

NAME

roff - roff language reference

DESCRIPTION

The **roff** language is a general-purpose text-formatting language. The purpose of this document is to consistently describe those language constructs accepted by the `mandoc(1)` utility. It is a work in progress.

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

`.xx` Macro lines change control state.

Other lines are interpreted within the current state.

LANGUAGE SYNTAX

roff 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.

MACRO SYNTAX

Macros are arbitrary in length and begin with a control character, `'.'` or `''`, 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:

```
.if
. if
```

REFERENCE

This section is a canonical reference of all macros, arranged alphabetically.

am

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

ami

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

am1

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

de

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

dei

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

ds

Define a reserved word. Its syntax is as follows:

```
.ds key val
```

The **key** and **val** strings are space-separated. The **key** values may be invoked in subsequent text by using `*(NN` for two-letter pairs, `*N` for one-letter, and `*[NNN]` for arbitrary-length values.

If **val** is begun with a double-quote mark, the mark is passed over. **val** consists of *all* text following this point, including whitespace and trailing double-quotes.

del

The syntax of this macro is the same as that of *ig*, except that a leading argument must be specified. It is ignored, as are its children.

el

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 macro is similar to *if* except that the conditional is missing.

ie

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

Begins a conditional. Right now, the conditional evaluates to true if and only if it starts with the letter **n**, indicating processing in *nroff*(1) style as opposed to *troff*(1) style. 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. This macro has the following syntax:

```
.if COND \{\
BODY...
.\}
.if COND \{ BODY
BODY... \}
.if COND \{ BODY
BODY...
.\}
.if COND \
BODY
```

COND is a conditional statement. roff allows for complicated conditionals; mandoc is much simpler. At this time, mandoc supports only ‘n’, evaluating to true; and ‘t’, ‘e’, and ‘o’, evaluating to false. All other invocations are read up to the next end of line or space and evaluate as false.

If the BODY section is begun by an escaped brace ‘\{’, scope continues until a closing-brace macro ‘.\}’. If the BODY is not enclosed in braces, scope continues until the next macro or word. If the COND is followed by a BODY on the same line, whether after a brace or not, then macros *must* begin with a control character. It is generally more intuitive, in this case, to write

```
.if COND \{\
.foo
bar
.\}
```

than having the macro follow as

```
.if COND \{ .foo
```

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

Note that text subsequent a ‘.\}’ macro is discarded. Furthermore, if an explicit closing sequence ‘.\}’ is specified in a free-form line, the entire line is accepted within the scope of the prior macro, not only the

text preceding the close, with the ‘\}’ collapsing into a zero-width space.

ig

Ignore input. Accepts the following syntax:

```
.ig
BODY...
..
.ig END
BODY...
.END
```

In the first case, input is ignored until a ‘.’ macro is encountered on its own line. In the second case, input is ignored until a ‘.END’ is encountered. Text subsequent the ‘.END’ or ‘.’ is discarded.

Do not use the escape ‘\’ anywhere in the definition of END. It causes very strange behaviour. Furthermore, if you redefine a **roff** macro, such as

```
.ig if
```

the subsequent invocation of *if* will first signify the end of comment, then be invoked as a macro. This behaviour really shouldn’t be counted upon.

rm

Remove a request, macro or string. This macro is intended to have one argument, the name of the request, macro or string to be undefined. Currently, it is ignored including its arguments, and the number of arguments is not checked.

nr

Define a register. A register is an arbitrary string value that defines some sort of state, which influences parsing and/or formatting. Its syntax is as follows:

```
.nr name value
```

The **value** may, at the moment, only be an integer. The **name** is defined up to the next whitespace. The following register **name** requests are recognised:

nS If set to a positive integer value, certain *mdoc(7)* macros will behave as if they were defined in the *SYNOPSIS* section. Otherwise, this behaviour is unset (even if called within the *SYNOPSIS* section itself). Note that invoking a new *mdoc(7)* section will unset this value.

tr

Output character translation. This macro is intended to have one argument, consisting of an even number of characters. Currently, it is ignored including its arguments, and the number of arguments is not checked.

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).

- The **nS** request to *nr* is only compatible with OpenBSD's groff.
- Historic groff did not accept white-space buffering the custom END tag for the *ig* macro.
- The *if* and family would print funny white-spaces with historic groff when depending on next-line syntax.

AUTHORS

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