## NAME

**mdoc** - mdoc language reference

## DESCRIPTION

The **mdoc** language is used to format BSD UNIX manuals. In this reference document, we describe its syntax, structure, and usage. Our reference implementation is mandoc; the *COMPATIBILITY* section describes compatibility with other troff -mdoc implementations.

An **mdoc** document follows simple rules: lines beginning with the control character '.' are parsed for macros. Other lines are interpreted within the scope of prior macros:

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

## LANGUAGE SYNTAX

**mdoc** documents may contain only graphable 7-bit ASCII characters, the space character, and, in certain circumstances, the tab character. All manuals must have UNIX line terminators.

### Comments

Text following a '\"', whether in a macro or free-form text line, is ignored to the end of line. A macro line with only a control character and comment escape, '.\"', is also ignored. Macro lines with only a control character and optionally whitespace are stripped from input.

### Reserved Characters

Within a macro line, the following characters are reserved:

| | |
|---|---|
| . | (period) |
| , | (comma) |
| : | (colon) |
| ; | (semicolon) |
| ( | (left-parenthesis) |
| ) | (right-parenthesis) |
| [ | (left-bracket) |
| ] | (right-bracket) |
| ? | (question) |
| ! | (exclamation) |
| \| | (vertical bar) |

Use of reserved characters is described in *MACRO SYNTAX*. For general use in macro lines, these characters must either be escaped with a non-breaking space ('\&') or, if applicable, an appropriate escape sequence used.

### Special Characters

Special characters may occur in both macro and free-form lines. Sequences begin with the escape character '\' followed by either an open-parenthesis '(' for two-character sequences; an open-bracket '[' for n-character sequences (terminated at a close-bracket ']'); or a single one-character sequence. See mandoc_char(7) for a complete list. Examples include '\(em' (em-dash) and '\e' (back-slash).

### Text Decoration

Terms may be text-decorated using the '\f' escape followed by an indicator: B (bold), I, (italic), R (Roman), or P (revert to previous mode):

```
\fBbold\fR \fIitalic\fP
```

A numerical representation 3, 2, or 1 (bold, italic, and Roman, respectively) may be used instead. A text decoration is valid within the current font scope only: if a macro opens a font scope alongside its own scope, such as *Bf* **Sy**, in-scope invocations of '\f' are only valid within the font scope of the macro. If '\f' is specified outside of any font scope, such as in unenclosed, free-form text, it will affect the remainder of the document.

Text may also be sized with the '\s' escape, whose syntax is one of '\s+-n' for one-digit numerals; '\s(+-nn' or '\s+-(nn' for two-digit numerals; and '\s[+-N]', '\s+-[N]', '\s'+-N", or '\s+-'N" for arbitrary-digit numerals:

```
\s+1bigger\s-1
\s[+10]much bigger\s[-10]
\s+(10much bigger\s-(10
\s+'100'much much bigger\s-'100'
```

Note these forms are *not* recommended for **mdoc**, which encourages semantic annotation.

### Predefined Strings

Historically, groff(1) also defined a set of package-specific "predefined strings", which, like *Special Characters*, mark special output characters and strings by way of input codes. Predefined strings are escaped with the slash-asterisk, '*': single-character '*X', two-character '*(XX', and N-character '*[N]'. See mandoc_char(7) for a complete list. Examples include '*(Am' (ampersand) and '*(Ba' (vertical bar).

### Whitespace

Whitespace consists of the space character. In free-form lines, whitespace is preserved within a line; unescaped trailing spaces are stripped from input (unless in a literal context). Blank free-form lines, which may include whitespace, are only permitted within literal contexts.

In macro lines, whitespace delimits arguments and is discarded. If arguments are quoted, whitespace within the quotes is retained.

### Quotation

Macro arguments may be quoted with a double-quote to group space-delimited terms or to retain blocks of whitespace. A quoted argument begins with a double-quote preceded by whitespace. The next double-quote not pair-wise adjacent to another double-quote terminates the literal, regardless of surrounding whitespace.

This produces tokens 'a"', 'b c', 'de', and 'fg"'. Note that any quoted term, be it argument or macro, is indiscriminately considered literal text. Thus, the following produces 'Em a':

```
.Em "Em a"
```

In free-form mode, quotes are regarded as opaque text.

### Dates

There are several macros in **mdoc** that require a date argument. The canonical form for dates is the American format:

> **Month Day**, **Year**

The **Day** value is an optionally zero-padded numeral. The **Month** value is the full month name. The **Year** value is the full four-digit year.

Reduced form dates are broken-down canonical form dates:

**Month**, **Year**
**Year**

Some examples of valid dates follow:

May, 2009 (reduced form)
2009 (reduced form)
May 20, 2009 (canonical form)

### Scaling Widths

Many macros support scaled widths for their arguments, such as stipulating a two-inch list indentation with the following:

.Bl -tag -width 2i

The syntax for scaled widths is '[+-]?[0-9]*.[0-9]*[:unit:]', where a decimal must be preceded or proceeded by at least one digit. Negative numbers, while accepted, are truncated to zero. The following scaling units are accepted:

| | |
|---|---|
| c | centimetre |
| i | inch |
| P | pica (~1/6 inch) |
| p | point (~1/72 inch) |
| f | synonym for 'u' |
| v | default vertical span |
| m | width of rendered 'm' (em) character |
| n | width of rendered 'n' (en) character |
| u | default horizontal span |
| M | mini-em (~1/100 em) |

Using anything other than 'm', 'n', 'u', or 'v' is necessarily non-portable across output media. See *COMPATIBILITY*.

### Sentence Spacing

When composing a manual, make sure that your sentences end at the end of a line. By doing so, front-ends will be able to apply the proper amount of spacing after the end of sentence (unescaped) period, exclamation mark, or question mark followed by zero or more non-sentence closing delimiters (')', ']', '"', '"').

The proper spacing is also intelligently preserved if a sentence ends at the boundary of a macro line, e.g.,

Xr mandoc 1 .
Fl T Ns Cm ascii .

## MANUAL STRUCTURE

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

The prologue, which consists of (in order) the *Dd*, *Dt*, and *Os* macros, 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:

```
.Dd $Mdocdate$
.Dt mdoc 7
.Os
.Sh NAME
.Nm foo
.Nd a description goes here
.\" The next is for sections 2, 3, & 9 only.
.\" .Sh LIBRARY
.Sh SYNOPSIS
.Nm foo
.Op Fl options
.Ar
.Sh DESCRIPTION
The
.Nm
utility processes files ...
.\" .Sh IMPLEMENTATION NOTES
.\" The next is for sections 2, 3, & 9 only.
.\" .Sh RETURN VALUES
.\" The next is for sections 1, 6, 7, & 8 only.
.\" .Sh ENVIRONMENT
.\" .Sh FILES
.\" The next is for sections 1 & 8 only.
.\" .Sh EXIT STATUS
.\" .Sh EXAMPLES
.\" The next is for sections 1, 4, 6, 7, & 8 only.
.\" .Sh DIAGNOSTICS
.\" The next is for sections 2, 3, & 9 only.
.\" .Sh ERRORS
.\" .Sh SEE ALSO
.\" .Xr foobar 1
.\" .Sh STANDARDS
.\" .Sh HISTORY
.\" .Sh AUTHORS
.\" .Sh CAVEATS
.\" .Sh BUGS
.\" .Sh SECURITY CONSIDERATIONS
```

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

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

```
.Nm name0
.Nm name1
.Nm name2
.Nd a short description
```

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

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

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

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.

*DESCRIPTION*
This expands upon the brief, one-line description in *NAME*. It usually contains a break-down of the options (if documenting a command), such as:

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

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

*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 is the dual of *EXIT STATUS*, which is used for commands. It documents the return values of functions in sections 2, 3, and 9.

See *Rv*.

*ENVIRONMENT*
Documents any usages of environment variables, e.g., environ(7).

See *Ev*.

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

See *Pa*.

*EXIT STATUS*
Command exit status for section 1, 6, and 8 manuals. This section is the dual of *RETURN VALUES*, which is used for functions. 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 doubly sure that your examples work properly!

*DIAGNOSTICS*
Documents error conditions. This is most useful in section 4 manuals. 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 error handling in sections 2, 3, 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.

See *Xr*.

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

See *St*.

*HISTORY*
The history of any manual without a *STANDARDS* section should be described in this section.

*AUTHORS*
Credits to authors, if applicable, should appear in this section. Authors should generally be noted by both name and an e-mail address.

See *An*.

*CAVEATS*
Explanations of common misuses and misunderstandings should be explained in this section.

*BUGS*
Extant bugs should be described in this section.

*SECURITY CONSIDERATIONS*
Documents any security precautions that operators should consider.

## MACRO SYNTAX

Macros are one to three three characters in length and begin with a control character , '.', at the beginning of the line. An arbitrary amount of whitespace may sit between the control character and the macro name. Thus, the following are equivalent:

```
.Pp
.   Pp
```

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 be called subsequent to the initial line-macro. If a macro is not callable, then its invocation after the initial line macro is interpreted as opaque text, such that '.Fl Sh' produces '**-Sh**'.

The *Parsable* column indicates whether the macro may be followed by further (ostensibly callable) macros. If a macro is not parsable, subsequent macro invocations on the line will be 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* contains a head.

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

| *Macro* | *Callable* | *Parsable* | *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* **-column**) has multiple heads.

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

| *Macro* | *Callable* | *Parsable* | *Scope* |
|---|---|---|---|
| *It* | No | Yes | closed by *It*, *El* |
| *Nd* | No | No | closed by *Sh* |
| *Sh* | No | No | closed by *Sh* |
| *Ss* | No | No | closed by *Sh*, *Ss* |

**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* | *Parsable* | *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 *Reserved Characters* or end of line.

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

| *Macro* | *Callable* | *Parsable* |
|---|---|---|
| *Aq* | Yes | Yes |
| *Bq* | Yes | Yes |
| *Brq* | Yes | Yes |
| *D1* | No | Yes |
| *Dl* | No | Yes |
| *Dq* | 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*.

### In-line

Closed by *Reserved Characters*, end of 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* | *Parsable* | *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 | n |
| *An* | Yes | Yes | n |
| *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 | n |
| *Db* | No | No | 1 |
| *Dd* | No | No | >0 |
| *Dt* | No | No | n |
| *Dv* | Yes | Yes | n |
| *Dx* | Yes | Yes | n |
| *Em* | Yes | Yes | >0 |
| *En* | No | No | 0 |
| *Er* | Yes | Yes | >0 |
| *Es* | No | No | 0 |
| *Ev* | Yes | Yes | n |
| *Ex* | No | No | n |
| *Fa* | Yes | Yes | n |
| *Fd* | No | No | >0 |
| *Fl* | Yes | Yes | n |
| *Fn* | Yes | Yes | >0 |
| *Fr* | No | No | n |
| *Ft* | Yes | Yes | n |
| *Fx* | Yes | Yes | n |
| *Hf* | No | No | n |
| *Ic* | Yes | Yes | >0 |
| *In* | No | No | n |
| *Lb* | No | No | 1 |
| *Li* | Yes | Yes | n |
| *Lk* | Yes | Yes | n |
| *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* | No | No | n |
| *Ox* | Yes | Yes | n |
| *Pa* | Yes | Yes | n |
| *Pf* | Yes | Yes | 1 |
| *Pp* | No | No | 0 |
| *Rv* | No | No | n |
| *Sm* | No | No | 1 |
| *St* | No | Yes | 1 |
| *Sx* | Yes | Yes | >0 |
| *Sy* | Yes | Yes | >0 |
| *Tn* | Yes | Yes | >0 |
| *Ud* | No | No | 0 |
| *Ux* | Yes | Yes | n |
| *Va* | Yes | Yes | n |
| *Vt* | Yes | Yes | >0 |
| *Xr* | Yes | Yes | >0 |
| *br* | No | No | 0 |
| *sp* | No | No | 1 |

**REFERENCE**

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

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

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

*Remarks*: this macro is not implemented in groff(1).

**%D**
Publication date of an *Rs* block. This should follow the reduced or canonical form syntax described in *Dates*.

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

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

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

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

**%P**
Book or journal page number of an *Rs* block.

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

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

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

**%U**
URI of reference document.

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

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

**Ad**

Address construct: usually in the context of an computational address in memory, not a physical (post) address.

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

**An**

Author name. This macro may alternatively accepts the following arguments, although these may not be specified along with a parameter:

**-split** Renders a line break before each author listing.

**-nosplit** The opposite of **-split**.

In the AUTHORS section, the default is not to split the first author listing, but all subsequent author listings, whether or not they're interspersed by other macros or text, are split. Thus, specifying **-split** will cause the first listing also to be split. If not in the AUTHORS section, the default is not to split.

Examples:
```
.An -nosplit
.An J. D. Ullman.
```

*Remarks*: the effects of **-split** or **-nosplit** are re-set when entering the AUTHORS section, so if one specifies *An* **-nosplit** in the general document body, it must be re-specified in the AUTHORS section.

**Ao**

Begins a block enclosed by angled brackets. Does not have any head arguments.

Examples:
```
.Fl -key= Ns Ao Ar val Ac
```

See also *Aq*.

**Ap**

Inserts an apostrophe without any surrounding white-space. This is generally used as a grammatical device when referring to the verb form of a function:

```
.Fn execve Ap d
```

**Aq**

Encloses its arguments in angled brackets.

Examples:
```
.Fl -key= Ns Aq Ar val
```

*Remarks*: this macro is often abused for rendering URIs, which should instead use *Lk* or *Mt*, or to note pre-processor "#include" statements, which should use *In*.

See also *Ao*.

**Ar**

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

Examples:
.Fl o Ns Ar file1
.Ar
.Ar arg1, arg2.

**At**
Formats an AT&T version. Accepts at most one parameter:

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

**V[.[1-4]]?** A system version of AT&T UNIX.

Note that these parameters do not begin with a hyphen.

Examples:
.At
.At V.1

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

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

**Bd**
Begins a display block. A display is collection of macros or text which may be collectively offset or justified in a manner different from that of the enclosing context. By default, the block is preceded by a vertical space.

Each display is associated with a type, which must be one of the following arguments:

**-ragged** Only left-justify the block.

**-unfilled** Do not justify the block at all.

**-filled** Left- and right-justify the block.

**-literal** Alias for **-unfilled**.

**-centered** Centre-justify each line.

The type must be provided first. Secondary arguments are as follows:

**-offset** *width* Offset by the value of *width*, which is interpreted as one of the following, specified in order:

As one of the pre-defined strings *indent*, the width of standard indentation; *indent-two*, twice *indent*; *left*, which has no effect ; *right*, which justifies to the right margin; and *center*, which aligns around an imagined centre axis.

As a precalculated width for a named macro. The most popular is the imaginary macro *Ds*, which resolves to *6n*.

As a scaling unit following the syntax described in *Scaling Widths*.

As the calculated string length of the opaque string.

If not provided an argument, it will be ignored.

**-compact** Do not assert a vertical space before the block.

**-file** *file* Prepend the file *file* before any text or macros within the block.

Examples:

```
.Bd -unfilled -offset two-indent -compact
   Hello       world.
.Ed
```

See also *D1* and *Dl*.

**Bf**

**Bk**

Begins a keep block, containing a collection of macros or text to be kept together in the output. One argument is required; additional arguments are ignored. Currently, the only argument implemented is **-words**, requesting to keep together all words of the contained text on the same output line. A **-lines** argument to keep together all lines of the contained text on the same page has been desired for a long time, but has never been implemented.

Examples:

```
.Bk -words
.Op o Ar output_file
.Ek
```

**Bl**

Begins a list composed of one or more list entries. A list is associated with a type, which is a required argument. Other arguments are **-width**, defined per-type as accepting a literal or *Scaling Widths* value; **-offset**, also accepting a literal or *Scaling Widths* value setting the list's global offset; and **-compact**, suppressing the default vertical space printed before each list entry. A list entry is specified by the *It* macro, which consists of a head and optional body (depending on the list type). A list must specify one of the following list types:

**-bullet** A list offset by a bullet. The head of list entries must be empty. List entry bodies are positioned after the bullet. The **-width** argument varies the width of list bodies' left-margins.

**-column** A columnated list. The **-width** argument has no effect. The number of columns is specified as parameters to the *Bl* macro. These dictate the width of columns either as *Scaling Widths* or literal text. If the initial macro of a **-column** list is not an *It*, an *It* context spanning each line is implied until an *It* line macro is encountered, at which point list bodies are interpreted as described in the *It* documentation.

**-dash** A list offset by a dash (hyphen). The head of list entries must be empty. List entry bodies are positioned past the dash. The **-width** argument varies the width of list bodies' left-margins.

**-diag** Like **-inset**, but with additional formatting to the head. The **-width** argument varies the width of list bodies' left-margins.

**-enum** An enumerated list offset by the enumeration from 1. The head of list entries must be empty. List entry bodies are positioned after the enumeration. The **-width** argument varies the width of list bodies' left-margins.

**-hang** Like **-tag**, but instead of list bodies positioned after the head, they trail the head text. The **-width** argument varies the width of list bodies' left-margins.

**-hyphen** Synonym for **-dash**.

**-inset** List bodies follow the list head. The **-width** argument is ignored.

**-item** This produces blocks of text. The head of list entries must be empty. The **-width** argument is ignored.

**-ohang** List bodies are positioned on the line following the head. The **-width** argument is ignored.

**-tag** A list offset by list entry heads. List entry bodies are positioned after the head as specified by the **-width** argument.

See also *It*.

**Bo**
Begins a block enclosed by square brackets. Does not have any head arguments.

Examples:

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

See also *Bq*.

**Bq**
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**
Closes a *Bro* block. Does not have any tail arguments.

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

Examples:

```
.Bro 1 , ... ,
.Va n Brc
```

See also *Brq*.

**Brq**

Encloses its arguments in curly braces.

Examples:

.Brq 1, ..., Va n

See also *Bro*.

**Bsx**

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*, *Ox*, and *Ux*.

**Bt**

Prints "is currently in beta test."

**Bx**

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

Examples:

.Bx 4.4
.Bx

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

**Cd**

Configuration declaration. This denotes strings accepted by config(8).

Examples:

.Cd device le0 at scode?

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

**Cm**

Command modifiers. Useful when specifying configuration options or keys.

Examples:

.Cm ControlPath
.Cm ControlMaster

See also *Fl*.

**D1**

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**

Start a debugging context. This macro is parsed, but generally ignored. Its syntax is as follows:

.*Db* **on | off**

**Dc**

Closes a *Do* block. Does not have any tail arguments.

**Dd**

Document date. This is the mandatory first macro of any **mdoc** manual. Its syntax is as follows:

.*Dd* **date**

The **date** field may be either *$Mdocdate$*, which signifies the current manual revision date dictated by cvs(1), or instead a valid canonical date as specified by *Dates*. If a date does not conform, the current date is used instead.

Examples:

```
.Dd $Mdocdate$
.Dd $Mdocdate: July 21 2007$
.Dd July 21, 2007
```

See also *Dt* and *Os*.

**Dl**

One-line intended 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 | less
```

See also *Bd* and *D1*.

**Do**

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

Examples:

```
.D1 Do April is the cruellest month Dc \(em T.S. Eliot
```

See also *Dq*.

**Dq**

Encloses its arguments in double quotes.

Examples:

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

See also *Do*.

**Dt**

Document title. This is the mandatory second macro of any **mdoc** file. Its syntax is as follows:

*.Dt* [**title** [**section** [**volume** | **arch**]]]

Its arguments are as follows:

**title** The document's title (name), defaulting to "UNKNOWN" if unspecified. It should be capitalised.

**section** The manual section. This may be one of *1* (utilities), *2* (system calls), *3* (libraries), *3p* (Perl libraries), *4* (devices), *5* (file formats), *6* (games), *7* (miscellaneous), *8* (system utilities), *9* (kernel functions), *X11* (X Window System), *X11R6* (X Window System), *unass* (unassociated), *local* (local system), *draft* (draft manual), or *paper* (paper). It should correspond to the manual's filename suffix and defaults to "1" if unspecified.

**volume** This overrides the volume inferred from *section*. This field is optional, and if specified, must be one of *USD* (users' supplementary documents), *PS1* (programmers' supplementary documents), *AMD* (administrators' supplementary documents), *SMM* (system managers' manuals), *URM* (users' reference manuals), *PRM* (programmers' reference manuals), *KM* (kernel manuals), *IND* (master index), *MMI* (master index), *LOCAL* (local manuals), *LOC* (local manuals), or *CON* (contributed manuals).

**arch** This specifies a specific relevant architecture. If **volume** is not provided, it may be used in its place, else it may be used subsequent that. It, too, is optional. It must be one of *alpha*, *amd64*, *amiga*, *arc*, *arm*, *armish*, *aviion*, *hp300*, *hppa*, *hppa64*, *i386*, *landisk*, *loongson*, *luna88k*, *mac68k*, *macppc*, *mvme68k*, *mvme88k*, *mvmeppc*, *pmax*, *sgi*, *socppc*, *sparc*, *sparc64*, *sun3*, *vax*, or *zaurus*.

Examples:

```
.Dt FOO 1
.Dt FOO 4 KM
.Dt FOO 9 i386
```

See also *Dd* and *Os*.

**Dv**

Defined variables such as preprocessor constants.

Examples:

```
.Dv BUFSIZ
.Dv STDOUT_FILENO
```

See also *Er*.

**Dx**

Format the DragonFly BSD 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*, *Ox*, and *Ux*.

**Ec**

**Ed**

Ef

Ek

Ends a keep context started by *Bk*.

**El**

Ends a list context started by *Bl*.

See also *Bl* and *It*.

**Em**

Denotes text that should be emphasised. Note that this is a presentation term and should not be used for stylistically decorating technical terms.

Examples:

```
.Em Warnings!
.Em Remarks:
```

**En**

**Eo**

**Er**

Display error constants.

Examples:

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

See also *Dv*.

**Es**

**Ev**

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

Examples:

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

**Ex**

Inserts text regarding a utility's exit values. This macro must have first the **-std** argument specified, then an optional *utility*. If *utility* is not provided, the document's name as stipulated in *Nm* is provided.

**Fa**

Function argument. Its syntax is as follows:

.*Fa* [**argtype**] **argname**

This may be invoked for names with or without the corresponding type. It is also used to specify the field name of a structure. Most often, the *Fa* macro is used in the *SYNOPSIS* within *Fo* section 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 foo

See also *Fo*.

**Fc**

**Fd**

Historically used to document include files. This usage has been deprecated in favour of *In*. Do not use this macro.

See also *MANUAL STRUCTURE* and *In*.

**Fl**

Command-line flag. Used when listing arguments to command-line utilities. Prints a fixed-width hyphen '-' directly followed by each argument. If no arguments are provided, a hyphen is printed followed by a space. If the argument is a macro, a hyphen is prefixed to the subsequent macro output.

Examples:

.Fl a b c
.Fl Pf a b
.Fl
.Op Fl o Ns Ar file

See also *Cm*.

**Fn**

A function name. Its syntax is as follows:

.*Fn* [**functype**] **funcname** [[**argtype**] **argname**]

Function arguments are surrounded in parenthesis and are delimited by commas. If no arguments are specified, blank parenthesis are output.

Examples:

.Fn int funcname int arg0 int arg1
.Fn funcname int arg0
.Fn funcname arg0
.Ft functype
.Fn funcname

See also *MANUAL STRUCTURE* and *Ft*.

**Fo**

Begin a function block. This is a multi-line version of *Fn*. Its syntax is as follows:

.*Fo* **funcname**

Invocations usually occur in the following context:

.*Ft* **functype**
.*Fo* **funcname**

.*Fa* [**argtype**] **argname**
...
.*Fc*

A *Fo* scope is closed by

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

**Ft**
A function type. Its syntax is as follows:

.*Ft* **functype**

Examples:
.Ft int
.Ft functype
.Fn funcname

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

**Fx**
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*, *Ox*, and *Ux*.

**Hf**

**Ic**

**In**
An "include" file. In the *SYNOPSIS* section (only if invoked as the line macro), the first argument is preceded by "#include", the arguments is enclosed in angled braces.

Examples:
.In sys/types

See also *MANUAL STRUCTURE*.

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

The **args** are phrases, a mix of macros and text corresponding to a line column, delimited by tabs or the special 'Ta' pseudo-macro. Lines subsequent the *It* are interpreted within the scope of the last phrase. Calling the pseudo-macro 'Ta' will open a new phrase scope (this must occur on a macro line to be interpreted as a macro). Note that the tab phrase delimiter may only be used within the *It* line itself. Subsequent this, only the 'Ta' pseudo-macro may be used to delimit phrases. Furthermore, note that quoted sections propagate over tab-delimited phrases on an *It*, for example,

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

will preserve the semicolon whitespace except for the last.

See also *Bl.*

**Lb**

Specify a library. The syntax is as follows:

.*Lb* **library**

The **library** parameter may be a system library, such as **libz** or **libpam**, 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 mdoc

**Li**

**Lk**

Format a hyperlink. Its syntax is as follows:

.*Lk* **uri** [**name**]

Examples:
.Lk http://bsd.lv The BSD.lv Project
.Lk http://bsd.lv

See also *Mt.*

**Lp**

**Ms**

Mt
Format a "mailto:" hyperlink. Its syntax is as follows:

.*Mt* **address**

Examples:
.Mt discuss@manpages.bsd.lv

**Nd**

**Nm**

**No**

**Ns**

**Nx**
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*, *Ox*, and *Ux*.

**Oc**

**Oo**

**Op**

**Os**
Document operating system version. This is the mandatory third macro of any **mdoc** file. Its syntax is as follows:

.*Os* [**system**]

The optional **system** parameter specifies the relevant operating system or environment. Left unspecified, it defaults to the local operating system version. This is the suggested form.

Examples:
.Os
.Os KTH/CSC/TCS
.Os BSD 4.3

See also *Dd* and *Dt*.

**Ot**
Unknown usage.

*Remarks*: this macro has been deprecated.

**Ox**
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*, *Nx*, and *Ux*.

**Pa**

**Pc**

**Pf**

**Po**

**Pp**

**Pq**

**Qc**

**Ql**

**Qo**

**Qq**

**Re**

Closes a *Rs* block. Does not have any tail arguments.

**Rs**

Begins 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, Massachusettes
.%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**

**Sc**

**Sh**

**Sm**

So

Sq

Ss

St

Sx

Sy

Tn

Ud
Prints out “currently under development.”

**Ux**
Format the UNIX name. Accepts no argument.

Examples:
.Ux

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

**Va**

**Vt**
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 in the *SYNOPSIS* section, else it accepts ordinary *In-line* syntax.

Note that this should not be confused with *Ft*, which is used for function return types.

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

See also *MANUAL STRUCTURE* and *Va*.

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

**Xo**
Open an extension scope. This macro originally existed to extend the 9-argument limit of troff; since this limit has been lifted, the macro has been deprecated.

**Xr**
Link to another manual ("cross-reference"). Its syntax is as follows:

.*Xr* **name section**

The **name** and **section** are the name and section of the linked manual. If **section** is followed by non-punctuation, an *Ns* is inserted into the token stream. This behaviour is for compatibility with groff(1).

Examples:

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

**br**

**sp**

## COMPATIBILITY

This section documents compatibility between mandoc and other other troff implementations, at this time limited to GNU troff ("groff"). The term "historic groff" refers to groff versions before the *doc.tmac* file re-write (somewhere between 1.15 and 1.19).

Heirloom troff, the other significant troff implementation accepting -mdoc, is similar to historic groff.

- Old groff fails to assert a newline before *Bd* **-ragged -compact**.
- groff behaves inconsistently when encountering non-*Fa* children of *Fo* regarding spacing between arguments. In mandoc, this is not the case: each argument is consistently followed by a single space and the trailing ')' suppresses prior spacing.
- groff behaves inconsistently when encountering *Ft* and *Fn* in the *SYNOPSIS*: at times newline(s) are suppressed depending on whether a prior *Fn* has been invoked. In mandoc, this is not the case. See *Ft* and *Fn* for the normalised behaviour.
- Historic groff does not break before an *Fn* when not invoked as the line macro in the *SYNOPSIS* section.
- Historic groff formats the *In* badly: trailing arguments are trashed and *SYNOPSIS* is not specially treated.
- groff does not accept the 'Ta' pseudo-macro as a line macro. mandoc does.
- The comment syntax '\."' is no longer accepted.
- In groff, the *Pa* macro does not format its arguments when used in the FILES section under certain list types. mandoc does.
- Historic groff does not print a dash for empty *Fl* arguments. mandoc and newer groff implementations do.
- groff behaves irregularly when specifying '\f' *Text Decoration* within line-macro scopes. mandoc follows a consistent system.
- In mandoc, negative scaling units are truncated to zero; groff would move to prior lines. Furthermore, the 'f' scaling unit, while accepted, is rendered as the default unit.
- In quoted literals, groff allowed pair-wise double-quotes to produce a standalone double-quote in formatted output. This idiosyncratic behaviour is not applicable in mandoc.
- Display offsets *Bd* **-offset** *center* and **-offset** *right* are disregarded in mandoc. Furthermore, the **-file** *file* argument is not supported in mandoc. Lastly, since text is not right-justified in mandoc (or even groff), **-ragged** and **-filled** are aliases, as are **-literal** and **-unfilled**.
- Historic groff has many un-callable macros. Most of these (excluding some block-level macros) are now callable.
- The vertical bar '|' made historic groff "go orbital" but has been a proper delimiter since then.
- *It* **-nested** is assumed for all lists (it wasn't in historic groff): any list may be nested and **-enum** lists will restart the sequence only for the sub-list.
- Some manuals use *Li* incorrectly by following it with a reserved character and expecting the delimiter to render. This is not supported in mandoc.
- In groff, the *Cd*, *Er*, *Ex*, and *Rv* macros were stipulated only to occur in certain manual sections. mandoc does not have these restrictions.
- Newer groff and mandoc print "AT&T UNIX" prior to unknown arguments of *At*; older groff did nothing.

## SEE ALSO

mandoc(1), mandoc_char(7)

## AUTHORS

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