NAME

groff_toc − a GNU roff macro framework for table of contents collation

DESCRIPTION

The **groff_toc** macro packages provide a minimal framework for managing the specification, and the collation of table of contents entries. As distributed, this framework comprises *two* component macro packages, namely **toc−base.tmac**, and **toc−class.tmac**; it is intended that this minimal framework should be extended, by the addition of user-defined macros, to control the layout of tables of contents.

USAGE

The **groff_toc** macro framework may be loaded directly from the **groff**(1) command line:

groff [*options* ...] **−mtoc−base** [*files* ...]
groff [*options* ...] **−mtoc−class** [*files* ...]

Alternatively, since user-written extension macros are a normal prerequisite for successful deployment of the **groff_toc** macro framework, it may, and often will, be found to be more convenient to load it from within a **groff**(7) document source file, or from some other dependent macro package:

.mso toc−base.tmac
.mso toc−class.tmac

All features of the **groff_toc** framework are accessed through a single primary macro, named **toc**, which should be called in accordance with the syntax specification:

.toc *opcode* [*arglist* ...]

The *opcode* argument is required; it causes the call to be redirected to a supplementary macro, named **.toc.***opcode*, which may correspond to one of the predefined actions, as specified below, or it may be a user-defined extension macro, to perform the desired operation.

The remaining arguments, designated by [*arglist* ...] are formally optional; however, depending on the particular *opcode* which has been specified, at least some of these formally optional arguments may be required.

When only the **toc−base** framework component is loaded, a minimal set of only three basic macros is defined; these predefined **groff_toc** macros correspond to **toc** macro calls of the form:

.toc file [*name*]
.toc put *opcode* [*arglist* ...]
.toc error *message text* ...

Alternatively, when the **toc−class** framework extension component is loaded, it will initially ensure that the **toc−base** component has been preloaded; the three basic macros are then augmented by provision of additional predefined macros, which correspond to supplementary **toc** macro calls of the form:

.toc activate [*class−name* ...]
.toc classified *opcode* ...
.toc classify [*class−name* ...]
.toc exclude [*class−name* ...]
.toc select [*class−name* ...]

The operation of each of those macros, which are predefined within the **toc−base** component of the **groff_toc** framework, is described below:

.toc file [*name*]

Closes any file stream, which may have been opened by a previous **.toc file** assignment, and opens *name* as a new file stream, for collection of table of contents reference data. if the *name* argument is omitted, or if the **groff**(7) input data stream is being processed by **pdfroff**(1), collected table of contents reference data will be directed to the standard error output data stream, rather than to a named file.

When the collected table of contents reference data is to be directed to a named file, it will normally be necessary to invoke **groff**(1) in *unsafe* mode, (that is, with its **−U** option in effect); however, in the special case of processing with **pdfroff**(1), this restriction may be re-

laxed, because the collected data will be directed through the standard error stream, whence **pdfroff**(1) will redirect it to the named file.

.toc put *opcode* [*arglist* ...]
Write a table of contents entry specification to the nominated reference data collection stream; this reference takes the form:

```
\*[TOC.REQUEST] opcode [arglist ...]
```

with the default value of `\*[TOC.REQUEST]` being specified as `.toc`, (as discussed in the **CONTROL VARIABLES** section, below); thus, the table of contents reference data is recorded as a sequence of macro calls, which will then be evaluated, when this data is read back into the document source input stream.

Note that, regardless of how **TOC.REQUEST** may have been defined, the onus falls on the user, to furnish any macros which may be required to evaluate each

```
\*[TOC.REQUEST] opcode [arglist ...]
```

request which has been inserted into the table of contents reference data stream, by any **.toc put** macro call; when **TOC.REQUEST** retains its default assignment, this implies that the user must furnish a **.toc.***opcode* macro definition for each and every distinct *opcode* which is specified, in any such **.toc put** macro call.

.toc error *message text* ...
Emit a diagnostic message, on the standard error output stream; this message is presented in the form:

```
toc macro error: message text ...
```

This form of the **toc** macro is typically used to report issues which are encountered when evaluating other macros, within the context of the **groff_toc** framework, (both those which are included within the basic framework, and user-defined extensions to it).

When the **toc−class** extension to the **groff_toc** framework has been loaded, the supplementary macros, which it provides, operate as follows:

.toc activate [*class−name* ...]
Mark the occurrence of change events in *class−name* state, for subsequent use during ultimate selective interpolation of recorded **.toc file** reference data. Such macro calls may be observed within the **.toc file** reference data stream; they are generated *automatically*, to reflect classification state changes as specified by **.toc classify** macro calls, and thus, there is normally no reason for it *ever* to be invoked *directly*, by the user.

.toc classified *opcode* ...
Stipulate that, when interpolating recorded **.toc file** reference data, the operation associated with each specified *opcode* argument should be performed *only* when the specified classification and selection states match, but *never* when the classification state matches any specified exclusion state.

.toc classify [*class−name* ...]
Stipulate that any table of contents entry specifications which follow, until a further "**.toc classify**" macro call is encountered, are to be considered as belonging to each of the classes specified in the list of *class−name* arguments.

If no *class−name* arguments are specified, any following table of contents entries will revert to being considered as *unclassified*.

.toc exclude [*class−name* ...]
Specify an exclusion state, in terms of a list of table of contents entry *class−name*s, and make it *active*. When any such exclusion state is active, invocation of any *opcode* handler macro which has been designated as *classified* will be suppressed, when interpolating any table of contents entry in the recorded **.toc file** data stream, and there is at least one *class−name* which is common to both the active classification state, if any, and the exclusion state.

Any active exclusion state may be cancelled, by invoking **.toc exclude** again, without specifying any *class−name* arguments; alternatively, an active exclusion state may be replaced by another, by invoking **.toc exclude** again, with a new list of *class−name* arguments.

.toc select [*class−name* ...]
Specify a selection state, in terms of a list of table of contents entry *class−name*s, and make it *active*. When any such selection state is active, only those table of contents entries with at least one *class−name* which is common to both the active classification state, if any, and the selection state, will be included in the output, when a **.toc file** data stream is interpolated by those *opcode* handler macros which have been designated as *classified*.

In the event of any selection state being active concurrently with any exclusion state, the effect of any *class−name* match between the classification state and the exclusion state will override the effect of any match between the classification state and the selection state.

Any active selection state may be cancelled, by invoking **.toc select** again, without specifying any *class−name* arguments; alternatively, an active selection state may be replaced by another, by invoking **.toc select** again, with a new list of *class−name* arguments.

On their own, these variants of the **toc** macro, as described above, will not be sufficient to lay out a table of contents; thus, it will be necessary for the user to define at least one, and possibly a collection of several supplementary macros, to fulfil the evaluation of the macro sequence, as written to, and subsequently read back from, the table of contents reference data collection stream.

CONTROL VARIABLES

The **groff_toc** macro framework defines *one* **groff**(7) string, namely **TOC.REQUEST**, which is used to specify the first token — nominally a macro name, preceded by the **groff**(7) *control character* — in each record written to the table of contents reference data stream, by the **toc put** macro. By default, this is defined as **.toc**, which implies that each record in this data stream will represent a macro call to another variant — which, of necessity, should usually be user-defined — of the **toc** macro.

After loading the *toc−base.tmac* macro file, the user may redefine the **TOC.REQUEST** string, thus substituting an alternative to the **toc** macro, for laying out the table of contents; (any such substitute macro will usually also need to be furnished by the user).

FILES

/usr/local/share/groff/site-tmac/toc−base.tmac
Implements the **toc−base** components of the **groff_toc** framework.

/usr/local/share/groff/site-tmac/toc−class.tmac
Implements the **toc−class** components of the **groff_toc** framework, having first included */usr/local/share/groff/site-tmac/toc−base.tmac*, to ensure that the **toc−base** components are also automatically incorporated.

/usr/local/share/groff/site-tmac/spdf-toc.tmac
Provides a concrete working example of integration of the **groff_toc** macro framework with **pdfroff**(1), the *spdf.tmac*, and the *pdfmark.tmac* macros.

CAVEATS AND BUGS

In general, when **.toc file** is called with a *file−name* argument, it will become necessary to run **groff**(1) in *unsafe* mode, (i.e. with the **−U** option in effect); due to the manner in which the **groff_toc** macro framework has been designed to interoperate with **pdfroff**(1), this limitation does not apply when **pdfroff**(1) is used to drive the **groff**(1) formatting process, but it must be borne in mind, if using **groff**(1) directly, or via any other process driver.

Also, when using **pdfroff**(1) as the **groff**(1) process driver, *only* the default setting of **.toc** is suitable as an assignment for **TOC.REQUEST**; **pdfroff**(1) will not recognize any value, other than **.toc**, as a valid record signature for a table of contents data reference.

A further consideration is that, when **.toc file** is called, with a *file−name* argument, any prior content, within the named file, is deleted. Since this content is precisely that which is to be interpolated, when writing any table of contents data to the document output stream, *all* such interpolations *must* precede the **.toc file** assignment, which itself *must* precede any **.toc put** macro call; this would thus appear to preclude any further table of contents data interpolation, *after* table of contents reference data collection has

commenced. Once again, this limitation does not affect the operation of **pdfroff**(1), because it uses a distinctly named intermediate file for reference data collection, while interpolating the content of the file named in the **.toc file** macro call, subsequently replacing this interpolated file by the intermediate file, only after completing one, and commencing the next, of its multiple **groff**(1) processing passes.

Finally, when using the extended capabilities offered by the **toc−class** macro package, it is recommended that classification state, selection state, and exclusion state specifications should be kept as simple as practicable; excessively complex specifications, and in particular, concurrent use of selection and exclusion state specifications, may become confusing to the user, or the document maintainer, and may lead to unexpected behaviour.

EXAMPLES

As a simple example of how the **groff_toc** framework might be deployed, consider the generation of a single table of contents, which is to be placed, conventionally, before the body text of its containing document. Typically, the first step would be to activate the **groff_toc** framework:

```
.mso toc-base.tmac
```

After activation of this framework, and assuming that the following output will be placed at the top of a new page, a suitable heading would normally be printed above the table of contents itself:

```
.\" If not at top of page, ".bp" may be needed before
.\" placing a centred heading over the table of contents,
.\" setting it in bold, with two points size increment.
.
.ce 1
.nop \fB\s'+2z'Table of Contents\s'-2z'\fP
.sp 2v
```

After printing the heading, the table of contents itself may be interpolated, by interpretation of the content of the reference data file, as collected during a previous processing pass, *before* reinitializing the data collector, in preparation of rebuilding, and updating it during the current pass:

```
.\" Initialization of the reference data collector may
.\" overwrite any existing content in the associated file,
.\" thus, this file must be read in, BEFORE the collector
.\" file stream is assigned.  Additionally, any required
.\" toc.extension macros must be defined here, BEFORE
.\" reading any existing content from the named file.
.
.so foo.toc
.toc file foo.toc
```

Following this, to add entries into the table of contents, place macro calls similar to the following, at each point throughout the document source, to which a table of contents entry is to refer:

```
.toc put pageref \n% reference text ...
```

Notice that, to service the macro calls which this will insert into the table of contents reference data file, a user-defined **toc.***extension* macro, named **toc.pageref**, must be defined, *before* the file is read in; (that is, it *must* have been defined *before*, or at least *at*, the position of the comment, within the initial example sequence of requests, above, which indicates this requirement). Assuming that appropriate tab stops have been set up already, and that no-fill mode has been activated, a suitable, albeit rudimentary, implementation for such an extension macro might look like:

```
.de toc.pageref
.\" Usage: .toc pageref page-number reference text ...
.\"
.\" Placing the page number as the first argument, before
.\" the reference text arguments, may seem counterintuitive;
.\" it is done this way to avoid any requirement to iterate
.\" over the reference text, to locate the page number.
.\"
.   nr \\$0.page \\$1            \" save page number ...
```

```
.   shift                          \" and remove from arguments
.   nop \\$* \a\t\\n[\\$0.page]    \" emit TOC entry
.   rr \\$0.page                   \" clean up local storage
..
```

The foregoing example may be sufficient, when the aggregate of the collected table of contents data is to be presented within a single output list, preceding the document body. However, if it is desired to segregate the collected data into two, or more separate categories, (a table of contents, followed by a list of figures, for example), then it may be adapted, using the **toc−class** extension to the **groff_toc** framework; this may be illustrated by the following adaptation, whereby the **toc−class** extension is loaded, in place of the **toc−base** framework, before printing the table of contents heading:

```
.mso toc-class.tmac
.
.\" Place a centred heading over the table of contents,
.\" setting it in bold, with two points size increment.
.
.ce 1
.nop \fB\s'+2z'Table of Contents\s'-2z'\fP
.sp 2v
```

The **toc.pageref** macro should then be defined, just as in the preceding example; however, it *must* then be designated for operation within classified scopes, *before* interpolation of any reference data:

```
.\" Enable handling of classified reference data,
.\" when processing ".toc pageref ..." macro calls.
.\"
.toc classified pageref
```

Having implemented these adaptations, and assuming that the table of contents reference data has been collected *without* classification, whereas each list of figures reference has been assigned a “figures” classification, the table of contents may be interpolated, thus:

```
.\" Interpolate the table of contents reference data,
.\" ignoring all "list of figures" references.
.\"
.toc exclude figures
.so foo.toc
```

The foregoing interpolation of the “unclassified” records, within the collected reference data stream file, will have reproduced a printed table of contents, similar to that of the earlier example. This may then be augmented by a printed list of figures, perhaps placing it on a new page, by interpolating the *same* reference data file a second time, *after* cancelling the active exclusion for the “figures” reference class, and explicitly selecting it instead:

```
.bp
.ce 1
.nop \fB\s'+2z'List of Figures\s'-2z'\fP
.sp 2v
.
.\" Cancel the preceding exclusion of "list of figures"
.\" references, and actively select them instead; (this
.\" will implicitly exclude all unclassified references,
.\" whence the "table of contents" is produced).
.\"
.toc exclude
.toc select figures
.
```

```
.\" Interpolate the reference data file a second time,
.\" to produce the list of figures.
.\"
.so foo.toc
```

Then, to prepare for production of the document body, and updating of the reference data stream file:

```
.\" Cancel the active "figures" class selection, and
.\" initialize the reference data collector.
.\"
.toc select
.toc file foo.toc
```

Recording of the updated reference data, for use in this second example, is performed in fundamentally the same manner as it is in the first illustration; for the *unclassified* records, which specify the table of contents entries, it is sufficient to call:

```
.toc put pageref \n% reference text ...
```

whereas, to institute the appropriate *figures* classification of the list of figures entries, the same macro call will suffice, *only* if it is framed by appropriate classification macro calls:

```
.toc classify figures
.toc put pageref \n% reference text ...
.toc classify
```

Finally, it may be observed that each of the preceding examples is fairly trivial; a more sophisticated working example, of the use of **groff_toc**, may be found in the *spdf-toc.tmac* file, which is provided as an exemplary accompaniment to the *groff-pdfmark* distribution.

AUTHORS

The **groff_toc** macro framework is provided as an adjunct to the *groff-pdfmark* package, which was written by Keith Marshall <keith@address.hidden>; formerly distributed with *GNU roff*, (albeit *without* support for the **groff_toc** feature), this is now independently maintained at, and distributed from Keith's *groff-pdfmark* project hosting web-site <https://savannah.nongnu.org/projects/groff-pdfmark/>, whence the latest version may *always* be obtained.

SEE ALSO

groff(1), **pdfroff**(1), **groff**(7)

More comprehensive documentation on the use of the *groff-pdfmark* macro suite, including a section which relates to integration with the **groff_toc** macro framework, may be found, in PDF format, in the reference guide "*Portable Document Format Publishing with GNU Troff*", which has also been written by Keith Marshall; the most recently published version of this guide may be read online, by following the appropriate document reference link on the *groff-pdfmark* project hosting web-site <https://savannah.nongnu.org/projects/groff-pdfmark/>, whence a copy may also be downloaded.