NAME

roff - roff language reference for mandoc

DESCRIPTION

The **roff** language is a general purpose text formatting language. Since traditional implementations of the mdoc(7) and man(7) manual formatting languages are based on it, many real-world manuals use small numbers of **roff** requests and escape sequences intermixed with their mdoc(7) or man(7) code. To properly format such manuals, the mandoc(1) utility supports a subset of **roff** requests and escapes. Even though this manual page lists all **roff** requests and escape sequences, it only contains partial information about requests not supported by mandoc(1) and about language features that do not matter for manual pages. For complete **roff** manuals, consult the *SEE ALSO* section.

Input lines beginning with the control character ‘.’ are parsed for requests and macros. Such lines are called "request lines" or "macro lines", respectively. Requests change the processing state and manipulate the formatting; some macros also define the document structure and produce formatted output. The single quote ("’") is accepted as an alternative control character, treated by mandoc(1) just like ‘.’

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

LANGUAGE SYNTAX

roff documents may contain only graphable 7-bit ASCII characters, the space character, and, in certain circumstances, the tab character. The backslash character ‘\’ indicates the start of an escape sequence, used for example for *Comments* and *Special Characters*. For a complete listing of escape sequences, consult the *ESCAPE SEQUENCE REFERENCE* below.

Comments

Text following an escaped double-quote ‘\"’, whether in a request, macro, or text line, is ignored to the end of the line. A request line beginning with a control character and comment escape ‘.\"’ is also ignored. Furthermore, request lines with only a control character and optional trailing whitespace are stripped from input.

Examples:

```
.\" This is a comment line.
.\" The next line is ignored:
.
.Sh EXAMPLES \" This is a comment, too.
example text \" And so is this.
```

Special Characters

Special characters are used to encode special glyphs and are rendered differently across output media. They may occur in request, macro, and text 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.

Examples:

\(em Two-letter em dash escape.
\e One-letter backslash escape.

See mandoc_char(7) for a complete list.

Font Selection

In mdoc(7) and man(7) documents, fonts are usually selected with macros. The **\f** escape sequence and the **ft** request can be used to manually change the font, but this is not recommended in mdoc(7) documents. Such manual font changes are overridden by many subsequent macros.

The following fonts are supported:

B Bold font.
BI A font that is both bold and italic.
CB Bold constant width font. Same as **B** in terminal output.
CI Italic constant width font. Same as **I** in terminal output.
CR Regular constant width font. Same as **R** in terminal output.
CW An alias for **CR**.
I Italic font.
P Return to the previous font. If a macro caused a font change since the last **\f** eascape sequence or **ft** request, this returns to the font before the last font change in the macro rather than to the font before the last manual font change.
R Roman font. This is the default font.
1 An alias for **R**.
2 An alias for **I**.
3 An alias for **B**.
4 An alias for **BI**.

Examples:

\fBbold\fR
Write in **bold**, then switch to regular font mode.
\fIitalic\fP
Write in *italic*, then return to previous font mode.

\f(BIbold italic\fP
: Write in ***bold italic***, then return to previous font mode.

Whitespace

Whitespace consists of the space character. In text lines, whitespace is preserved within a line. In request and macro lines, whitespace delimits arguments and is discarded.

Unescaped trailing spaces are stripped from text line input unless in a literal context. In general, trailing whitespace on any input line is discouraged for reasons of portability. In the rare case that a space character is needed at the end of an input line, it may be forced by '\ \&'.

Literal space characters can be produced in the output using escape sequences. In macro lines, they can also be included in arguments using quotation; see *MACRO SYNTAX* for details.

Blank text lines, which may include whitespace, are only permitted within literal contexts. If the first character of a text line is a space, that line is printed with a leading newline.

Scaling Widths

Many requests and macros support scaled widths for their arguments. The syntax for a scaled width is '[+-]?[0-9]*.[0-9]*[:unit:]', where a decimal must be preceded or followed by at least one digit.

The following scaling units are accepted:

c	centimetre
i	inch
P	pica (1/6 inch)
p	point (1/72 inch)
f	scale 'u' by 65536
v	default vertical span
m	width of rendered 'm' (em) character
n	width of rendered 'n' (en) character
u	default horizontal span for the terminal
M	mini-em (1/100 em)

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

If a scaling unit is not provided, the numerical value is interpreted under the default rules of 'v' for vertical spaces and 'u' for horizontal ones.

Examples:

```
.Bl -tag -width 2i
                two-inch tagged list indentation in mdoc(7)
.HP 2i          two-inch tagged list indentation in man(7)
.sp 2v          two vertical spaces
```

Sentence Spacing

Each sentence should terminate at the end of an input line. By doing this, a formatter 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.

If an input line happens to end with a period, exclamation or question mark that isn’t the end of a sentence, append a zero-width space (‘\&’).

Examples:

```
Do not end sentences mid-line like this.  Instead,
end a sentence like this.
A macro would end like this:
.Xr mandoc 1 .
An abbreviation at the end of an input line needs escaping, e.g.\&
like this.
```

REQUEST SYNTAX

A request or macro line consists of:

1. the control character ‘.’ or ‘’’ at the beginning of the line,
2. optionally an arbitrary amount of whitespace,
3. the name of the request or the macro, which is one word of arbitrary length, terminated by whitespace,
4. and zero or more arguments delimited by whitespace.

Thus, the following request lines are all equivalent:

```
.ig end
.ig    end
.  ig end
```

MACRO SYNTAX

Macros are provided by the mdoc(7) and man(7) languages and can be defined by the **de** request. When called, they follow the same syntax as requests, except that macro arguments may optionally be quoted by enclosing them in double quote characters (‘"’). Quoted text, even if it contains whitespace or would cause a macro invocation when unquoted, is always considered literal text. Inside quoted text, pairs of double quote characters (‘""’) resolve to single double quote characters.

To be recognised as the beginning of a quoted argument, the opening quote character must be preceded by a space character. A quoted argument extends to the next double quote character that is not part of a pair, or to the end of the input line, whichever comes earlier. Leaving out the terminating double quote character at the end of the line is discouraged. For clarity, if more arguments follow on the same input line, it is recommended to follow the terminating double quote character by a space character; in case the next character after the terminating double quote character is anything else, it is regarded as the beginning of the next, unquoted argument.

Both in quoted and unquoted arguments, pairs of backslashes (‘\\’) resolve to single backslashes. In unquoted arguments, space characters can alternatively be included by preceding them with a backslash (‘\ ’), but quoting is usually better for clarity.

Examples:

.Fn strlen "const char *s"
: Group arguments "const char *s" into one function argument. If unspecified, "const", "char", and "*s" would be considered separate arguments.

.Op "Fl a"
: Consider "Fl a" as literal text instead of a flag macro.

REQUEST REFERENCE

The mandoc(1) **roff** parser recognises the following requests. For requests marked as "ignored" or "unsupported", any arguments are ignored, and the number of arguments is not checked.

ab [*message*]
: Abort processing. Currently unsupported.

ad [**b** | **c** | **l** | **n** | **r**]
: Set line adjustment mode for subsequent text. Currently ignored.

af *registername format*
: Assign an output format to a number register. Currently ignored.

aln *newname oldname*
: Create an alias for a number register. Currently unsupported.

als *newname oldname*

Create an alias for a request, string, macro, or diversion.

am *macroname* [*endmacro*]

Append to a macro definition. The syntax of this request is the same as that of **de**.

am1 *macroname* [*endmacro*]

Append to a macro definition, switching roff compatibility mode off during macro execution (groff extension). The syntax of this request is the same as that of **de1**. Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **am**.

ami *macrostring* [*endstring*]

Append to a macro definition, specifying the macro name indirectly (groff extension). The syntax of this request is the same as that of **dei**.

ami1 *macrostring* [*endstring*]

Append to a macro definition, specifying the macro name indirectly and switching roff compatibility mode off during macro execution (groff extension). The syntax of this request is the same as that of **dei1**. Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **ami**.

as *stringname* [*string*]

Append to a user-defined string. The syntax of this request is the same as that of **ds**. If a user-defined string with the specified name does not yet exist, it is set to the empty string before appending.

as1 *stringname* [*string*]

Append to a user-defined string, switching roff compatibility mode off during macro execution (groff extension). The syntax of this request is the same as that of **ds1**. Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **as**.

asciify *divname*

Fully unformat a diversion. Currently unsupported.

backtrace

Print a backtrace of the input stack. This is a groff extension and currently ignored.

bd *font* [*curfont*] [*offset*]

Artificially embolden by repeated printing with small shifts. Currently ignored.

bleedat *left top width height*
Set the BleedBox page parameter for PDF generation. This is a Heirloom extension and currently ignored.

blm *macroname*
Set a blank line trap. Currently unsupported.

box *divname*
Begin a diversion without including a partially filled line. Currently unsupported.

boxa *divname*
Add to a diversion without including a partially filled line. Currently unsupported.

bp [+|-]*pagenumber*
Begin a new page. Currently ignored.

BP *source height width position offset flags label*
Define a frame and place a picture in it. This is a Heirloom extension and currently unsupported.

br Break the output line.

break Break out of the innermost **while** loop.

breakchar *char* ...
Optional line break characters. This is a Heirloom extension and currently ignored.

brnl *N* Break output line after the next *N* input lines. This is a Heirloom extension and currently ignored.

brp Break and spread output line. Currently, this is implemented as an alias for **br**.

brpnl *N*
Break and spread output line after the next *N* input lines. This is a Heirloom extension and currently ignored.

c2 [*char*]
Change the no-break control character. Currently unsupported.

cc [*char*]

Change the control character. If *char* is not specified, the control character is reset to '.'. Trailing characters are ignored.

ce [*N*] Center the next *N* input lines without filling. *N* defaults to 1. An argument of 0 or less ends centering. Currently, high level macros abort centering.

cf *filename*
Output the contents of a file. Ignored because insecure.

cflags *flags char ...*
Set character flags. This is a groff extension and currently ignored.

ch *macroname* [*dist*]
Change a trap location. Currently ignored.

char *glyph* [*string*]
Define or redefine the ASCII character or character escape sequence *glyph* to be rendered as *string*, which can be empty. Only partially supported in mandoc(1); may interact incorrectly with **tr**.

chop *stringname*
Remove the last character from a macro, string, or diversion. Currently unsupported.

class *classname char ...*
Define a character class. This is a groff extension and currently ignored.

close *streamname*
Close an open file. Ignored because insecure.

CL *color text*
Print text in color. This is a Heirloom extension and currently unsupported.

color [**1** | **0**]
Activate or deactivate colors. This is a groff extension and currently ignored.

composite *from to*
Define a name component for composite glyph names. This is a groff extension and currently unsupported.

continue

Immediately start the next iteration of a **while** loop. Currently unsupported.

cp [**1** | **0**]
Switch **roff** compatibility mode on or off. Currently ignored.

cropat *left top width height*
Set the CropBox page parameter for PDF generation. This is a Heirloom extension and currently ignored.

cs *font* [*width* [*emsize*]]
Constant character spacing mode. Currently ignored.

cu [*N*] Underline next *N* input lines including whitespace. Currently ignored.

da *divname*
Append to a diversion. Currently unsupported.

dch *macroname* [*dist*]
Change a trap location in the current diversion. This is a Heirloom extension and currently unsupported.

de *macroname* [*endmacro*]
Define a **roff** macro. Its syntax can be either

.**de** *macroname*
definition
..

or

.**de** *macroname endmacro*
definition
.*endmacro*

Both forms define or redefine the macro *macroname* to represent the *definition*, which may consist of one or more input lines, including the newline characters terminating each line, optionally containing calls to **roff** requests, **roff** macros or high-level macros like man(7) or mdoc(7) macros, whichever applies to the document in question.

Specifying a custom *endmacro* works in the same way as for **ig**; namely, the call to ‘.*endmacro*’

first ends the *definition*, and after that, it is also evaluated as a **roff** request or **roff** macro, but not as a high-level macro.

The macro can be invoked later using the syntax

.*macroname* [*argument* [*argument* ...]]

Regarding argument parsing, see *MACRO SYNTAX* above.

The line invoking the macro will be replaced in the input stream by the *definition*, replacing all occurrences of \\$*N*, where *N* is a digit, by the *N*th *argument*. For example,

```
.de ZN
\fI\^\\$1\^\fP\\$2
..
.ZN XtFree .
```

produces

```
\fI\^XtFree\^\fP.
```

in the input stream, and thus in the output: *XtFree*. Each occurrence of \\$* is replaced with all the arguments, joined together with single space characters. The variant \\$@ is similar, except that each argument is individually quoted.

Since macros and user-defined strings share a common string table, defining a macro *macroname* clobbers the user-defined string *macroname*, and the *definition* can also be printed using the '*' string interpolation syntax described below **ds**, but this is rarely useful because every macro definition contains at least one explicit newline character.

In order to prevent endless recursion, both groff and mandoc(1) limit the stack depth for expanding macros and strings to a large, but finite number, and mandoc(1) also limits the length of the expanded input line. Do not rely on the exact values of these limits.

de1 *macroname* [*endmacro*]

Define a **roff** macro that will be executed with **roff** compatibility mode switched off during macro execution. This is a groff extension. Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **de**.

defcolor *newname scheme component ...*

Define a color name. This is a groff extension and currently ignored.

dei *macrostring* [*endstring*]

Define a **roff** macro, specifying the macro name indirectly (groff extension). The syntax of this request is the same as that of **de**. The effect is the same as:

.**de** *[*macrostring*] [*[*endstring*]]

dei1 *macrostring* [*endstring*]

Define a **roff** macro that will be executed with **roff** compatibility mode switched off during macro execution, specifying the macro name indirectly (groff extension). Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **dei**.

device *string ...*

devicem *stringname*

These two requests only make sense with the groff-specific intermediate output format and are unsupported.

di *divname*

Begin a diversion. Currently unsupported.

do *command* [*argument ...*]

Execute **roff** request or macro line with compatibility mode disabled. Currently unsupported.

ds *stringname* [[']*string*]

Define a user-defined string. The *stringname* and *string* arguments are space-separated. If the *string* begins with a double-quote character, that character will not be part of the string. All remaining characters on the input line form the *string*, including whitespace and double-quote characters, even trailing ones.

The *string* can be interpolated into subsequent text by using *[*stringname*] for a *stringname* of arbitrary length, or *(NN or *N if the length of *stringname* is two or one characters, respectively. Interpolation can be prevented by escaping the leading backslash; that is, an asterisk preceded by an even number of backslashes does not trigger string interpolation.

Since user-defined strings and macros share a common string table, defining a string *stringname* clobbers the macro *stringname*, and the *stringname* used for defining a string can also be invoked as a macro, in which case the following input line will be appended to the *string*, forming a new input line passed to the **roff** parser. For example,

```
.ds badidea .S
.badidea
H SYNOPSIS
```

invokes the **SH** macro when used in a man(7) document. Such abuse is of course strongly discouraged.

ds1 *stringname* [["]*string*]
Define a user-defined string that will be expanded with **roff** compatibility mode switched off during string expansion. This is a groff extension. Since mandoc(1) does not implement **roff** compatibility mode at all, it handles this request as an alias for **ds**.

dwh *dist macroname*
Set a location trap in the current diversion. This is a Heirloom extension and currently unsupported.

dt [*dist macroname*]
Set a trap within a diversion. Currently unsupported.

ec [*char*]
Enable the escape mechanism and change the escape character. The *char* argument defaults to the backslash (‘\’).

ecr Restore the escape character. Currently unsupported.

ecs Save the escape character. Currently unsupported.

el *body* The "else" half of an if/else conditional. Pops a result off the stack of conditional evaluations pushed by **ie** and uses it as its conditional. If no stack entries are present (e.g., due to no prior **ie** calls) then false is assumed. The syntax of this request is similar to **if** except that the conditional is missing.

em *macroname*
Set a trap at the end of input. Currently unsupported.

EN End an equation block. See **EQ**.

eo Disable the escape mechanism completely.

EP End a picture started by **BP**. This is a Heirloom extension and currently unsupported.

EQ Begin an equation block. See eqn(7) for a description of the equation language.

errprint *message*
Print a string like an error message. This is a Heirloom extension and currently ignored.

ev [*envname*]
Switch to another environment. Currently unsupported.

evc [*envname*]
Copy an environment into the current environment. Currently unsupported.

ex Abort processing and exit. Currently unsupported.

fallback *curfont font ...*
Select the fallback sequence for a font. This is a Heirloom extension and currently ignored.

fam [*familyname*]
Change the font family. This is a groff extension and currently ignored.

fc [*delimchar* [*padchar*]]
Define a delimiting and a padding character for fields. Currently unsupported.

fchar *glyphname* [*string*]
Define a fallback glyph. Currently unsupported.

fcolor *colorname*
Set the fill color for \D objects. This is a groff extension and currently ignored.

fdeferlig *font string ...*
Defer ligature building. This is a Heirloom extension and currently ignored.

feature +|-*name*
Enable or disable an OpenType feature. This is a Heirloom extension and currently ignored.

fi Break the output line and switch to fill mode, which is active by default but can be ended with the **nf** request. In fill mode, input from subsequent input lines is added to the same output line until the next word no longer fits, at which point the output line is broken. This request is implied by the mdoc(7) **Sh** macro and by the man(7) **SH**, **SS**, and **EE** macros.

fkern *font minkern*

Control the use of kerning tables for a font. This is a Heirloom extension and currently ignored.

fl Flush output. Currently ignored.

flig *font string char ...*
Define ligatures. This is a Heirloom extension and currently ignored.

fp *position font* [*filename*]
Assign font position. Currently ignored.

fps *mapname ...*
Mount a font with a special character map. This is a Heirloom extension and currently ignored.

fschar *font glyphname* [*string*]
Define a font-specific fallback glyph. This is a groff extension and currently unsupported.

fspacewidth *font* [*afmunits*]
Set a font-specific width for the space character. This is a Heirloom extension and currently ignored.

fspecial *curfont* [*font ...*]
Conditionally define a special font. This is a groff extension and currently ignored.

ft [*font*]
Change the font; see *Font Selection*. The *font* argument defaults to **P**.

ftr *newname* [*oldname*]
Translate font name. This is a groff extension and currently ignored.

fzoom *font* [*permille*]
Zoom font size. Currently ignored.

gcolor [*colorname*]
Set glyph color. This is a groff extension and currently ignored.

hc [*char*]
Set the hyphenation character. Currently ignored.

hcode *char code ...*
Set hyphenation codes of characters. Currently ignored.

hidechar *font char ...*

Hide characters in a font. This is a Heirloom extension and currently ignored.

hla *language*

Set hyphenation language. This is a groff extension and currently ignored.

hlm [*number*]

Set maximum number of consecutive hyphenated lines. Currently ignored.

hpf *filename*

Load hyphenation pattern file. This is a groff extension and currently ignored.

hpfa *filename*

Load hyphenation pattern file, appending to the current patterns. This is a groff extension and currently ignored.

hpfcode *code code ...*

Define mapping values for character codes in hyphenation patterns. This is a groff extension and currently ignored.

hw *word ...*

Specify hyphenation points in words. Currently ignored.

hy [*mode*]

Set automatic hyphenation mode. Currently ignored.

hylang *language*

Set hyphenation language. This is a Heirloom extension and currently ignored.

hylen *nchar*

Minimum word length for hyphenation. This is a Heirloom extension and currently ignored.

hym [*length*]

Set hyphenation margin. This is a groff extension and currently ignored.

hypp *penalty ...*

Define hyphenation penalties. This is a Heirloom extension and currently ignored.

hys [*length*]

Set hyphenation space. This is a groff extension and currently ignored.

ie *condition body*

The "if" half of an if/else conditional. The result of the conditional is pushed into a stack used by subsequent invocations of **el**, which may be separated by any intervening input (or not exist at all). Its syntax is equivalent to **if**.

if *condition body*

Begin a conditional. This request can also be written as follows:

.**if** *condition* \{*body*
body ...\}

.**if** *condition* \{\
body ...
.\}

The *condition* is a boolean expression. Currently, mandoc(1) supports the following subset of roff conditionals:

- If '!' is prefixed to *condition*, it is logically inverted.
- If the first character of *condition* is 'n' (nroff mode) or 'o' (odd page), it evaluates to true, and the *body* starts with the next character.
- If the first character of *condition* is 'e' (even page), 't' (troff mode), or 'v' (vroff mode), it evaluates to false, and the *body* starts with the next character.
- If the first character of *condition* is 'c' (character available), it evaluates to true if the following character is an ASCII character or a valid character escape sequence, or to false otherwise. The *body* starts with the character following that next character.
- If the first character of *condition* is 'd', it evaluates to true if the rest of *condition* is the name of an existing user defined macro or string; otherwise, it evaluates to false.
- If the first character of *condition* is 'r', it evaluates to true if the rest of *condition* is the name of an existing number register; otherwise, it evaluates to false.
- If the *condition* starts with a parenthesis or with an optionally signed integer number, it is evaluated according to the rules of *Numerical expressions* explained below. It evaluates to true if the result is positive, or to false if the result is zero or negative.

- Otherwise, the first character of *condition* is regarded as a delimiter and it evaluates to true if the string extending from its first to its second occurrence is equal to the string extending from its second to its third occurrence.

- If *condition* cannot be parsed, it evaluates to false.

If a conditional is false, its children are not processed, but are syntactically interpreted to preserve the integrity of the input document. Thus,

```
.if t .ig
```

will discard the '.ig', which may lead to interesting results, but

```
.if t .if t \{\
```

will continue to syntactically interpret to the block close of the final conditional. Sub-conditionals, in this case, obviously inherit the truth value of the parent.

If the *body* section is begun by an escaped brace '\{', scope continues until the end of the input line containing the matching closing-brace escape sequence '\}'. If the *body* is not enclosed in braces, scope continues until the end of the line. If the *condition* is followed by a *body* on the same line, whether after a brace or not, then requests and macros *must* begin with a control character. It is generally more intuitive, in this case, to write

```
.if condition \{\
.request
.\}
```

than having the request or macro follow as

```
.if condition \{.request
```

The scope of a conditional is always parsed, but only executed if the conditional evaluates to true.

Note that the '\}' is converted into a zero-width escape sequence if not passed as a standalone macro '.\}'. For example,

```
.Fl a \} b
```

will result in ‘\}’ being considered an argument of the ‘Fl’ macro.

ig [*endmacro*]

Ignore input. Its syntax can be either

.**ig**
ignored text
..

or

.**ig** *endmacro*
ignored text
.*endmacro*

In the first case, input is ignored until a ‘..’ request is encountered on its own line. In the second case, input is ignored until the specified ‘.*endmacro*’ is encountered. Do not use the escape character ‘\’ anywhere in the definition of *endmacro*; it would cause very strange behaviour.

When the *endmacro* is a roff request or a roff macro, like in

.ig if

the subsequent invocation of **if** will first terminate the *ignored text*, then be invoked as usual. Otherwise, it only terminates the *ignored text*, and arguments following it or the ‘..’ request are discarded.

in [[+|-]*width*]

Change indentation. See man(7). Ignored in mdoc(7).

index *register stringname substring*

Find a substring in a string. This is a Heirloom extension and currently unsupported.

it *expression macro*

Set an input line trap. The named *macro* will be invoked after processing the number of input text lines specified by the numerical *expression*. While evaluating the *expression*, the unit suffixes described below *Scaling Widths* are ignored.

itc *expression macro*

Set an input line trap, not counting lines ending with \c. Currently unsupported.

IX *class keystring*
To support the generation of a table of contents, pod2man(1) emits this user-defined macro, usually without defining it. To avoid reporting large numbers of spurious errors, mandoc(1) ignores it.

kern [**1** | **0**]
Switch kerning on or off. Currently ignored.

kernafter *font char ... afmunits ...*
Increase kerning after some characters. This is a Heirloom extension and currently ignored.

kernbefore *font char ... afmunits ...*
Increase kerning before some characters. This is a Heirloom extension and currently ignored.

kernpair *font char ... font char ... afmunits*
Add a kerning pair to the kerning table. This is a Heirloom extension and currently ignored.

lc [*glyph*]
Define a leader repetition character. Currently unsupported.

lc_ctype *localename*
Set the LC_CTYPE locale. This is a Heirloom extension and currently unsupported.

lds *macroname string*
Define a local string. This is a Heirloom extension and currently unsupported.

length *register string*
Count the number of input characters in a string. Currently unsupported.

letadj *lspmin lshmin letss lspmax lshmax*
Dynamic letter spacing and reshaping. This is a Heirloom extension and currently ignored.

lf *lineno* [*filename*]
Change the line number for error messages. Ignored because insecure.

lg [**1** | **0**]
Switch the ligature mechanism on or off. Currently ignored.

lhang *font char ... afmunits*

Hang characters at left margin. This is a Heirloom extension and currently ignored.

linetabs [**1** | **0**]

Enable or disable line-tabs mode. This is a groff extension and currently unsupported.

ll [[+|-]*width*]

Change the output line length. If the *width* argument is omitted, the line length is reset to its previous value. The default setting for terminal output is 78n. If a sign is given, the line length is added to or subtracted from; otherwise, it is set to the provided value. Using this request in new manuals is discouraged for several reasons, among others because it overrides the mandoc(1) **-O width** command line option.

lnr *register* [+|-]*value* [*increment*]

Set local number register. This is a Heirloom extension and currently unsupported.

lnrf *register* [+|-]*value* [*increment*]

Set local floating-point register. This is a Heirloom extension and currently unsupported.

lpfx *string*

Set a line prefix. This is a Heirloom extension and currently unsupported.

ls [*factor*]

Set line spacing. It takes one integer argument specifying the vertical distance of subsequent output text lines measured in v units. Currently ignored.

lsm *macroname*

Set a leading spaces trap. This is a groff extension and currently unsupported.

lt [[+|-]*width*]

Set title line length. Currently ignored.

mc *glyph* [*dist*]

Print margin character in the right margin. The *dist* is currently ignored; instead, 1n is used.

mediasize *media*

Set the device media size. This is a Heirloom extension and currently ignored.

minss *width*

Set minimum word space. This is a Heirloom extension and currently ignored.

mk [*register*]
Mark vertical position. Currently ignored.

mso *filename*
Load a macro file using the search path. Ignored because insecure.

na Disable adjusting without changing the adjustment mode. Currently ignored.

ne [*height*]
Declare the need for the specified minimum vertical space before the next trap or the bottom of the page. Currently ignored.

nf Break the output line and switch to no-fill mode. Subsequent input lines are kept together on the same output line even when exceeding the right margin, and line breaks in subsequent input cause output line breaks. This request is implied by the mdoc(7) **Bd -unfilled** and **Bd -literal** macros and by the man(7) **EX** macro. The **fi** request switches back to the default fill mode.

nh Turn off automatic hyphenation mode. Currently ignored.

nhychar *char ...*
Define hyphenation-inhibiting characters. This is a Heirloom extension and currently ignored.

nm [*start* [*inc* [*space* [*indent*]]]]
Print line numbers. Currently unsupported.

nn [*number*]
Temporarily turn off line numbering. Currently unsupported.

nop *body*
Execute the rest of the input line as a request, macro, or text line, skipping the **nop** request and any space characters immediately following it. This is mostly used to indent text lines inside macro definitions.

nr *register* [+|-]*expression* [*stepsize*]
Define or change a register. A register is an arbitrary string value that defines some sort of state, which influences parsing and/or formatting. For the syntax of *expression*, see *Numerical expressions* below. If it is prefixed by a sign, the register will be incremented or decremented instead of assigned to.

The *stepsize* is used by the **\n+** auto-increment feature. It remains unchanged when omitted

while changing an existing register, and it defaults to 0 when defining a new register.

The following *register* is handled specially:

nS If set to a positive integer value, certain mdoc(7) macros will behave in the same way as in the *SYNOPSIS* section. If set to 0, these macros will behave in the same way as outside the *SYNOPSIS* section, even when called within the *SYNOPSIS* section itself. Note that starting a new mdoc(7) section with the **Sh** macro will reset this register.

nrf *register* [+|-]*expression* [*increment*]
Define or change a floating-point register. This is a Heirloom extension and currently unsupported.

nroff Force nroff mode. This is a groff extension and currently ignored.

ns Turn on no-space mode. Currently ignored.

nx [*filename*]
Abort processing of the current input file and process another one. Ignored because insecure.

open *stream file*
Open a file for writing. Ignored because insecure.

opena *stream file*
Open a file for appending. Ignored because insecure.

os Output saved vertical space. Currently ignored.

output *string*
Output directly to intermediate output. Not supported.

padj [**1** | **0**]
Globally control paragraph-at-once adjustment. This is a Heirloom extension and currently ignored.

papersize *media*
Set the paper size. This is a Heirloom extension and currently ignored.

pc [*char*]
Change the page number character. Currently ignored.

pev Print environments. This is a groff extension and currently ignored.

pi *command*
Pipe output to a shell command. Ignored because insecure.

PI Low-level request used by **BP**. This is a Heirloom extension and currently unsupported.

pl [[+|-]*height*]
Change page length. Currently ignored.

pm Print names and sizes of macros, strings, and diversions to standard error output. Currently ignored.

pn [+|-]*number*
Change the page number of the next page. Currently ignored.

pnr Print all number registers on standard error output. Currently ignored.

po [[+|-]*offset*]
Set a horizontal page offset. If no argument is specified, the page offset is reverted to its previous value. If a sign is specified, the new page offset is calculated relative to the current one; otherwise, it is absolute. The argument follows the syntax of *Scaling Widths* and the default scaling unit is **m**.

ps [[+|-]size]
Change point size. Currently ignored.

psbb *filename*
Retrieve the bounding box of a PostScript file. Currently unsupported.

pshape *indent length ...*
Set a special shape for the current paragraph. This is a Heirloom extension and currently unsupported.

pso *command*
Include output of a shell command. Ignored because insecure.

ptr Print the names and positions of all traps on standard error output. This is a groff extension and currently ignored.

pvs [[+|-]*height*]
Change post-vertical spacing. This is a groff extension and currently ignored.

rchar *glyph ...*
Remove glyph definitions. Currently unsupported.

rd [*prompt* [*argument ...*]]
Read from standard input. Currently ignored.

recursionlimit *maxrec maxtail*
Set the maximum stack depth for recursive macros. This is a Heirloom extension and currently ignored.

return [*twice*]
Exit the presently executed macro and return to the caller. The argument is currently ignored.

rfschar *font glyph ...*
Remove font-specific fallback glyph definitions. Currently unsupported.

rhang *font char ... afmunits*
Hang characters at right margin. This is a Heirloom extension and currently ignored.

rj [*N*] Justify the next *N* input lines to the right margin without filling. *N* defaults to 1. An argument of 0 or less ends right adjustment.

rm *macroname*
Remove a request, macro or string.

rn *oldname newname*
Rename a request, macro, diversion, or string. In mandoc(1), user-defined macros, mdoc(7) and man(7) macros, and user-defined strings can be renamed, but renaming of predefined strings and of **roff** requests is not supported, and diversions are not implemented at all.

rnn *oldname newname*
Rename a number register. Currently unsupported.

rr *register*
Remove a register.

rs End no-space mode. Currently ignored.

rt [*dist*] Return to marked vertical position. Currently ignored.

schar *glyph* [*string*]
Define global fallback glyph. This is a groff extension and currently unsupported.

sentchar *char* ...
Define sentence-ending characters. This is a Heirloom extension and currently ignored.

shc [*glyph*]
Change the soft hyphen character. Currently ignored.

shift [*number*]
Shift macro arguments *number* times, by default once: \\$i becomes what \\$i+number was. Also decrement \n(.$ by *number*.

sizes *size* ...
Define permissible point sizes. This is a groff extension and currently ignored.

so *filename*
Include a source file. The file is read and its contents processed as input in place of the **so** request line. To avoid inadvertent inclusion of unrelated files, mandoc(1) only accepts relative paths not containing the strings "../" and "/..".

This request requires man(1) to change to the right directory before calling mandoc(1), per convention to the root of the manual tree. Typical usage looks like:

```
.so man3/Xcursor.3
```

As the whole concept is rather fragile, the use of **so** is discouraged. Use ln(1) instead.

sp [*height*]
Break the output line and emit vertical space. The argument follows the syntax of *Scaling Widths* and defaults to one blank line (1v).

spacewidth [**1** | **0**]
Set the space width from the font metrics file. This is a Heirloom extension and currently ignored.

special [*font* ...]
Define a special font. This is a groff extension and currently ignored.

spreadwarn [*width*]
Warn about wide spacing between words. Currently ignored.

ss *wordspace* [*sentencespace*]
Set space character size. Currently ignored.

sty *position style*
Associate style with a font position. This is a groff extension and currently ignored.

substring *stringname startpos* [*endpos*]
Replace a user-defined string with a substring. Currently unsupported.

sv [*height*]
Save vertical space. Currently ignored.

sy *command*
Execute shell command. Ignored because insecure.

T& Re-start a table layout, retaining the options of the prior table invocation. See **TS**.

ta [*width* ... [**T** *width* ...]]
Set tab stops. Each *width* argument follows the syntax of *Scaling Widths*. If prefixed by a plus sign, it is relative to the previous tab stop. The arguments after the **T** marker are used repeatedly as often as needed; for each reuse, they are taken relative to the last previously established tab stop. When **ta** is called without arguments, all tab stops are cleared.

tc [*glyph*]
Change tab repetition character. Currently unsupported.

TE End a table context. See **TS**.

ti [+|-]*width*
Break the output line and indent the next output line by *width*. If a sign is specified, the temporary indentation is calculated relative to the current indentation; otherwise, it is absolute. The argument follows the syntax of *Scaling Widths* and the default scaling unit is **m**.

tkf *font minps width1 maxps width2*
Enable track kerning for a font. Currently ignored.

tl ’*left*’*center*’*right*’

Print a title line. Currently unsupported.

tm *string*
Print to standard error output. Currently ignored.

tm1 *string*
Print to standard error output, allowing leading blanks. This is a groff extension and currently ignored.

tmc *string*
Print to standard error output without a trailing newline. This is a groff extension and currently ignored.

tr *glyph glyph ...*
Output character translation. The first glyph in each pair is replaced by the second one. Character escapes can be used; for example,

```
tr \(xx\(yy
```

replaces all invocations of \(xx with \(yy.

track *font minps width1 maxps width2*
Static letter space tracking. This is a Heirloom extension and currently ignored.

transchar *char ...*
Define transparent characters for sentence-ending. This is a Heirloom extension and currently ignored.

trf *filename*
Output the contents of a file, disallowing invalid characters. This is a groff extension and ignored because insecure.

trimat *left top width height*
Set the TrimBox page parameter for PDF generation. This is a Heirloom extension and currently ignored.

trin *glyph glyph ...*
Output character translation, ignored by **asciify**. Currently unsupported.

trnt *glyph glyph ...*

Output character translation, ignored by \!. Currently unsupported.

troff Force troff mode. This is a groff extension and currently ignored.

TS Begin a table, which formats input in aligned rows and columns. See tbl(7) for a description of the tbl language.

uf *font* Globally set the underline font. Currently ignored.

ul [*N*] Underline next *N* input lines. Currently ignored.

unformat *divname*
Unformat spaces and tabs in a diversion. Currently unsupported.

unwatch *macroname*
Disable notification for string or macro. This is a Heirloom extension and currently ignored.

unwatchn *register*
Disable notification for register. This is a Heirloom extension and currently ignored.

vpt [**1** | **0**]
Enable or disable vertical position traps. This is a groff extension and currently ignored.

vs [[+|-]*height*]
Change vertical spacing. Currently ignored.

warn *flags*
Set warning level. Currently ignored.

warnscale *si*
Set the scaling indicator used in warnings. This is a groff extension and currently ignored.

watch *macroname*
Notify on change of string or macro. This is a Heirloom extension and currently ignored.

watchlength *maxlength*
On change, report the contents of macros and strings up to the specified length. This is a Heirloom extension and currently ignored.

watchn *register*

Notify on change of register. This is a Heirloom extension and currently ignored.

wh *dist* [*macroname*]
Set a page location trap. Currently unsupported.

while *condition body*
Repeated execution while a *condition* is true, with syntax similar to **if**. Currently implemented with two restrictions: cannot nest, and each loop must start and end in the same scope.

write ["]*string*
Write to an open file. Ignored because insecure.

writec ["]*string*
Write to an open file without appending a newline. Ignored because insecure.

writem *macroname*
Write macro or string to an open file. Ignored because insecure.

xflag *level*
Set the extension level. This is a Heirloom extension and currently ignored.

Numerical expressions

The **nr**, **if**, and **ie** requests accept integer numerical expressions as arguments. These are always evaluated using the C *int* type; integer overflow works the same way as in the C language. Numbers consist of an arbitrary number of digits ‘0’ to ‘9’ prefixed by an optional sign ‘+’ or ‘-’. Each number may be followed by one optional scaling unit described below *Scaling Widths*. The following equations hold:

1i = 6v = 6P = 10m = 10n = 72p = 1000M = 240u = 240
254c = 100i = 24000u = 24000
1f = 65536u = 65536

The following binary operators are implemented. Unless otherwise stated, they behave as in the C language:

+ addition
- subtraction
***** multiplication
/ division
% remainder of division

< less than
> greater than
==
equal to
= equal to, same effect as == (this differs from C)
<=
less than or equal to
>=
greater than or equal to
<>
not equal to (corresponds to C **!=**; this one is of limited portability, it is supported by Heirloom roff, but not by groff)
& logical and (corresponds to C **&&**)
: logical or (corresponds to C ||)
<**?**
minimum (not available in C)
>**?**
maximum (not available in C)

There is no concept of precedence; evaluation proceeds from left to right, except when subexpressions are enclosed in parentheses. Inside parentheses, whitespace is ignored.

ESCAPE SEQUENCE REFERENCE

The mandoc(1) **roff** parser recognises the following escape sequences. In mdoc(7) and man(7) documents, using escape sequences is discouraged except for those described in the *LANGUAGE SYNTAX* section above.

A backslash followed by any character not listed here simply prints that character itself.

\<**newline**>
A backslash at the end of an input line can be used to continue the logical input line on the next physical input line, joining the text on both lines together as if it were on a single input line.

\<**space**>
The escape sequence backslash-space (‘\ ’) is an unpaddable space-sized non-breaking space character; see *Whitespace* and mandoc_char(7).

\! Embed text up to and including the end of the input line into the current diversion or into intermediate output without interpreting requests, macros, and escapes. Currently unsupported.

\" The rest of the input line is treated as *Comments*.

\# Line continuation with comment. Discard the rest of the physical input line and continue the logical input line on the next physical input line, joining the text on both lines together as if it were on a single input line. This is a groff extension.

\$*arg* Macro argument expansion, see **de**.

\% Hyphenation allowed at this point of the word; ignored by mandoc(1).

\& Non-printing zero-width character, often used for various kinds of escaping; see *Whitespace*, mandoc_char(7), and the "MACRO SYNTAX" and "Delimiters" sections in mdoc(7).

\' Acute accent special character; use **\(aa** instead.

\(*cc* *Special Characters* with two-letter names, see mandoc_char(7).

\) Zero-width space transparent to end-of-sentence detection; ignored by mandoc(1).

[name***]**

Interpolate the string with the *name*. For short names, there are variants *****c and **\(***cc*.

One string is predefined on the **roff** language level: ***(.T** expands to the name of the output device, for example ascii, utf8, ps, pdf, html, or markdown.

Macro sets traditionally predefine additional strings which are not portable and differ across implementations. Those supported by mandoc(1) are listed in mandoc_char(7).

Strings can be defined, changed, and deleted with the **ds**, **as**, and **rm** requests.

\, Left italic correction (groff extension); ignored by mandoc(1).

\- Special character "mathematical minus sign"; see mandoc_char(7) for details.

\/ Right italic correction (groff extension); ignored by mandoc(1).

\: Breaking the line is allowed at this point of the word without inserting a hyphen.

\? Embed the text up to the next **\?** into the current diversion without interpreting requests, macros, and escapes. This is a groff extension and currently unsupported.

\[*name***]**
Special Characters with names of arbitrary length, see mandoc_char(7).

\^ One-twelfth em half-narrow space character, effectively zero-width in mandoc(1).

_ Underline special character; use **\(ul** instead.

\‘ Grave accent special character; use **\(ga** instead.

\{ Begin conditional input; see **if**.

\| One-sixth em narrow space character, effectively zero-width in mandoc(1).

\} End conditional input; see **if**.

\~ Paddable non-breaking space character.

\0 Digit width space character.

\A’*string***’**
Anchor definition; ignored by mandoc(1).

\a Leader character; ignored by mandoc(1).

\B’*string***’**
Interpolate ‘1’ if *string* conforms to the syntax of *Numerical expressions* explained above or ‘0’ otherwise.

\b’*string***’**
Bracket building function; ignored by mandoc(1).

\C’*name***’**
Special Characters with names of arbitrary length.

\c When encountered at the end of an input text line, the next input text line is considered to continue that line, even if there are request or macro lines in between. No whitespace is inserted.

\D’*string***’**
Draw graphics function; ignored by mandoc(1).

\d Move down by half a line; ignored by mandoc(1).

\E Escape character intended to not be interpreted in copy mode. In mandoc(1), it currently does the same as \ itself.

\e Backslash special character.

\F[*name***]**
Switch font family (groff extension); ignored by mandoc(1). For short names, there are variants **\F***c* and **\F(***cc*.

\f[*name***]**
Switch to the font *name*, see *Font Selection*. For short names, there are variants **\f***c* and **\f(***cc*. An empty name **\f[]** defaults to **\fP**.

\g[*name***]**
Interpolate the format of a number register; ignored by mandoc(1). For short names, there are variants **\g***c* and **\g(***cc*.

\H'[+|-]*number***'**
Set the height of the current font; ignored by mandoc(1).

\h'[|]*width***'**
Horizontal motion. If the vertical bar is given, the motion is relative to the current indentation. Otherwise, it is relative to the current position. The default scaling unit is **m**.

\k[*name***]**
Mark horizontal input place in register; ignored by mandoc(1). For short names, there are variants **\k***c* and **\k(***cc*.

\L'*number*[*c*]**'**
Vertical line drawing function; ignored by mandoc(1).

\l'*width*[*c*]**'**
Draw a horizontal line of *width* using the glyph *c*.

\M[*name***]**
Set fill (background) color (groff extension); ignored by mandoc(1). For short names, there are variants **\M***c* and **\M(***cc*.

\m[*name***]**

Set glyph drawing color (groff extension); ignored by mandoc(1). For short names, there are variants **\m***c* and **\m(***cc*.

\N'*number***'**

Character *number* on the current font.

\n[+|-]**[***name***]**

Interpolate the number register *name*. For short names, there are variants **\n***c* and **\n(***cc*. If the optional sign is specified, the register is first incremented or decremented by the *stepsize* that was specified in the relevant **nr** request, and the changed value is interpolated.

\O*digit*, **\O[5**arguments**]**

Suppress output. This is a groff extension and currently unsupported. With an argument of **1**, **2**, **3**, or **4**, it is ignored.

\o'*string***'**

Overstrike, writing all the characters contained in the *string* to the same output position. In terminal and HTML output modes, only the last one of the characters is visible.

\p Break the output line at the end of the current word.

\R'*name* [+|-]*number***'**

Set number register; ignored by mandoc(1).

\r Move up by one line; ignored by mandoc(1).

\S'*number***'**

Slant output; ignored by mandoc(1).

\s'[+|-]*number***'**

Change point size; ignored by mandoc(1). Alternative forms **\s**[+|-]*n*, **\s**[+|-]**'***number***'**, **\s[**[+|-]*number***]**, and **\s**[+|-]**[***number***]** are also parsed and ignored.

\t Horizontal tab; ignored by mandoc(1).

\u Move up by half a line; ignored by mandoc(1).

\V[*name***]**

Interpolate an environment variable; ignored by mandoc(1). For short names, there are variants

V*c* and **V(***cc*.

v'*number***'**
Vertical motion; ignored by mandoc(1).

w'*string***'**
Interpolate the width of the *string*. The mandoc(1) implementation assumes that after expansion of user-defined strings, the *string* only contains normal characters, no escape sequences, and that each character has a width of 24 basic units.

X'*string***'**
Output *string* as device control function; ignored in nroff mode and by mandoc(1).

x'*number***'**
Extra line space function; ignored by mandoc(1).

Y[*name***]**
Output a string as a device control function; ignored in nroff mode and by mandoc(1). For short names, there are variants **Y***c* and **Y(***cc*.

Z'*string***'**
Print *string* with zero width and height; ignored by mandoc(1).

z Output the next character without advancing the cursor position.

COMPATIBILITY

The mandoc(1) implementation of the **roff** language is incomplete. Major unimplemented features include:

- For security reasons, mandoc(1) never reads or writes external files except via **so** requests with safe relative paths.
- There is no automatic hyphenation, no adjustment to the right margin, and very limited support for centering; the output is always set flush-left.
- Support for setting tabulator and leader characters is missing, and support for manually changing indentation is limited.
- The 'u' scaling unit is the default terminal unit. In traditional troff systems, this unit changes depending on the output media.
- Width measurements are implemented in a crude way and often yield wrong results. Support for explicit movement requests and escapes is limited.
- There is no concept of output pages, no support for floats, graphics drawing, and picture inclusion;

terminal output is always continuous.
- Requests regarding color, font families, font sizes, and glyph manipulation are ignored. Font support is very limited. Kerning is not implemented, and no ligatures are produced.
- The "’" macro control character does not suppress output line breaks.
- Diversions and environments are not implemented, and support for traps is very incomplete.
- Use of macros is not supported inside tbl(7) code.

The special semantics of the **nS** number register is an idiosyncrasy of OpenBSD manuals and not supported by other mdoc(7) implementations.

SEE ALSO

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

Joseph F. Ossanna and Brian W. Kernighan, *Troff User’s Manual*, *AT&T Bell Laboratories*, Computing Science Technical Report, 54, http://www.kohala.com/start/troff/cstr54.ps, Murray Hill, New Jersey, 1976 and 1992.

Joseph F. Ossanna, Brian W. Kernighan, and Gunnar Ritter, *Heirloom Documentation Tools Nroff/Troff User’s Manual*, http://heirloom.sourceforge.net/doctools/troff.pdf, September 17, 2007.

HISTORY

The RUNOFF typesetting system, whose input forms the basis for **roff**, was written in MAD and FAP for the CTSS operating system by Jerome E. Saltzer in 1964. Doug McIlroy rewrote it in BCPL in 1969, renaming it **roff**. Dennis M. Ritchie rewrote McIlroy’s **roff** in PDP-11 assembly for Version 1 AT&T UNIX, Joseph F. Ossanna improved roff and renamed it nroff for Version 2 AT&T UNIX, then ported nroff to C as troff, which Brian W. Kernighan released with Version 7 AT&T UNIX. In 1989, James Clark re-implemented troff in C++, naming it groff.

AUTHORS

This **roff** reference was written by Kristaps Dzonsons <*kristaps@bsd.lv*> and Ingo Schwarze <*schwarze@openbsd.org*>.