semantics of their values. The environ(7) manual provides examples of typical content and formatting.

See **Ev**.

FILES
Documents files used. It’s helpful to document both the file name and a short description of how the file is used (created, modified, etc.).

See **Pa**.

EXIT STATUS
This section documents the command exit status for section 1, 6, and 8 utilities. Historically, this information was described in *DIAGNOSTICS*, a practise that is now discouraged.

See **Ex**.

EXAMPLES
Example usages. This often contains snippets of well-formed, well-tested invocations. Make sure that examples work properly!

DIAGNOSTICS
Documents error messages. In section 4 and 9 manuals, these are usually messages printed by the kernel to the console and to the kernel log. In section 1, 6, 7, and 8, these are usually messages printed by userland programs to the standard error output.

Historically, this section was used in place of *EXIT STATUS* for manuals in sections 1, 6, and 8; however, this practise is discouraged.

See **Bl -diag**.

ERRORS
Documents errno(2) settings in sections 2, 3, 4, and 9.

See **Er**.

SEE ALSO
References other manuals with related topics. This section should exist for most manuals. Cross-references should conventionally be ordered first by section, then alphabetically (ignoring case).

References to other documentation concerning the topic of the manual page, for example authoritative books or journal articles, may also be provided in this section.

See **Rs** and **Xr**.

STANDARDS
References any standards implemented or used. If not adhering to any standards, the *HISTORY* section should be used instead.

See **St**.

HISTORY
A brief history of the subject, including where it was first implemented, and when it was ported to or reimplemented for the operating system at hand.

AUTHORS
Credits to the person or persons who wrote the code and/or documentation. Authors should generally be noted by both name and email address.

See **An**.

CAVEATS
Common misuses and misunderstandings should be explained in this section.

BUGS
Known bugs, limitations, and work-arounds should be described in this section.

SECURITY CONSIDERATIONS
Documents any security precautions that operators should consider.

MACRO OVERVIEW

This overview is sorted such that macros of similar purpose are listed together, to help find the best macro for any given purpose. Deprecated macros are not included in the overview, but can be found below in the alphabetical *MACRO REFERENCE*.

Document preamble and NAME section macros

Dd	document date: **$Mdocdate$** | *month day*, *year*
Dt	document title: *TITLE section* [*arch*]
Os	operating system version: [*system* [*version*]]
Nm	document name (one argument)
Nd	document description (one line)

Sections and cross references

Sh	section header (one line)
Ss	subsection header (one line)

Sx	internal cross reference to a section or subsection
Xr	cross reference to another manual page: *name section*
Tg	tag the definition of a *term* (<= 1 arguments)
Pp	start a text paragraph (no arguments)

Displays and lists

Bd, **Ed**	display block: **-***type* [**-offset** *width*] [**-compact**]
D1	indented display (one line)
Dl	indented literal display (one line)
Ql	in-line literal display: ‘text’
Bl, **El**	list block: **-***type* [**-width** *val*] [**-offset** *val*] [**-compact**]
It	list item (syntax depends on **-***type*)
Ta	table cell separator in **Bl -column** lists
Rs, **%***, **Re**	bibliographic block (references)

Spacing control

Pf	prefix, no following horizontal space (one argument)
Ns	roman font, no preceding horizontal space (no arguments)
Ap	apostrophe without surrounding whitespace (no arguments)
Sm	switch horizontal spacing mode: [**on** | **off**]
Bk, **Ek**	keep block: **-words**

Semantic markup for command line utilities

Nm	start a SYNOPSIS block with the name of a utility
Fl	command line options (flags) (>=0 arguments)
Cm	command modifier (>0 arguments)
Ar	command arguments (>=0 arguments)
Op, **Oo**, **Oc**	optional syntax elements (enclosure)
Ic	internal or interactive command (>0 arguments)
Ev	environmental variable (>0 arguments)
Pa	file system path (>=0 arguments)

Semantic markup for function libraries

Lb	function library (one argument)
In	include file (one argument)
Fd	other preprocessor directive (>0 arguments)
Ft	function type (>0 arguments)
Fo, **Fc**	function block: *funcname*
Fn	function name: *funcname* [*argument* ...]
Fa	function argument (>0 arguments)

Vt	variable type (>0 arguments)
Va	variable name (>0 arguments)
Dv	defined variable or preprocessor constant (>0 arguments)
Er	error constant (>0 arguments)
Ev	environmental variable (>0 arguments)

Various semantic markup

An	author name (>0 arguments)
Lk	hyperlink: *uri* [*display_name*]
Mt	"mailto" hyperlink: *localpart@domain*
Cd	kernel configuration declaration (>0 arguments)
Ad	memory address (>0 arguments)
Ms	mathematical symbol (>0 arguments)

Physical markup

Em	italic font or underline (emphasis) (>0 arguments)
Sy	boldface font (symbolic) (>0 arguments)
No	return to roman font (normal) (>0 arguments)
Bf, **Ef**	font block: **-***type* | **Em** | **Li** | **Sy**

Physical enclosures

Dq, **Do**, **Dc**	enclose in typographic double quotes: "text"
Qq, **Qo**, **Qc**	enclose in typewriter double quotes: "text"
Sq, **So**, **Sc**	enclose in single quotes: ‘text’
Pq, **Po**, **Pc**	enclose in parentheses: (text)
Bq, **Bo**, **Bc**	enclose in square brackets: [text]
Brq, **Bro**, **Brc**	enclose in curly braces: {text}
Aq, **Ao**, **Ac**	enclose in angle brackets: <text>
Eo, **Ec**	generic enclosure

Text production

Ex -std	standard command exit values: [*utility* ...]
Rv -std	standard function return values: [*function* ...]
St	reference to a standards document (one argument)
At	AT&T UNIX
Bx	BSD
Bsx	BSD/OS
Nx	NetBSD
Fx	FreeBSD
Ox	OpenBSD

Dx DragonFly

MACRO REFERENCE

This section is a canonical reference of all macros, arranged alphabetically. For the scoping of individual macros, see *MACRO SYNTAX*.

%A *first_name ... last_name*
Author name of an **Rs** block. Multiple authors should each be accorded their own **%A** line. Author names should be ordered with full or abbreviated forename(s) first, then full surname.

%B *title*
Book title of an **Rs** block. This macro may also be used in a non-bibliographic context when referring to book titles.

%C *location*
Publication city or location of an **Rs** block.

%D [*month day*,] *year*
Publication date of an **Rs** block. Provide the full English name of the *month* and all four digits of the *year*.

%I *name*
Publisher or issuer name of an **Rs** block.

%J *name*
Journal name of an **Rs** block.

%N *number*
Issue number (usually for journals) of an **Rs** block.

%O *line*
Optional information of an **Rs** block.

%P *number*
Book or journal page number of an **Rs** block. Conventionally, the argument starts with ‘p.’ for a single page or ‘pp.’ for a range of pages, for example:

```
.%P pp. 42\(en47
```

%Q *name*

Institutional author (school, government, etc.) of an **Rs** block. Multiple institutional authors should each be accorded their own **%Q** line.

%R *name*
Technical report name of an **Rs** block.

%T *title*
Article title of an **Rs** block. This macro may also be used in a non-bibliographical context when referring to article titles.

%U *protocol*://*path*
URI of reference document.

%V *number*
Volume number of an **Rs** block.

Ac Close an **Ao** block. Does not have any tail arguments.

Ad *address*
Memory address. Do not use this for postal addresses.

Examples:
```
.Ad [0,$]
.Ad 0x00000000
```

An **-split** | **-nosplit** | *first_name ... last_name*
Author name. Can be used both for the authors of the program, function, or driver documented in the manual, or for the authors of the manual itself. Requires either the name of an author or one of the following arguments:

-split Start a new output line before each subsequent invocation of **An**.
-nosplit The opposite of **-split**.

The default is **-nosplit**. The effect of selecting either of the **-split** modes ends at the beginning of the *AUTHORS* section. In the *AUTHORS* section, the default is **-nosplit** for the first author listing and **-split** for all other author listings.

Examples:
```
.An -nosplit
.An Kristaps Dzonsons Aq Mt kristaps@bsd.lv
```

Ao *block*

Begin a block enclosed by angle brackets. Does not have any head arguments. This macro is almost never useful. See **Aq** for more details.

Ap Inserts an apostrophe without any surrounding whitespace. This is generally used as a grammatical device when referring to the verb form of a function.

Examples:

```
.Fn execve Ap d
```

Aq *line*

Enclose the rest of the input line in angle brackets. The only important use case is for email addresses. See **Mt** for an example.

Occasionally, it is used for names of characters and keys, for example:

```
Press the
.Aq escape
key to ...
```

For URIs, use **Lk** instead, and **In** for "#include" directives. Never wrap **Ar** in **Aq**.

Since **Aq** usually renders with non-ASCII characters in non-ASCII output modes, do not use it where the ASCII characters ‘<’ and ‘>’ are required as syntax elements. Instead, use these characters directly in such cases, combining them with the macros **Pf**, **Ns**, or **Eo** as needed.

See also **Ao**.

Ar [*placeholder* ...]

Command arguments. If an argument is not provided, the string "file ..." is used as a default.

Examples:

```
.Fl o Ar file
.Ar
.Ar arg1 , arg2 .
```

The arguments to the **Ar** macro are names and placeholders for command arguments; for fixed strings to be passed verbatim as arguments, use **Fl** or **Cm**.

At [*version*]

Formats an AT&T UNIX version. Accepts one optional argument:

v[1-7] | **32v** A version of AT&T UNIX.
III AT&T System III UNIX.
V | **V.[1-4]** A version of AT&T System V UNIX.

Note that these arguments do not begin with a hyphen.

Examples:

```
.At
.At III
.At V.1
```

See also **Bsx**, **Bx**, **Dx**, **Fx**, **Nx**, and **Ox**.

Bc Close a **Bo** block. Does not have any tail arguments.

Bd **-***type* [**-offset** *width*] [**-compact**]

Begin a display block. Display blocks are used to select a different indentation and justification than the one used by the surrounding text. They may contain both macro lines and text lines. By default, a display block is preceded by a vertical space.

The *type* must be one of the following:

-centered Produce one output line from each input line, and center-justify each line. Using this display type is not recommended; many **mdoc** implementations render it poorly.

-filled Change the positions of line breaks to fill each line, and left- and right-justify the resulting block.

-literal Produce one output line from each input line, and do not justify the block at all. Preserve white space as it appears in the input. Always use a constant-width font. Use this for displaying source code.

-ragged Change the positions of line breaks to fill each line, and left-justify the resulting block.

-unfilled The same as **-literal**, but using the same font as for normal text, which is a variable width font if supported by the output device.

The *type* must be provided first. Additional arguments may follow:

-offset *width* Indent the display by the *width*, which may be one of the following:

One of the pre-defined strings **indent**, the width of a standard indentation (six constant width characters); **indent-two**, twice **indent**; **left**, which has no effect; **right**, which justifies to the right margin; or **center**, which aligns around an imagined center axis.

A macro invocation, which selects a predefined width associated with that macro. The most popular is the imaginary macro *Ds*, which resolves to **6n**.

A scaling width as described in roff(7).

An arbitrary string, which indents by the length of this string.

When the argument is missing, **-offset** is ignored.

-compact Do not assert vertical space before the display.

Examples:

```
.Bd -literal -offset indent -compact
  Hello      world.
.Ed
```

See also **D1** and **Dl**.

Bf -emphasis | **-literal** | **-symbolic** | **Em** | **Li** | **Sy**

Change the font mode for a scoped block of text. The **-emphasis** and **Em** argument are equivalent, as are **-symbolic** and **Sy**, and **-literal** and **Li**. Without an argument, this macro does nothing. The font mode continues until broken by a new font mode in a nested scope or **Ef** is encountered.

See also **Li**, **Ef**, **Em**, and **Sy**.

Bk -words

For each macro, keep its output together on the same output line, until the end of the macro or the end of the input line is reached, whichever comes first. Line breaks in text lines are unaffected.

The **-words** argument is required; additional arguments are ignored.

The following example will not break within each **Op** macro line:

```
.Bk -words
.Op Fl f Ar flags
.Op Fl o Ar output
.Ek
```

Be careful in using over-long lines within a keep block! Doing so will clobber the right margin.

Bl **-***type* [**-width** *val*] [**-offset** *val*] [**-compact**] [*col ...*]

Begin a list. Lists consist of items specified using the **It** macro, containing a head or a body or both.

The list *type* is mandatory and must be specified first. The **-width** and **-offset** arguments accept macro names as described for **Bd -offset**, scaling widths as described in roff(7), or use the length of the given string. The **-offset** is a global indentation for the whole list, affecting both item heads and bodies. For those list types supporting it, the **-width** argument requests an additional indentation of item bodies, to be added to the **-offset**. Unless the **-compact** argument is specified, list entries are separated by vertical space.

A list must specify one of the following list types:

-bullet No item heads can be specified, but a bullet will be printed at the head of each item. Item bodies start on the same output line as the bullet and are indented according to the **-width** argument.

-column A columnated list. The **-width** argument has no effect; instead, the string length of each argument specifies the width of one column. If the first line of the body of a **-column** list is not an **It** macro line, **It** contexts spanning one input line each are implied until an **It** macro line is encountered, at which point items start being interpreted as described in the **It** documentation.

-dash Like **-bullet**, except that dashes are used in place of bullets.

-diag Like **-inset**, except that item heads are not parsed for macro invocations. Most often used in the *DIAGNOSTICS* section with error constants in the item heads.

-enum A numbered list. No item heads can be specified. Formatted like **-bullet**, except that cardinal numbers are used in place of bullets, starting at 1.

-hang Like **-tag**, except that the first lines of item bodies are not indented, but follow the item heads like in **-inset** lists.

-hyphen Synonym for **-dash**.

-inset Item bodies follow items heads on the same line, using normal inter-word spacing. Bodies are not indented, and the **-width** argument is ignored.

-item No item heads can be specified, and none are printed. Bodies are not indented, and the **-width** argument is ignored.

-ohang Item bodies start on the line following item heads and are not indented. The **-width** argument is ignored.

-tag Item bodies are indented according to the **-width** argument. When an item head fits inside the indentation, the item body follows this head on the same output line. Otherwise, the body starts on the output line following the head.

Lists may be nested within lists and displays. Nesting of **-column** and **-enum** lists may not be portable.

See also **El** and **It**.

Bo *block*

Begin a block enclosed by square brackets. Does not have any head arguments.

Examples:

```
.Bo 1 ,
.Dv BUFSIZ Bc
```

See also **Bq**.

Bq *line*

Encloses its arguments in square brackets.

Examples:

```
.Bq 1, Dv BUFSIZ
```

Remarks: this macro is sometimes abused to emulate optional arguments for commands; the correct macros to use for this purpose are **Op**, **Oo**, and **Oc**.

See also **Bo**.

Brc Close a **Bro** block. Does not have any tail arguments.

Bro *block*
Begin a block enclosed by curly braces. Does not have any head arguments.

Examples:
.Bro 1 , ... ,
.Va n Brc

See also **Brq**.

Brq *line*
Encloses its arguments in curly braces.

Examples:
.Brq 1, ..., Va n

See also **Bro**.

Bsx [*version*]
Format the BSD/OS version provided as an argument, or a default value if no argument is provided.

Examples:
.Bsx 1.0
.Bsx

See also **At**, **Bx**, **Dx**, **Fx**, **Nx**, and **Ox**.

Bt Supported only for compatibility, do not use this in new manuals. Prints "is currently in beta test."

Bx [*version* [*variant*]]
Format the BSD version provided as an argument, or a default value if no argument is provided.

Examples:
.Bx 4.3 Tahoe
.Bx 4.4
.Bx

See also **At**, **Bsx**, **Dx**, **Fx**, **Nx**, and **Ox**.

Cd *line*

Kernel configuration declaration. This denotes strings accepted by config(8). It is most often used in section 4 manual pages.

Examples:

```
.Cd device le0 at scode?
```

Remarks: this macro is commonly abused by using quoted literals to retain whitespace and align consecutive **Cd** declarations. This practise is discouraged.

Cm *keyword ...*

Command modifiers. Typically used for fixed strings passed as arguments to interactive commands, to commands in interpreted scripts, or to configuration file directives, unless **Fl** is more appropriate.

Examples:

```
.Nm mt Fl f Ar device Cm rewind
.Nm ps Fl o Cm pid , Ns Cm command
.Nm dd Cm if= Ns Ar file1 Cm of= Ns Ar file2
.Ic set Fl o Cm vi
.Ic lookup Cm file bind
.Ic permit Ar identity Op Cm as Ar target
```

D1 *line*

One-line indented display. This is formatted by the default rules and is useful for simple indented statements. It is followed by a newline.

Examples:

```
.D1 Fl abcdefgh
```

See also **Bd** and **Dl**.

Db This macro is obsolete. No replacement is needed. It is ignored by mandoc(1) and groff including its arguments. It was formerly used to toggle a debugging mode.

Dc Close a **Do** block. Does not have any tail arguments.

Dd $Mdocdate$ | *month day, year*

Document date for display in the page footer, by convention the date of the last change. This is the mandatory first macro of any **mdoc** manual.

The *month* is the full English month name, the *day* is an integer number, and the *year* is the full four-digit year.

Other arguments are not portable; the mandoc(1) utility handles them as follows:

- To have the date automatically filled in by the OpenBSD version of cvs(1), the special string "$Mdocdate$" can be given as an argument.
- The traditional, purely numeric man(7) format *year-month-day* is accepted, too.
- If a date string cannot be parsed, it is used verbatim.
- If no date string is given, the current date is used.

Examples:

```
.Dd $Mdocdate$
.Dd $Mdocdate: July 2 2018$
.Dd July 2, 2018
```

See also **Dt** and **Os**.

Dl *line*

One-line indented display. This is formatted as literal text and is useful for commands and invocations. It is followed by a newline.

Examples:

```
.Dl % mandoc mdoc.7 \(ba less
```

See also **Ql**, **Bd -literal**, and **D1**.

Do *block*

Begin a block enclosed by double quotes. Does not have any head arguments.

Examples:

```
.Do
April is the cruellest month
.Dc
\(em T.S. Eliot
```

See also **Dq**.

Dq *line*

Encloses its arguments in "typographic" double-quotes.

Examples:

```
.Dq April is the cruellest month
\(em T.S. Eliot
```

See also **Qq**, **Sq**, and **Do**.

Dt *TITLE section* [*arch*]

Document title for display in the page header. This is the mandatory second macro of any **mdoc** file.

Its arguments are as follows:

TITLE The document's title (name), defaulting to "UNTITLED" if unspecified. To achieve a uniform appearance of page header lines, it should by convention be all caps.

section The manual section. This may be one of **1** (General Commands), **2** (System Calls), **3** (Library Functions), **3p** (Perl Library), **4** (Device Drivers), **5** (File Formats), **6** (Games), **7** (Miscellaneous Information), **8** (System Manager's Manual), or **9** (Kernel Developer's Manual). It should correspond to the manual's filename suffix and defaults to the empty string if unspecified.

arch This specifies the machine architecture a manual page applies to, where relevant, for example **alpha**, **amd64**, **i386**, or **sparc64**. The list of valid architectures varies by operating system.

Examples:

```
.Dt FOO 1
.Dt FOO 9 i386
```

See also **Dd** and **Os**.

Dv *identifier ...*

Defined variables such as preprocessor constants, constant symbols, enumeration values, and so on.

Examples:

```
.Dv NULL
.Dv BUFSIZ
```

.Dv STDOUT_FILENO

See also **Er** and **Ev** for special-purpose constants, **Va** for variable symbols, and **Fd** for listing preprocessor variable definitions in the *SYNOPSIS*.

Dx [*version*]
Format the DragonFly version provided as an argument, or a default value if no argument is provided.

Examples:
.Dx 2.4.1
.Dx

See also **At**, **Bsx**, **Bx**, **Fx**, **Nx**, and **Ox**.

Ec [*closing_delimiter*]
Close a scope started by **Eo**.

The *closing_delimiter* argument is used as the enclosure tail, for example, specifying \(rq will emulate **Dc**.

Ed End a display context started by **Bd**.

Ef End a font mode context started by **Bf**.

Ek End a keep context started by **Bk**.

El End a list context started by **Bl**. See also **It**.

Em *word ...*
Request an italic font. If the output device does not provide that, underline.

This is most often used for stress emphasis (not to be confused with importance, see **Sy**). In the rare cases where none of the semantic markup macros fit, it can also be used for technical terms and placeholders, except that for syntax elements, **Sy** and **Ar** are preferred, respectively.

Examples:
Selected lines are those
.Em not
matching any of the specified patterns.

Some of the functions use a
.Em hold space
to save the pattern space for subsequent retrieval.

See also **No**, **Ql**, and **Sy**.

En *word ...*

This macro is obsolete. Use **Eo** or any of the other enclosure macros.

It encloses its argument in the delimiters specified by the last **Es** macro.

Eo [*opening_delimiter*]

An arbitrary enclosure. The *opening_delimiter* argument is used as the enclosure head, for example, specifying \(lq will emulate **Do**.

Er *identifier ...*

Error constants for definitions of the *errno* libc global variable. This is most often used in section 2 and 3 manual pages.

Examples:

```
.Er EPERM
.Er ENOENT
```

See also **Dv** for general constants.

Es *opening_delimiter closing_delimiter*

This macro is obsolete. Use **Eo** or any of the other enclosure macros.

It takes two arguments, defining the delimiters to be used by subsequent **En** macros.

Ev *identifier ...*

Environmental variables such as those specified in environ(7).

Examples:

```
.Ev DISPLAY
.Ev PATH
```

See also **Dv** for general constants.

Ex -std [*utility ...*]

Insert a standard sentence regarding command exit values of 0 on success and >0 on failure. This is most often used in section 1, 6, and 8 manual pages.

If *utility* is not specified, the document's name set by **Nm** is used. Multiple *utility* arguments are treated as separate utilities.

See also **Rv**.

Fa *argument ...*

Function argument or parameter. Each argument may be a name and a type (recommended for the *SYNOPSIS* section), a name alone (for function invocations), or a type alone (for function prototypes). If both a type and a name are given or if the type consists of multiple words, all words belonging to the same function argument have to be given in a single argument to the **Fa** macro.

This macro is also used to specify the field name of a structure.

Most often, the **Fa** macro is used in the *SYNOPSIS* within **Fo** blocks when documenting multi-line function prototypes. If invoked with multiple arguments, the arguments are separated by a comma. Furthermore, if the following macro is another **Fa**, the last argument will also have a trailing comma.

Examples:

```
.Fa "const char *p"
.Fa "int a" "int b" "int c"
.Fa "char *" size_t
```

See also **Fo**.

Fc End a function context started by **Fo**.

Fd *#directive* [*argument ...*]

Preprocessor directive, in particular for listing it in the *SYNOPSIS*. Historically, it was also used to document include files. The latter usage has been deprecated in favour of **In**.

Examples:

```
.Fd #define sa_handler __sigaction_u.__sa_handler
.Fd #define SIO_MAXNFDS
.Fd #ifdef FS_DEBUG
.Ft void
.Fn dbg_open "const char *"
```

```
.Fd #endif
```

See also *MANUAL STRUCTURE*, **In**, and **Dv**.

Fl [*word* ...]

Command-line flag or option. Used when listing arguments to command-line utilities. For each argument, prints an ASCII hyphen-minus character ‘-’, immediately followed by the argument. If no arguments are provided, a hyphen-minus is printed followed by a space. If the argument is a macro, a hyphen-minus is prefixed to the subsequent macro output.

Examples:

```
.Nm du Op Fl H | L | P
.Nm ls Op Fl 1AaCcdFfgHhikLlmnopqRrSsTtux
.Nm route Cm add Fl inet Ar destination gateway
.Nm locate.updatedb Op Fl \-fcodes Ns = Ns Ar dbfile
.Nm aucat Fl o Fl
.Nm kill Fl Ar signal_number
```

For GNU-sytle long options, escaping the additional hyphen-minus is not strictly required, but may be safer with future versions of GNU troff; see mandoc_char(7) for details.

See also **Cm**.

Fn *funcname* [*argument* ...]

A function name.

Function arguments are surrounded in parenthesis and are delimited by commas. If no arguments are specified, blank parenthesis are output. In the *SYNOPSIS* section, this macro starts a new output line, and a blank line is automatically inserted between function definitions.

Examples:

```
.Fn "int funcname" "int arg0" "int arg1"
.Fn funcname "int arg0"
.Fn funcname arg0

.Ft functype
.Fn funcname
```

When referring to a function documented in another manual page, use **Xr** instead. See also *MANUAL STRUCTURE*, **Fo**, and **Ft**.

Fo *funcname*

Begin a function block. This is a multi-line version of **Fn**.

Invocations usually occur in the following context:

.**Ft** *functype*
.**Fo** *funcname*
.**Fa** "*argtype argname*"
...
.**Fc**

A **Fo** scope is closed by **Fc**.

See also *MANUAL STRUCTURE*, **Fa**, **Fc**, and **Ft**.

Fr *number*

This macro is obsolete. No replacement markup is needed.

It was used to show numerical function return values in an italic font.

Ft *functype*

A function type.

In the *SYNOPSIS* section, a new output line is started after this macro.

Examples:

```
.Ft int
.Ft functype
.Fn funcname
```

See also *MANUAL STRUCTURE*, **Fn**, and **Fo**.

Fx [*version*]

Format the FreeBSD version provided as an argument, or a default value if no argument is provided.

Examples:

```
.Fx 7.1
.Fx
```

See also **At**, **Bsx**, **Bx**, **Dx**, **Nx**, and **Ox**.

Hf *filename*

This macro is not implemented in mandoc(1). It was used to include the contents of a (header) file literally.

Ic *keyword ...*

Internal or interactive command, or configuration instruction in a configuration file. See also **Cm**.

Examples:

```
.Ic :wq
.Ic hash
.Ic alias
```

Note that using **Ql**, **Dl**, or **Bd -literal** is preferred for displaying code samples; the **Ic** macro is used when referring to an individual command name.

In *filename*

The name of an include file. This macro is most often used in section 2, 3, and 9 manual pages.

When invoked as the first macro on an input line in the *SYNOPSIS* section, the argument is displayed in angle brackets and preceded by "#include", and a blank line is inserted in front if there is a preceding function declaration. In other sections, it only encloses its argument in angle brackets and causes no line break.

Examples:

```
.In sys/types.h
```

See also *MANUAL STRUCTURE*.

It [*head*]

A list item. The syntax of this macro depends on the list type.

Lists of type **-hang**, **-ohang**, **-inset**, and **-diag** have the following syntax:

.**It** *args*

Lists of type **-bullet**, **-dash**, **-enum**, **-hyphen** and **-item** have the following syntax:

.**It**

with subsequent lines interpreted within the scope of the **It** until either a closing **El** or another **It**.

The **-tag** list has the following syntax:

.**It** [**args**]

Subsequent lines are interpreted as with **-bullet** and family. The line arguments correspond to the list's left-hand side; body arguments correspond to the list's contents.

The **-column** list is the most complicated. Its syntax is as follows:

.**It** *cell* [**Ta** *cell* ...]
.**It** *cell* [<TAB> *cell* ...]

The arguments consist of one or more lines of text and macros representing a complete table line. Cells within the line are delimited by the special **Ta** block macro or by literal tab characters.

Using literal tabs is strongly discouraged because they are very hard to use correctly and **mdoc** code using them is very hard to read. In particular, a blank character is syntactically significant before and after the literal tab character. If a word precedes or follows the tab without an intervening blank, that word is never interpreted as a macro call, but always output literally.

The tab cell delimiter may only be used within the **It** line itself; on following lines, only the **Ta** macro can be used to delimit cells, and portability requires that **Ta** is called by other macros: some parsers do not recognize it when it appears as the first macro on a line.

Note that quoted strings may span tab-delimited cells on an **It** line. For example,

.It "col1 , <TAB> col2 ," ;

will preserve the whitespace before both commas, but not the whitespace before the semicolon.

See also **Bl**.

Lb lib*name*

Specify a library.

The *name* parameter may be a system library, such as **z** or **pam**, in which case a small library description is printed next to the linker invocation; or a custom library, in which case the library name is printed in quotes. This is most commonly used in the *SYNOPSIS* section as described in

MANUAL STRUCTURE.

Examples:

```
.Lb libz
.Lb libmandoc
```

Li *word ...*

Request a typewriter (literal) font. Deprecated because on terminal output devices, this is usually indistinguishable from normal text. For literal displays, use **Ql** (in-line), **Dl** (single line), or **Bd -literal** (multi-line) instead.

Lk *uri* [*display_name*]

Format a hyperlink.

Examples:

```
.Lk https://bsd.lv "The BSD.lv Project"
.Lk https://bsd.lv
```

See also **Mt**.

Lp Deprecated synonym for **Pp**.

Ms *name*

Display a mathematical symbol.

Examples:

```
.Ms sigma
.Ms aleph
```

Mt *localpart@domain*

Format a "mailto:" hyperlink.

Examples:

```
.Mt discuss@manpages.bsd.lv
.An Kristaps Dzonsons Aq Mt kristaps@bsd.lv
```

Nd *line*

A one line description of the manual’s content. This is the mandatory last macro of the *NAME* section and not appropriate for other sections.

Examples:

.**Nd mdoc language reference**
.**Nd format and display UNIX manuals**

The **Nd** macro technically accepts child macros and terminates with a subsequent **Sh** invocation. Do not assume this behaviour: some whatis(1) database generators are not smart enough to parse more than the line arguments and will display macros verbatim.

See also **Nm**.

Nm [*name*]

The name of the manual page, or -- in particular in section 1, 6, and 8 pages -- of an additional command or feature documented in the manual page. When first invoked, the **Nm** macro expects a single argument, the name of the manual page. Usually, the first invocation happens in the *NAME* section of the page. The specified name will be remembered and used whenever the macro is called again without arguments later in the page. The **Nm** macro uses *Block full-implicit* semantics when invoked as the first macro on an input line in the *SYNOPSIS* section; otherwise, it uses ordinary *In-line* semantics.

Examples:

.Sh SYNOPSIS
.Nm cat
.Op Fl benstuv
.Op Ar

In the *SYNOPSIS* of section 2, 3 and 9 manual pages, use the **Fn** macro rather than **Nm** to mark up the name of the manual page.

No *word* ...

Normal text. Closes the scope of any preceding in-line macro. When used after physical formatting macros like **Em** or **Sy**, switches back to the standard font face and weight. Can also be used to embed plain text strings in macro lines using semantic annotation macros.

Examples:

.Em italic , Sy bold , No and roman

.Sm off
.Cm :C No / Ar pattern No / Ar replacement No /
.Sm on

See also **Em**, **Ql**, and **Sy**.

Ns Suppress a space between the output of the preceding macro and the following text or macro. Following invocation, input is interpreted as normal text just like after an **No** macro.

This has no effect when invoked at the start of a macro line.

Examples:

```
.Ar name Ns = Ns Ar value
.Cm :M Ns Ar pattern
.Fl o Ns Ar output
```

See also **No** and **Sm**.

Nx [*version*]

Format the NetBSD version provided as an argument, or a default value if no argument is provided.

Examples:

```
.Nx 5.01
.Nx
```

See also **At**, **Bsx**, **Bx**, **Dx**, **Fx**, and **Ox**.

Oc Close multi-line **Oo** context.

Oo *block*

Multi-line version of **Op**.

Examples:

```
.Oo
.Op Fl flag Ns Ar value
.Oc
```

Op *line*

Optional part of a command line. Prints the argument(s) in brackets. This is most often used in the *SYNOPSIS* section of section 1 and 8 manual pages.

Examples:

```
.Op Fl a Ar b
.Op Ar a | b
```

See also **Oo**.

Os [*system* [*version*]]

Operating system version for display in the page footer. This is the mandatory third macro of any **mdoc** file.

The optional *system* parameter specifies the relevant operating system or environment. It is suggested to leave it unspecified, in which case mandoc(1) uses its **-Ios** argument or, if that isn't specified either, *sysname* and *release* as returned by uname(3).

Examples:

```
.Os
.Os KTH/CSC/TCS
.Os BSD 4.3
```

See also **Dd** and **Dt**.

Ot *functype*

This macro is obsolete. Use **Ft** instead; with mandoc(1), both have the same effect.

Historical **mdoc** packages described it as "old function type (FORTRAN)".

Ox [*version*]

Format the OpenBSD version provided as an argument, or a default value if no argument is provided.

Examples:

```
.Ox 4.5
.Ox
```

See also **At**, **Bsx**, **Bx**, **Dx**, **Fx**, and **Nx**.

Pa *name* ...

An absolute or relative file system path, or a file or directory name. If an argument is not provided, the character '~' is used as a default.

Examples:

```
.Pa /usr/bin/mandoc
.Pa /usr/share/man/man7/mdoc.7
```

See also **Lk**.

Pc Close parenthesised context opened by **Po**.

Pf *prefix macro* [*argument* ...]
Removes the space between its argument and the following macro. It is equivalent to:

No \&*prefix* **Ns** *macro* [*argument* ...]

The *prefix* argument is not parsed for macro names or delimiters, but used verbatim as if it were escaped.

Examples:
.Pf $ Ar variable_name
.Pf . Ar macro_name
.Pf 0x Ar hex_digits

See also **Ns** and **Sm**.

Po *block*
Multi-line version of **Pq**.

Pp Break a paragraph. This will assert vertical space between prior and subsequent macros and/or text.

Paragraph breaks are not needed before or after **Sh** or **Ss** macros or before displays (**Bd** *line*) or lists (**Bl**) unless the **-compact** flag is given.

Pq *line*
Parenthesised enclosure.

See also **Po**.

Qc Close quoted context opened by **Qo**.

Ql *line*
In-line literal display. This can be used for complete command invocations and for multi-word code examples when an indented display is not desired.

See also **Dl** and **Bd -literal**.

Qo *block*
Multi-line version of **Qq**.

Qq *line*
Encloses its arguments in "typewriter" double-quotes. Consider using **Dq**.

See also **Dq**, **Sq**, and **Qo**.

Re Close an **Rs** block. Does not have any tail arguments.

Rs Begin a bibliographic ("reference") block. Does not have any head arguments. The block macro may only contain **%A**, **%B**, **%C**, **%D**, **%I**, **%J**, **%N**, **%O**, **%P**, **%Q**, **%R**, **%T**, **%U**, and **%V** child macros (at least one must be specified).

Examples:

```
.Rs
.%A J. E. Hopcroft
.%A J. D. Ullman
.%B Introduction to Automata Theory, Languages, and Computation
.%I Addison-Wesley
.%C Reading, Massachusetts
.%D 1979
.Re
```

If an **Rs** block is used within a SEE ALSO section, a vertical space is asserted before the rendered output, else the block continues on the current line.

Rv -std [*function* ...]
Insert a standard sentence regarding a function call’s return value of 0 on success and -1 on error, with the *errno* libc global variable set on error.

If *function* is not specified, the document’s name set by **Nm** is used. Multiple *function* arguments are treated as separate functions.

See also **Ex**.

Sc Close single-quoted context opened by **So**.

Sh *TITLE LINE*
Begin a new section. For a list of conventional manual sections, see *MANUAL STRUCTURE*.

These sections should be used unless it’s absolutely necessary that custom sections be used.

Section names should be unique so that they may be keyed by **Sx**. Although this macro is parsed, it should not consist of child node or it may not be linked with **Sx**.

See also **Pp**, **Ss**, and **Sx**.

Sm [**on** | **off**]
Switches the spacing mode for output generated from macros.

By default, spacing is **on**. When switched **off**, no white space is inserted between macro arguments and between the output generated from adjacent macros, but text lines still get normal spacing between words and sentences.

When called without an argument, the **Sm** macro toggles the spacing mode. Using this is not recommended because it makes the code harder to read.

So *block*
Multi-line version of **Sq**.

Sq *line*
Encloses its arguments in ‘typewriter’ single-quotes.

See also **Dq**, **Qq**, and **So**.

Ss *Title line*
Begin a new subsection. Unlike with **Sh**, there is no convention for the naming of subsections. Except *DESCRIPTION*, the conventional sections described in *MANUAL STRUCTURE* rarely have subsections.

Sub-section names should be unique so that they may be keyed by **Sx**. Although this macro is parsed, it should not consist of child node or it may not be linked with **Sx**.

See also **Pp**, **Sh**, and **Sx**.

St -*abbreviation*
Replace an abbreviation for a standard with the full form. The following standards are recognised. Where multiple lines are given without a blank line in between, they all refer to the same standard, and using the first form is recommended.

C language standards

-ansiC	ANSI X3.159-1989 ("ANSI C89")
-ansiC-89	ANSI X3.159-1989 ("ANSI C89")
-isoC	ISO/IEC 9899:1990 ("ISO C90")
-isoC-90	ISO/IEC 9899:1990 ("ISO C90") The original C standard.
-isoC-amd1	ISO/IEC 9899/AMD1:1995 ("ISO C90, Amendment 1")
-isoC-tcor1	ISO/IEC 9899/TCOR1:1994 ("ISO C90, Technical Corrigendum 1")
-isoC-tcor2	ISO/IEC 9899/TCOR2:1995 ("ISO C90, Technical Corrigendum 2")
-isoC-99	ISO/IEC 9899:1999 ("ISO C99") The second major version of the C language standard.
-isoC-2011	ISO/IEC 9899:2011 ("ISO C11") The third major version of the C language standard.

POSIX.1 before the Single UNIX Specification

-p1003.1-88	IEEE Std 1003.1-1988 ("POSIX.1")
-p1003.1	IEEE Std 1003.1 ("POSIX.1") The original POSIX standard, based on ANSI C.
-p1003.1-90	IEEE Std 1003.1-1990 ("POSIX.1")
-iso9945-1-90	ISO/IEC 9945-1:1990 ("POSIX.1") The first update of POSIX.1.
-p1003.1b-93	IEEE Std 1003.1b-1993 ("POSIX.1b")
-p1003.1b	IEEE Std 1003.1b ("POSIX.1b") Real-time extensions.
-p1003.1c-95	IEEE Std 1003.1c-1995 ("POSIX.1c") POSIX thread interfaces.
-p1003.1i-95	IEEE Std 1003.1i-1995 ("POSIX.1i") Technical Corrigendum.

-p1003.1-96 ISO/IEC 9945-1:1996 ("POSIX.1")
-iso9945-1-96 ISO/IEC 9945-1:1996 ("POSIX.1")
Includes POSIX.1-1990, 1b, 1c, and 1i.

X/Open Portability Guide version 4 and related standards

-xpg3 X/Open Portability Guide Issue 3 ("XPG3")
An XPG4 precursor, published in 1989.

-p1003.2 IEEE Std 1003.2 ("POSIX.2")
-p1003.2-92 IEEE Std 1003.2-1992 ("POSIX.2")
-iso9945-2-93 ISO/IEC 9945-2:1993 ("POSIX.2")
An XCU4 precursor.

-p1003.2a-92 IEEE Std 1003.2a-1992 ("POSIX.2")
Updates to POSIX.2.

-xpg4 X/Open Portability Guide Issue 4 ("XPG4")
Based on POSIX.1 and POSIX.2, published in 1992.

Single UNIX Specification version 1 and related standards

-susv1 Version 1 of the Single UNIX Specification ("SUSv1")
-xpg4.2 X/Open Portability Guide Issue 4, Version 2 ("XPG4.2")
This standard was published in 1994. It was used as the basis for UNIX 95 certification. The following three refer to parts of it.

-xsh4.2 X/Open System Interfaces and Headers Issue 4, Version 2 ("XSH4.2")

-xcurses4.2 X/Open Curses Issue 4, Version 2 ("XCURSES4.2")

-p1003.1g-2000 IEEE Std 1003.1g-2000 ("POSIX.1g")
Networking APIs, including sockets.

-svid4 System V Interface Definition, Fourth Edition ("SVID4"),
Published in 1995.

Single UNIX Specification version 2 and related standards

-susv2 Version 2 of the Single UNIX Specification ("SUSv2") This Standard was

published in 1997 and is also called X/Open Portability Guide version 5. It was used as the basis for UNIX 98 certification. The following refer to parts of it.

-xbd5 X/Open Base Definitions Issue 5 ("XBD5")

-xsh5 X/Open System Interfaces and Headers Issue 5 ("XSH5")

-xcu5 X/Open Commands and Utilities Issue 5 ("XCU5")

-xns5 X/Open Networking Services Issue 5 ("XNS5")
-xns5.2 X/Open Networking Services Issue 5.2 ("XNS5.2")

Single UNIX Specification version 3

-p1003.1-2001 IEEE Std 1003.1-2001 ("POSIX.1")
-susv3 Version 3 of the Single UNIX Specification ("SUSv3")
This standard is based on C99, SUSv2, POSIX.1-1996, 1d, and 1j. It is also called X/Open Portability Guide version 6. It is used as the basis for UNIX 03 certification.

-p1003.1-2004 IEEE Std 1003.1-2004 ("POSIX.1")
The second and last Technical Corrigendum.

Single UNIX Specification version 4

-p1003.1-2008 IEEE Std 1003.1-2008 ("POSIX.1")
-susv4 Version 4 of the Single UNIX Specification ("SUSv4")
This standard is also called X/Open Portability Guide version 7.

Other standards

-ieee754 IEEE Std 754-1985
Floating-point arithmetic.

-iso8601 ISO 8601
Representation of dates and times, published in 1988.

-iso8802-3 ISO 8802-3: 1989
Ethernet local area networks.

-ieee1275-94 IEEE Std 1275-1994 ("Open Firmware")

Sx *Title line*

Reference a section or subsection in the same manual page. The referenced section or subsection name must be identical to the enclosed argument, including whitespace.

Examples:

```
.Sx MANUAL STRUCTURE
```

See also **Sh** and **Ss**.

Sy *word ...*

Request a boldface font.

This is most often used to indicate importance or seriousness (not to be confused with stress emphasis, see **Em**). When none of the semantic macros fit, it is also adequate for syntax elements that have to be given or that appear verbatim.

Examples:

```
.Sy Warning :
If
.Sy s
appears in the owner permissions, set-user-ID mode is set.
This utility replaces the former
.Sy dumpdir
program.
```

See also **Em**, **No**, and **Ql**.

Ta Table cell separator in **Bl -column** lists; can only be used below **It**.

Tg [*term*]

Announce that the next input line starts a definition of the *term*. This macro must appear alone on its own input line. The argument defaults to the first argument of the first macro on the next line. The argument may not contain whitespace characters, not even when it is quoted. This macro is a mandoc(1) extension and is typically ignored by other formatters.

When viewing terminal output with less(1), the interactive **:t** command can be used to go to the definition of the *term* as described for the MANPAGER variable in man(1); when producing HTML output, a fragment identifier (**id** attribute) is generated, to be used for deep linking to this

place of the document.

In most cases, adding a **Tg** macro would be redundant because mandoc(1) is able to automatically tag most definitions. This macro is intended for cases where automatic tagging of a *term* is unsatisfactory, for example if a definition is not tagged automatically (false negative) or if places are tagged that do not define the *term* (false positives). When there is at least one **Tg** macro for a *term*, no other places are automatically marked as definitions of that *term*.

Tn *word* ...
Supported only for compatibility, do not use this in new manuals. Even though the macro name ("tradename") suggests a semantic function, historic usage is inconsistent, mostly using it as a presentation-level macro to request a small caps font.

Ud Supported only for compatibility, do not use this in new manuals. Prints out "currently under development."

Ux Supported only for compatibility, do not use this in new manuals. Prints out "UNIX".

Va [*type*] *identifier* ...
A variable name.

Examples:
.Va foo
.Va const char *bar;

For function arguments and parameters, use **Fa** instead. For declarations of global variables in the *SYNOPSIS* section, use **Vt**.

Vt *type* [*identifier*]
A variable type.

This is also used for indicating global variables in the *SYNOPSIS* section, in which case a variable name is also specified. Note that it accepts *Block partial-implicit* syntax when invoked as the first macro on an input line in the *SYNOPSIS* section, else it accepts ordinary *In-line* syntax. In the former case, this macro starts a new output line, and a blank line is inserted in front if there is a preceding function definition or include directive.

Examples:
.Vt unsigned char
.Vt extern const char * const sys_signame[] ;

For parameters in function prototypes, use **Fa** instead, for function return types **Ft**, and for variable names outside the *SYNOPSIS* section **Va**, even when including a type with the name. See also *MANUAL STRUCTURE*.

Xc Close a scope opened by **Xo**.

Xo *block*
Extend the header of an **It** macro or the body of a partial-implicit block macro beyond the end of the input line. This macro originally existed to work around the 9-argument limit of historic roff(7).

Xr *name section*
Link to another manual ("cross-reference").

Cross reference the *name* and *section* number of another man page.

Examples:
.Xr mandoc 1
.Xr mandoc 1 ;
.Xr mandoc 1 Ns s behaviour

MACRO SYNTAX

The syntax of a macro depends on its classification. In this section, ‘-arg’ refers to macro arguments, which may be followed by zero or more ‘parm’ parameters; ‘Yo’ opens the scope of a macro; and if specified, ‘Yc’ closes it out.

The *Callable* column indicates that the macro may also be called by passing its name as an argument to another macro. For example, ‘.Op Fl O Ar file’ produces ‘[**-O** *file*]’. To prevent a macro call and render the macro name literally, escape it by prepending a zero-width space, ‘\&’. For example, ‘Op \&Fl O’ produces ‘[Fl O]’. If a macro is not callable but its name appears as an argument to another macro, it is interpreted as opaque text. For example, ‘.Fl Sh’ produces ‘**-Sh**’.

The *Parsed* column indicates whether the macro may call other macros by receiving their names as arguments. If a macro is not parsed but the name of another macro appears as an argument, it is interpreted as opaque text.

The *Scope* column, if applicable, describes closure rules.

Block full-explicit

Multi-line scope closed by an explicit closing macro. All macros contains bodies; only **Bf** and

(optionally) **Bl** contain a head.

.Yo [-arg [parm...]] [head...]
[body...]
.Yc

Macro	*Callable*	*Parsed*	*Scope*
Bd	No	No	closed by **Ed**
Bf	No	No	closed by **Ef**
Bk	No	No	closed by **Ek**
Bl	No	No	closed by **El**
Ed	No	No	opened by **Bd**
Ef	No	No	opened by **Bf**
Ek	No	No	opened by **Bk**
El	No	No	opened by **Bl**

Block full-implicit

Multi-line scope closed by end-of-file or implicitly by another macro. All macros have bodies; some (**It -bullet**, **-hyphen**, **-dash**, **-enum**, **-item**) don't have heads; only one (**It** in **Bl -column**) has multiple heads.

.Yo [-arg [parm...]] [head... [Ta head...]]
[body...]

Macro	*Callable*	*Parsed*	*Scope*
It	No	Yes	closed by **It**, **El**
Nd	No	No	closed by **Sh**
Nm	No	Yes	closed by **Nm**, **Sh**, **Ss**
Sh	No	Yes	closed by **Sh**
Ss	No	Yes	closed by **Sh**, **Ss**

Note that the **Nm** macro is a *Block full-implicit* macro only when invoked as the first macro in a *SYNOPSIS* section line, else it is *In-line*.

Block partial-explicit

Like block full-explicit, but also with single-line scope. Each has at least a body and, in limited circumstances, a head (**Fo**, **Eo**) and/or tail (**Ec**).

.Yo [-arg [parm...]] [head...]
[body...]
.Yc [tail...]

.Yo [-arg [parm...]] [head...] [body...] Yc [tail...]

Macro	*Callable*	*Parsed*	*Scope*
Ac	Yes	Yes	opened by **Ao**
Ao	Yes	Yes	closed by **Ac**
Bc	Yes	Yes	closed by **Bo**
Bo	Yes	Yes	opened by **Bc**
Brc	Yes	Yes	opened by **Bro**
Bro	Yes	Yes	closed by **Brc**
Dc	Yes	Yes	opened by **Do**
Do	Yes	Yes	closed by **Dc**
Ec	Yes	Yes	opened by **Eo**
Eo	Yes	Yes	closed by **Ec**
Fc	Yes	Yes	opened by **Fo**
Fo	No	No	closed by **Fc**
Oc	Yes	Yes	closed by **Oo**
Oo	Yes	Yes	opened by **Oc**
Pc	Yes	Yes	closed by **Po**
Po	Yes	Yes	opened by **Pc**
Qc	Yes	Yes	opened by **Oo**
Qo	Yes	Yes	closed by **Oc**
Re	No	No	opened by **Rs**
Rs	No	No	closed by **Re**
Sc	Yes	Yes	opened by **So**
So	Yes	Yes	closed by **Sc**
Xc	Yes	Yes	opened by **Xo**
Xo	Yes	Yes	closed by **Xc**

Block partial-implicit

Like block full-implicit, but with single-line scope closed by the end of the line.

.Yo [-arg [val...]] [body...] [res...]

Macro	*Callable*	*Parsed*
Aq	Yes	Yes
Bq	Yes	Yes
Brq	Yes	Yes
D1	No	Yes
Dl	No	Yes
Dq	Yes	Yes

En	Yes	Yes
Op	Yes	Yes
Pq	Yes	Yes
Ql	Yes	Yes
Qq	Yes	Yes
Sq	Yes	Yes
Vt	Yes	Yes

Note that the **Vt** macro is a *Block partial-implicit* only when invoked as the first macro in a *SYNOPSIS* section line, else it is *In-line*.

Special block macro

The **Ta** macro can only be used below **It** in **Bl -column** lists. It delimits blocks representing table cells; these blocks have bodies, but no heads.

Macro	*Callable*	*Parsed*	*Scope*
Ta	Yes	Yes	closed by **Ta**, **It**

In-line

Closed by the end of the line, fixed argument lengths, and/or subsequent macros. In-line macros have only text children. If a number (or inequality) of arguments is (n), then the macro accepts an arbitrary number of arguments.

.Yo [-arg [val...]] [args...] [res...]

.Yo [-arg [val...]] [args...] Yc...

.Yo [-arg [val...]] arg0 arg1 argN

Macro	*Callable*	*Parsed*	*Arguments*
%A	No	No	>0
%B	No	No	>0
%C	No	No	>0
%D	No	No	>0
%I	No	No	>0
%J	No	No	>0
%N	No	No	>0
%O	No	No	>0
%P	No	No	>0
%Q	No	No	>0

%R	No	No	>0
%T	No	No	>0
%U	No	No	>0
%V	No	No	>0
Ad	Yes	Yes	>0
An	Yes	Yes	>0
Ap	Yes	Yes	0
Ar	Yes	Yes	n
At	Yes	Yes	1
Bsx	Yes	Yes	n
Bt	No	No	0
Bx	Yes	Yes	n
Cd	Yes	Yes	>0
Cm	Yes	Yes	>0
Db	No	No	1
Dd	No	No	n
Dt	No	No	n
Dv	Yes	Yes	>0
Dx	Yes	Yes	n
Em	Yes	Yes	>0
Er	Yes	Yes	>0
Es	Yes	Yes	2
Ev	Yes	Yes	>0
Ex	No	No	n
Fa	Yes	Yes	>0
Fd	No	No	>0
Fl	Yes	Yes	n
Fn	Yes	Yes	>0
Fr	Yes	Yes	>0
Ft	Yes	Yes	>0
Fx	Yes	Yes	n
Hf	No	No	n
Ic	Yes	Yes	>0
In	No	No	1
Lb	No	No	1
Li	Yes	Yes	>0
Lk	Yes	Yes	>0
Lp	No	No	0
Ms	Yes	Yes	>0
Mt	Yes	Yes	>0

Nm	Yes	Yes	n
No	Yes	Yes	>0
Ns	Yes	Yes	0
Nx	Yes	Yes	n
Os	No	No	n
Ot	Yes	Yes	>0
Ox	Yes	Yes	n
Pa	Yes	Yes	n
Pf	Yes	Yes	1
Pp	No	No	0
Rv	No	No	n
Sm	No	No	<2
St	No	Yes	1
Sx	Yes	Yes	>0
Sy	Yes	Yes	>0
Tg	No	No	<2
Tn	Yes	Yes	>0
Ud	No	No	0
Ux	Yes	Yes	n
Va	Yes	Yes	n
Vt	Yes	Yes	>0
Xr	Yes	Yes	2

Delimiters

When a macro argument consists of one single input character considered as a delimiter, the argument gets special handling. This does not apply when delimiters appear in arguments containing more than one character. Consequently, to prevent special handling and just handle it like any other argument, a delimiter can be escaped by prepending a zero-width space (‘\&’). In text lines, delimiters never need escaping, but may be used as normal punctuation.

For many macros, when the leading arguments are opening delimiters, these delimiters are put before the macro scope, and when the trailing arguments are closing delimiters, these delimiters are put after the macro scope. Spacing is suppressed after opening delimiters and before closing delimiters. For example,

```
.Aq ( [ word ] ) .
```

renders as:

([<word>]).

Opening delimiters are:

(left parenthesis
[left bracket

Closing delimiters are:

. period
, comma
: colon
; semicolon
) right parenthesis
] right bracket
? question mark
! exclamation mark

Note that even a period preceded by a backslash (‘\.’) gets this special handling; use ‘\&.’ to prevent that.

Many in-line macros interrupt their scope when they encounter delimiters, and resume their scope when more arguments follow that are not delimiters. For example,

```
.Fl a ( b | c \*(Ba d ) e
```

renders as:

-a (**-b** | **-c** | **-d**) **-e**

This applies to both opening and closing delimiters, and also to the middle delimiter, which does not suppress spacing:

| vertical bar

As a special case, the predefined string *(Ba is handled and rendered in the same way as a plain ‘|’ character. Using this predefined string is not recommended in new manuals.

Appending a zero-width space (‘\&’) to the end of an input line is also useful to prevent the interpretation of a trailing period, exclamation or question mark as the end of a sentence, for example when an abbreviation happens to occur at the end of a text or macro input line.

Font handling

In **mdoc** documents, usage of semantic markup is recommended in order to have proper fonts automatically selected; only when no fitting semantic markup is available, consider falling back to *Physical markup* macros. Whenever any **mdoc** macro switches the roff(7) font mode, it will automatically restore the previous font when exiting its scope. Manually switching the font using the roff(7) ‘\f’ font escape sequences is never required.

COMPATIBILITY

This section provides an incomplete list of compatibility issues between mandoc and GNU troff ("groff").

The following problematic behaviour is found in groff:

- **Pa** does not format its arguments when used in the FILES section under certain list types.
- **Ta** can only be called by other macros, but not at the beginning of a line.
- ‘\f’ (font face) and ‘\F’ (font family face) *Text Decoration* escapes behave irregularly when specified within line-macro scopes.
- Negative scaling units return to prior lines. Instead, mandoc truncates them to zero.

The following features are unimplemented in mandoc:

- **Bd -file** *file* is unsupported for security reasons.
- **Bd -filled** does not adjust the right margin, but is an alias for **Bd -ragged**.
- **Bd -literal** does not use a literal font, but is an alias for **Bd -unfilled**.
- **Bd -offset center** and **-offset right** don’t work. Groff does not implement centered and flush-right rendering either, but produces large indentations.

SEE ALSO

man(1), mandoc(1), eqn(7), man(7), mandoc_char(7), roff(7), tbl(7)

The web page *extended documentation for the mdoc language*: **https://mandoc.bsd.lv/mdoc/** provides a few tutorial-style pages for beginners, an extensive style guide for advanced authors, and an alphabetic index helping to choose the best macros for various kinds of content.

The manual page *groff_mdoc(7)*: **https://man.voidlinux.org/groff_mdoc** contained in the "groff" package documents exactly the same language in a somewhat different style.

HISTORY

The **mdoc** language first appeared as a troff macro package in 4.4BSD. It was later significantly updated by Werner Lemberg and Ruslan Ermilov in groff-1.17. The standalone implementation that is part of

the mandoc(1) utility written by Kristaps Dzonsons appeared in OpenBSD 4.6.

AUTHORS

The **mdoc** reference was written by Kristaps Dzonsons <*kristaps@bsd.lv*>.