NAME

mandoc_char - mandoc special characters

DESCRIPTION

This page documents the roff(7) escape sequences accepted by mandoc(1) to represent special characters in mdoc(7) and man(7) documents.

The rendering depends on the mandoc(1) output mode; it can be inspected by calling man(1) on the **mandoc_char** manual page with different **-T** arguments. In ASCII output, the rendering of some characters may be hard to interpret for the reader. Many are rendered as descriptive strings like "<integral>", "<degree>", or "<Gamma>", which may look ugly, and many are replaced by similar ASCII characters. In particular, accented characters are usually shown without the accent. For that reason, try to avoid using any of the special characters documented here except those discussed in the *DESCRIPTION*, unless they are essential for explaining the subject matter at hand, for example when documenting complicated mathematical functions.

In particular, in English manual pages, do not use special-character escape sequences to represent national language characters in author names; instead, provide ASCII transcriptions of the names.

Dashes and Hyphens

In typography there are different types of dashes of various width: the hyphen (-), the en-dash (-), the em-dash (--), and the mathematical minus sign (-).

Hyphens are used for adjectives; to separate the two parts of a compound word; or to separate a word across two successive lines of text. The hyphen does not need to be escaped:

```
blue-eyed
lorry-driver
```

The en-dash is used to separate the two elements of a range, or can be used the same way as an em-dash. It should be written as ‘\(en’:

```
pp. 95\(en97.
Go away \(en or else!
```

The em-dash can be used to show an interruption or can be used the same way as colons, semi-colons, or parentheses. It should be written as ‘\(em’:

```
Three things \(em apples, oranges, and bananas.
This is not that \(em rather, this is that.
```

In roff(7) documents, the minus sign is normally written as ‘\-’. In manual pages, some style guides recommend to also use ‘\-’ if an ASCII 0x2d "hyphen-minus" output glyph that can be copied and pasted is desired in output modes supporting it, for example in **-T utf8** and **-T html**. But currently, no practically relevant manual page formatter requires that subtlety, so in manual pages, it is sufficient to write plain ‘-’ to represent hyphen, minus, and hyphen-minus.

If a word on a text input line contains a hyphen, a formatter may decide to insert an output line break after the hyphen if that helps filling the current output line, but the whole word would overflow the line. If it is important that the word is not broken across lines in this way, a zero-width space (‘\&’) can be inserted before or after the hyphen. While mandoc(1) never breaks the output line after hyphens adjacent to a zero-width space, after any of the other dash- or hyphen-like characters represented by escape sequences, or after hyphens inside words in macro arguments, other software may not respect these rules and may break the line even in such cases.

Some roff(7) implementations contains dictionaries allowing to break the line at syllable boundaries even inside words that contain no hyphens. Such automatic hyphenation is not supported by mandoc(1), which only breaks the line at whitespace, and inside words only after existing hyphens.

Spaces

To separate words in normal text, for indenting and alignment in literal context, and when none of the following special cases apply, just use the normal space character (‘ ’).

When filling text, output lines may be broken between words, i.e. at space characters. To prevent a line break between two particular words, use the unpaddable non-breaking space escape sequence (‘\ ’) instead of the normal space character. For example, the input string "number\ 1" will be kept together as "number 1" on the same output line.

On request and macro lines, the normal space character serves as an argument delimiter. To include whitespace into arguments, quoting is usually the best choice; see the MACRO SYNTAX section in roff(7). In some cases, using the non-breaking space escape sequence (‘\ ’) may be preferable.

To escape macro names and to protect whitespace at the end of input lines, the zero-width space (‘\&’) is often useful. For example, in mdoc(7), a normal space character can be displayed in single quotes in either of the following ways:

```
.Sq " "
.Sq \ \&
```

Quotes

On request and macro lines, the double-quote character (‘"’) is handled specially to allow quoting. One

way to prevent this special handling is by using the ‘\(dq’ escape sequence.

Note that on text lines, literal double-quote characters can be used verbatim. All other quote-like characters can be used verbatim as well, even on request and macro lines.

Accents

In output modes supporting such special output characters, for example **-T pdf**, and sometimes less consistently in **-T utf8**, some roff(7) formatters convert the following ASCII input characters to the following Unicode special output characters:

‘	U+2018	left single quotation mark
’	U+2019	right single quotation mark
~	U+02DC	small tilde
^	U+02C6	modifier letter circumflex

In prose, this automatic substitution is often desirable; but when these characters have to be displayed as plain ASCII characters, for example in source code samples, they require escaping to render as follows:

\(ga	U+0060	grave accent
\(aq	U+0027	apostrophe
\(ti	U+007E	tilde
\(ha	U+005E	circumflex accent

Periods

The period (‘.’) is handled specially at the beginning of an input line, where it introduces a roff(7) request or a macro, and when appearing alone as a macro argument in mdoc(7). In such situations, prepend a zero-width space (‘\&.’) to make it behave like normal text.

Do not use the ‘\.’ escape sequence. It does not prevent special handling of the period.

Backslashes

To include a literal backslash (‘\’) into the output, use the (‘\e’) escape sequence.

Note that doubling it (‘\\’) is not the right way to output a backslash. Because mandoc(1) does not implement full roff(7) functionality, it may work with mandoc(1), but it may have weird effects on complete roff(7) implementations.

SPECIAL CHARACTERS

Special characters are encoded as ‘\X’ (for a one-character escape), ‘\(XX’ (two-character), and ‘\[N]’ (N-character). For details, see the *Special Characters* subsection of the roff(7) manual.

Spaces, non-breaking unless stated otherwise:

Input	*Description*
‘\ ’	unpaddable space
\~	paddable space
\0	digit-width space
|	one-sixth \(em narrow space, zero width in nroff mode
\^	one-twelfth \(em half-narrow space, zero width in nroff
\&	zero-width space
\)	zero-width space transparent to end-of-sentence detection
\%	zero-width space allowing hyphenation
\:	zero-width space allowing line break

Lines:

Input	*Rendered*	*Description*
\(ba	|	bar
\(br	|	box rule
\(ul	_	underscore
\(ru	_	underscore (width 0.5m)
\(rn	-	overline
\(bb	|	broken bar
\(sl	/	forward slash
\(rs	\	backward slash

Text markers:

Input	*Rendered*	*Description*
\(ci	O	circle
\(bu	⟐	bullet
\(dd	<**>	double dagger
\(dg	<*>	dagger
\(lz	<>	lozenge
\(sq	[]	white square
\(ps	<paragraph>	paragraph
\(sc	<section>	section
\(lh	<=	left hand
\(rh	=>	right hand
\(at	@	at
\(sh	#	hash (pound)
\(CR	<cr>	carriage return
\(OK	\/	check mark

\(CL	C	club suit
\(SP	S	spade suit
\(HE	H	heart suit
\(DI	D	diamond suit

Legal symbols:

Input	*Rendered*	*Description*
\(co	(C)	copyright
\(rg	(R)	registered
\(tm	tm	trademarked

Punctuation:

Input	*Rendered*	*Description*
\(em	--	em-dash
\(en	-	en-dash
\(hy	-	hyphen
\e	\	back-slash
\.	.	period
\(r!	!	upside-down exclamation
\(r?	?	upside-down question

Quotes:

Input	*Rendered*	*Description*
\(Bq	„	right low double-quote
\(bq	‚	right low single-quote
\(lq	"	left double-quote
\(rq	"	right double-quote
\(oq	‘	left single-quote
\(cq	’	right single-quote
\(aq	’	apostrophe quote (ASCII character)
\(dq	"	double quote (ASCII character)
\(Fo	<<	left guillemet
\(Fc	>>	right guillemet
\(fo	<	left single guillemet
\(fc	>	right single guillemet

Brackets:

Input	*Rendered*	*Description*
\(lB	[	left bracket
\(rB	]	right bracket

\(lC	{	left brace
\(rC	}	right brace
\(la	<	left angle
\(ra	>	right angle
\(bv			brace extension (special font)
\[braceex]			brace extension
\[bracketlefttp]			top-left hooked bracket
\[bracketleftbt]			bottom-left hooked bracket
\[bracketleftex]			left hooked bracket extension
\[bracketrighttp]			top-right hooked bracket
\[bracketrightbt]			bottom-right hooked bracket
\[bracketrightex]			right hooked bracket extension
\(lt	,-	top-left hooked brace
\[bracelefttp]	,-	top-left hooked brace
\(lk	{	mid-left hooked brace
\[braceleftmid]	{	mid-left hooked brace
\(lb	‘-	bottom-left hooked brace
\[braceleftbt]	‘-	bottom-left hooked brace
\[braceleftex]			left hooked brace extension
\(rt	-.	top-left hooked brace
\[bracerighttp]	-.	top-right hooked brace
\(rk	}	mid-right hooked brace
\[bracerightmid]	}	mid-right hooked brace
\(rb	-’	bottom-right hooked brace
\[bracerightbt]	-’	bottom-right hooked brace
\[bracerightex]			right hooked brace extension
\[parenlefttp]	/	top-left hooked parenthesis
\[parenleftbt]	\	bottom-left hooked parenthesis
\[parenleftex]			left hooked parenthesis extension
\[parenrighttp]	\	top-right hooked parenthesis
\[parenrightbt]	/	bottom-right hooked parenthesis
\[parenrightex]			right hooked parenthesis extension

Arrows:

Input	*Rendered*	*Description*
\(<-	<-	left arrow
\(->	->	right arrow
\(<>	<->	left-right arrow
\(da	v	down arrow
\(ua	^	up arrow

\(va	^v	up-down arrow
\(lA	<=	left double-arrow
\(rA	=>	right double-arrow
\(hA	<=>	left-right double-arrow
\(uA	⇑	up double-arrow
\(dA	⇓	down double-arrow
\(vA	^=v	up-down double-arrow
\(an	-	horizontal arrow extension

Logical:

Input	*Rendered*	*Description*
\(AN	^	logical and
\(OR	v	logical or
\[tno]	~	logical not (text font)
\(no	~	logical not (special font)
\(te	<there exists>	existential quantifier
\(fa	<for all>	universal quantifier
\(st	<such that>	such that
\(tf	<therefore>	therefore
\(3d	<therefore>	therefore
\(or	|	bitwise or

Mathematical:

Input	*Rendered*	*Description*
\-	-	minus (text font)
\(mi	-	minus (special font)
+	+	plus (text font)
\(pl	+	plus (special font)
\(-+	-+	minus-plus
\[t+-]	+-	plus-minus (text font)
\(+-	+-	plus-minus (special font)
\(pc	.	center-dot
\[tmu]	x	multiply (text font)
\(mu	x	multiply (special font)
\(c*	⊗	circle-multiply
\(c+	⊕	circle-plus

\[tdi]	/	divide (text font)
\(di	/	divide (special font)
\(f/	/	fraction
\(**	*	asterisk
\(<=	<=	less-than-equal
\(>=	>=	greater-than-equal
\(<<	<<	much less
\(>>	>>	much greater
\(eq	=	equal
\(!=	!=	not equal
\(==	==	equivalent
\(ne	!==	not equivalent
\(ap	~	tilde operator
\(|=	-~	asymptotically equal
\(=~	=~	approximately equal
\(~~	~~	almost equal
\(~=	~=	almost equal
\(pt	<proportional to>	proportionate
\(es	{}	empty set
\(mo	<element of>	element
\(nm	<not element of>	not element
\(sb	<proper subset>	proper subset
\(nb	<not subset>	not subset
\(sp	<proper superset>	proper superset
\(nc	<not superset>	not superset
\(ib	<subset or equal>	reflexive subset
\(ip	<superset or equal>	reflexive superset
\(ca	<intersection>	intersection
\(cu	<union>	union
\(/_	<angle>	angle

\(pp	<perpendicular>	perpendicular
\(is	<integral>	integral
\[integral]	<integral>	integral
\[sum]	<sum>	summation
\[product]	<product>	product
\[coproduct]	<coproduct>	coproduct
\(gr	<nabla>	gradient
\(sr	<sqrt>	square root
\[sqrt]	<sqrt>	square root
\(lc	|~	left-ceiling
\(rc	~|	right-ceiling
\(lf	|_	left-floor
\(rf	_|	right-floor
\(if	<infinity>	infinity
\(Ah	<Aleph>	aleph
\(Im	<Im>	imaginary
\(Re	<Re>	real
\(wp	p	Weierstrass p
\(pd	<del>	partial differential
\(-h	/h	Planck constant over 2<pi>
\[hbar]	/h	Planck constant over 2<pi>
\(12	1/2	one-half
\(14	1/4	one-fourth
\(34	3/4	three-fourths
\(18	1/8	one-eighth
\(38	3/8	three-eighths
\(58	5/8	five-eighths
\(78	7/8	seven-eighths
\(S1	^1	superscript 1
\(S2	^2	superscript 2
\(S3	^3	superscript 3

Ligatures:

Input	*Rendered*	*Description*
\(ff	ff	ff ligature
\(fi	fi	fi ligature
\(fl	fl	fl ligature
\(Fi	ffi	ffi ligature

\(Fl	ffl	ffl ligature
\(AE	AE	AE
\(ae	ae	ae
\(OE	OE	OE
\(oe	oe	oe
\(ss	ss	German eszett
\(IJ	IJ	IJ ligature
\(ij	ij	ij ligature

Accents:

Input	*Rendered*	*Description*
\(a"	˝	Hungarian umlaut
\(a-	¯	macron
\(a.	˙	dotted
\(a^	^	circumflex
\(aa	´	acute
\’	´	acute
\(ga	`	grave
\‘	`	grave
\(ab	˘	breve
\(ac	¸	cedilla
\(ad	¨	dieresis
\(ah	ˇ	caron
\(ao	˚	ring
\(a~	~	tilde
\(ho	˛	ogonek
\(ha	^	hat (ASCII character)
\(ti	~	tilde (ASCII character)

Accented letters:

Input	*Rendered*	*Description*
\(’A	Á	acute A
\(’E	É	acute E
\(’I	Í	acute I
\(’O	Ó	acute O
\(’U	Ú	acute U
\(’Y	Ý	acute Y
\(’a	á	acute a
\(’e	é	acute e
\(’i	í	acute i

\('o	ó	acute o
\('u	ú	acute u
\('y	ý	acute y
\(`A	À	grave A
\(`E	È	grave E
\(`I	Ì	grave I
\(`O	Ò	grave O
\(`U	Ù	grave U
\(`a	à	grave a
\(`e	è	grave e
\(`i	ì	grave i
\(`o	ò	grave o
\(`u	ù	grave u
\(~A	Ã	tilde A
\(~N	Ñ	tilde N
\(~O	Õ	tilde O
\(~a	ã	tilde a
\(~n	ñ	tilde n
\(~o	õ	tilde o
\(:A	Ä	dieresis A
\(:E	Ë	dieresis E
\(:I	Ï	dieresis I
\(:O	Ö	dieresis O
\(:U	Ü	dieresis U
\(:a	ä	dieresis a
\(:e	ë	dieresis e
\(:i	ï	dieresis i
\(:o	ö	dieresis o
\(:u	ü	dieresis u
\(:y	ÿ	dieresis y
\(^A	Â	circumflex A
\(^E	Ê	circumflex E
\(^I	Î	circumflex I
\(^O	Ô	circumflex O
\(^U	Û	circumflex U
\(^a	â	circumflex a
\(^e	ê	circumflex e
\(^i	î	circumflex i
\(^o	ô	circumflex o
\(^u	û	circumflex u

\(,C	Ç	cedilla C
\(,c	ç	cedilla c
\(/L	Ł	stroke L
\(/l	ł	stroke l
\(/O	Ø	stroke O
\(/o	ø	stroke o
\(oA	Å	ring A
\(oa	å	ring a

Special letters:

Input	*Rendered*	*Description*
\(-D	Dh	Eth
\(Sd	dh	eth
\(TP	Th	Thorn
\(Tp	th	thorn
\(.i	i	dotless i
\(.j	j	dotless j

Currency:

Input	*Rendered*	*Description*
\(Do	$	dollar
\(ct	¢	cent
\(Eu	EUR	Euro symbol
\(eu	EUR	Euro symbol
\(Ye	¥	yen
\(Po	Ł	pound
\(Cs	¤	Scandinavian
\(Fn	ƒ	florin

Units:

Input	*Rendered*	*Description*
\(de	<degree>	degree
\(%0	<permille>	per-thousand
\(fm	’	minute
\(sd	’’	second
\(mc	<micro>	micro
\(Of	*a*	Spanish female ordinal
\(Om	*o*	Spanish masculine ordinal

Greek letters:

Input	*Rendered*	*Description*
\(*A	A	Alpha
\(*B	B	Beta
\(*G	<Gamma>	Gamma
\(*D	<Delta>	Delta
\(*E	E	Epsilon
\(*Z	Z	Zeta
\(*Y	H	Eta
\(*H	<Theta>	Theta
\(*I	I	Iota
\(*K	K	Kappa
\(*L	<Lambda>	Lambda
\(*M	M	Mu
\(*N	N	Nu
\(*C	<Xi>	Xi
\(*O	O	Omicron
\(*P	<Pi>	Pi
\(*R	P	Rho
\(*S	<Sigma>	Sigma
\(*T	T	Tau
\(*U	Y	Upsilon
\(*F	<Phi>	Phi
\(*X	X	Chi
\(*Q	<Psi>	Psi
\(*W	<Omega>	Omega
\(*a	<alpha>	alpha
\(*b	<beta>	beta
\(*g	<gamma>	gamma
\(*d	<delta>	delta
\(*e	<epsilon>	epsilon
\(*z	<zeta>	zeta
\(*y	<eta>	eta
\(*h	<theta>	theta
\(*i	<iota>	iota
\(*k	<kappa>	kappa
\(*l	<lambda>	lambda
\(*m	<mu>	mu
\(*n	<nu>	nu
\(*c	<xi>	xi
\(*o	o	omicron

\(*p	<pi>	pi
\(*r	<rho>	rho
\(*s	<sigma>	sigma
\(*t	<tau>	tau
\(*u	<upsilon>	upsilon
\(*f	<phi>	phi
\(*x	<chi>	chi
\(*q	<psi>	psi
\(*w	<omega>	omega
\(+h	<theta>	theta variant
\(+f	<phi>	phi variant
\(+p	<pi>	pi variant
\(+e	<epsilon>	epsilon variant
\(ts	<sigma>	sigma terminal

PREDEFINED STRINGS

Predefined strings are inherited from the macro packages of historical troff implementations. They are *not recommended* for use, as they differ across implementations. Manuals using these predefined strings are almost certainly not portable.

Their syntax is similar to special characters, using ‘*X’ (for a one-character escape), ‘*(XX’ (two-character), and ‘*[N]’ (N-character).

Input	*Rendered*	*Description*
*(Ba	|	vertical bar
*(Ne	!=	not equal
*(Ge	>=	greater-than-equal
*(Le	<=	less-than-equal
*(Gt	>	greater-than
*(Lt	<	less-than
*(Pm	+-	plus-minus
*(If	infinity	infinity
*(Pi	pi	pi
*(Na	NaN	NaN
*(Am	&	ampersand
*R	(R)	restricted mark
*(Tm	(Tm)	trade mark
*q	"	double-quote
*(Rq	"	right-double-quote
*(Lq	"	left-double-quote

*(lp	(	right-parenthesis
*(rp	)	left-parenthesis
*(lq	"	left double-quote
*(rq	"	right double-quote
*(ua	↑	up arrow
*(va	^v	up-down arrow
*(<=	<=	less-than-equal
*(>=	>=	greater-than-equal
*(aa	’	acute
*(ga	‘	grave
*(Px	POSIX	POSIX standard name
*(Ai	ANSI	ANSI standard name

UNICODE CHARACTERS

The escape sequences

\[uXXXX] and \C’uXXXX’

are interpreted as Unicode codepoints. The codepoint must be in the range above U+0080 and less than U+10FFFF. For compatibility, the hexadecimal digits ‘A’ to ‘F’ must be given as uppercase characters, and points must be zero-padded to four characters; if greater than four characters, no zero padding is allowed. Unicode surrogates are not allowed.

NUMBERED CHARACTERS

For backward compatibility with existing manuals, mandoc(1) also supports the

\N’*number*’ and \[**char***number*]

escape sequences, inserting the character *number* from the current character set into the output. Of course, this is inherently non-portable and is already marked as deprecated in the Heirloom roff manual; on top of that, the second form is a GNU extension. For example, do not use \N’34’ or \[char34], use \(dq, or even the plain ‘"’ character where possible.

COMPATIBILITY

This section documents compatibility between mandoc and other troff implementations, at this time limited to GNU troff ("groff").

- The \N’’ escape sequence is limited to printable characters; in groff, it accepts arbitrary character numbers.
- In **-Tascii**, the \(ss, \(nm, \(nb, \(nc, \(ib, \(ip, \(pp, \[sum], \[product], \[coproduct], \(gr, \(-h, and \(a.

special characters render differently between mandoc and groff.

- In **-Thtml**, the \(~=, \(nb, and \(nc special characters render differently between mandoc and groff.
- The **-Tps** and **-Tpdf** modes format like **-Tascii** instead of rendering glyphs as in groff.
- The \[radicalex], \[sqrtex], and \(ru special characters have been omitted from mandoc either because they are poorly documented or they have no known representation.

SEE ALSO

mandoc(1), man(7), mdoc(7), roff(7)

AUTHORS

The **mandoc_char** manual page was written by Kristaps Dzonsons <*kristaps@bsd.lv*>.

CAVEATS

The predefined string ‘*(Ba’ mimics the behaviour of the ‘|’ character in mdoc(7); thus, if you wish to render a vertical bar with no side effects, use the ‘\(ba’ escape.