

NAME

man, **man_alloc**, **man_endparse**, **man_free**, **man_meta**, **man_node**, **man_parseln**, **man_reset** - man macro compiler library

SYNOPSIS

```
#include <mandoc.h>
```

```
#include <man.h>
```

```
extern const char * const * man_macronames;
```

```
struct man *
```

```
man_alloc(struct regset *regs, void *data, int pflags, mandocmsg msgs);
```

```
int
```

```
man_endparse(struct man *man);
```

```
void
```

```
man_free(struct man *man);
```

```
const struct man_meta *
```

```
man_meta(const struct man *man);
```

```
const struct man_node *
```

```
man_node(const struct man *man);
```

```
int
```

```
man_parseln(struct man *man, int line, char *buf);
```

```
void
```

```
man_reset(struct man *man);
```

DESCRIPTION

The **man** library parses lines of man(7) input into an abstract syntax tree (AST).

In general, applications initiate a parsing sequence with **man_alloc**(), parse each line in a document with **man_parseln**(), close the parsing session with **man_endparse**(), operate over the syntax tree returned by **man_node**() and **man_meta**(), then free all allocated memory with **man_free**(). The **man_reset**() function may be used in order to reset the parser for another input sequence. See the *EXAMPLES* section for a full example.

Beyond the full set of macros defined in `man(7)`, the **man** library also accepts the following macros:

- PD Has no effect. Handled as a current-scope line macro.
- Sp A synonym for ‘`sp 0.5v`’ (part of the standard preamble for Perl documentation). Handled as a line macro.
- Vb A synonym for ‘`nf`’ (part of the standard preamble for Perl documentation). Handled as a current-scope line macro.
- Ve A synonym for ‘`fi`’, closing ‘`Vb`’ (part of the standard preamble for Perl documentation). Handled as a current-scope line macro.

Furthermore, the following escapes are accepted to allow `pod2man(1)` documents to be correctly formatted: `*(-` (dash), `*(PI` (pi), `*(L"` (left double-quote), `*(R"` (right double-quote), `*(C+` (C++), `*(C‘` (left single-quote), `*(C’` (right single-quote), `*(Aq` (apostrophe), `*^` (hat), `*`, (comma), `*~` (tilde), `*/` (forward slash), `*:` (umlaut), `*8` (beta), `*o` (degree), `*(D-` (Eth), `*(d-` (eth), `*(Th` (Thorn), and `*(th` (thorn).

REFERENCE

This section further defines the *Types*, *Functions* and *Variables* available to programmers. Following that, the *Abstract Syntax Tree* section documents the output tree.

Types

Both functions (see *Functions*) and variables (see *Variables*) may use the following types:

struct man

An opaque type defined in *man.c*. Its values are only used privately within the library.

mandocmsg

A function callback type defined in *mandoc.h*.

struct man_node

A parsed node. Defined in *man.h*. See *Abstract Syntax Tree* for details.

Functions

Function descriptions follow:

man_alloc()

Allocates a parsing structure. The *data* pointer is passed to *msgs*. The *pflags* arguments are defined in *man.h*. Returns NULL on failure. If non-NULL, the pointer must be freed with **man_free()**.

man_reset()

Reset the parser for another parse routine. After its use, **man_parseln()** behaves as if invoked for the first time.

man_free()

Free all resources of a parser. The pointer is no longer valid after invocation.

man_parseln()

Parse a nil-terminated line of input. This line should not contain the trailing newline. Returns 0 on failure, 1 on success. The input buffer *buf* is modified by this function.

man_endparse()

Signals that the parse is complete. Note that if **man_endparse()** is called subsequent to **man_node()**, the resulting tree is incomplete. Returns 0 on failure, 1 on success.

man_node()

Returns the first node of the parse. Note that if **man_parseln()** or **man_endparse()** return 0, the tree will be incomplete.

man_meta()

Returns the document's parsed meta-data. If this information has not yet been supplied or **man_parseln()** or **man_endparse()** return 0, the data will be incomplete.

Variables

The following variables are also defined:

man_macronames

An array of string-ified token names.

Abstract Syntax Tree

The **man** functions produce an abstract syntax tree (AST) describing input in a regular form. It may be reviewed at any time with **man_nodes()**; however, if called before **man_endparse()**, or after **man_endparse()** or **man_parseln()** fail, it may be incomplete.

This AST is governed by the ontological rules dictated in man(7) and derives its terminology accordingly.

The AST is composed of *struct man_node* nodes with element, root and text types as declared by the *type* field. Each node also provides its parse point (the *line*, *sec*, and *pos* fields), its position in the tree (the *parent*, *child*, *next* and *prev* fields) and some type-specific data.

The tree itself is arranged according to the following normal form, where capitalised non-terminals represent nodes.

```

ROOT      <- mnode+
mnode     <- ELEMENT | TEXT | BLOCK
BLOCK     <- HEAD BODY
HEAD      <- mnode*
BODY      <- mnode*
ELEMENT   <- ELEMENT | TEXT*
TEXT      <- [[:alpha:]]*
```

The only elements capable of nesting other elements are those with next-lint scope as documented in man(7).

EXAMPLES

The following example reads lines from stdin and parses them, operating on the finished parse tree with **parsed()**. This example does not error-check nor free memory upon failure.

```

struct regset regs;
struct man *man;
struct man_node *node;
char *buf;
size_t len;
int line;
bzero(&regs, sizeof(struct regset));
line = 1;
man = man_alloc(&regs, NULL, 0, NULL);
buf = NULL;
alloc_len = 0;
while ((len = getline(&buf, &alloc_len, stdin)) >= 0) {
    if (len && buflen[len - 1] == '\n')
        buf[len - 1] = '\0';
    if (! man_parseln(man, line, buf))
        errx(1, "man_parseln");
    line++;
}
free(buf);
if (! man_endparse(man))
    errx(1, "man_endparse");
if (NULL == (node = man_node(man)))
```

```
    errx(1, "man_node");  
    parsed(man, node);  
    man_free(man);
```

Please see *main.c* in the source archive for a rigorous reference.

SEE ALSO

mandoc(1), man(7)

AUTHORS

The **man** library was written by Kristaps Dzonsons <kristaps@bsd.lv>.