## NAME

groff_pdfmark – embed pdfmark controls in GNU roff PostScript output stream

## SYNOPSIS

Command line invocation:

**pdfroff –mpdfmark** [*option* ...] [*file* ...]
**pdfroff –m pdfmark** [*option* ...] [*file* ...]

Macro calls, from within **groff**(7) document source:

**.pdfmark** *arbitrary pdfmark instruction code ...*
**.pdfinfo** /<*key-name*> *record content ...*
**.pdfview** *initial page view specification ...*
**.pdfnote** [[**–T** "*title text ...*"] [\[**#** *comment ...*]]
[**–PD** <*paragraph-spacing-line-count*>] [\[**#** *comment ...*]]
[**–C** <*red-value*> <*green-value*> <*blue-value*>] [\[**#** *comment ...*]]
[**–O**] [**–I** <*icon-style-name*>] [**––**] [\[**#** *comment ...*]]
[ *first of multiple lines of annotation text ...* \[**#** *comment ...*]] ...
[*additional non-terminal line(s) of annotation text ...* \[**#** *comment ...*]]
[ ... ]
] *last (or only) line of annotation text ...*
**.pdfhref** <*opcode*> [*options ...*] [**––**] *descriptive text ...*
**.pdfbookmark** [**–T** <*tag*> | –N <*refname*>] <*level*> <*context ...*>
**.pdfsync** [**O** | **M**] ...

## DESCRIPTION

This manual page describes the capabilities, and, for those which are not separately documented in other manual pages, the usage of the collection of macros which are provided by the *groff-pdfmark* macro package.

The command line invocation synopses, above, relate specifically to the invocation of **groff**(1), (with *indirect* invocation, via the **pdfroff**(1) wrapper command, being *strongly* recommended, in order to take advantage of its multi-pass **groff**(1) processing, which is normally a prerequisite for successful operation of the various macros within the *groff-pdfmark* suite).

On the other hand, the macro calls synopses relate to the usage of the various *groff-pdfmark* macros themselves, from within any user-written **groff**(7) input file; each of these macros performs, respectively, one of the following functions:

**.pdfmark** Writes arbitrary *pdfmark* code to the **groff**(1) PostScript® output data stream, (via the **grops**(1) input stream).

**.pdfinfo** Inserts a **/DOCINFO** *pdfmark* record, with a specified *key-name*, and arbitrary content, into the internal document meta-data cache.

**.pdfview** Inserts a **/DOCVIEW** *pdfmark* specification record into the internal document meta-data cache.

**.pdfnote** Inserts PDF annotations, in a style resembling “sticky notes”, directly from within the **groff**(7) input data stream, into a PDF document; usage is described in **groff_pdfnote**(7).

**.pdfhref** Provides various capabilities, as described in **groff_pdfhref**(7), and selected by the *opcode* argument, for creating and linking to document reference marks, or for linking to internet resources.

**.pdfbookmark** Creates a *document outline* reference mark, at the current output position, and inserts a link, which both represents, and leads to the specified context, into the internal document outline cache; this is an *exact* analogue for the **pdfhref** macro, when it is invoked with its “**O**” operator, as described in **groff_pdfhref**(7).

**.pdfsync** Flushes content from the internal document outline and/or meta-data caches, to the PostScript® output stream.

With the exception of the **.pdfhref** and **.pdfnote** macros, which are comprehensively documented in the **groff_pdfhref**(7) and **groff_pdfnote**(7) manual pages respectively, further guidance on the usage of each

of the above macros may be found in the **USAGE** section, which follows.

## USAGE

### Invoking Arbitrary `pdfmark` Instructions

Each of the capabilities, which is supported by the *groff-pdfmark* macro suite, is implemented in terms of the fundamental

**.pdfmark** *arbitrary instruction code . . . /keyword*

macro; execution of this results in insertion, into the **grops**(1) output stream, of the *pdfmark* code:

```
[ arbitrary instruction code ... /keyword pdfmark
```

In general, it will usually be more convenient to use one of the higher level *groff-pdfmark* macros, rather than to invoke the **.pdfmark** macro directly; however, if it is desired to exploit any *pdfmark* feature, which is not supported by the available higher level macros, this may be achievable by use of the **.pdfmark** macro.

Notice that the **.pdfmark** macro supplies the initial "**[**" instruction mark, and the terminating **pdfmark** operator, for inclusion in the **grops**(1) output; thus, these should *not* be included within the specified macro arguments.

Conversely, the terminal */keyword* argument *is* required; it *must* be specified as one of the *pdfmark* feature object-name keywords, as documented in the the Adobe® PDFMark Reference Manual <https://www.adobe.com/go/acrobatsdk_pdfmark>. For example, a PDF document outline entry may be represented by **grops**(1) output similar to:

```
[ /Count 2
  /Title (A PDF bookmark with two children)
  /View  [/FitH \n(Y? u]
  /OUT
 pdfmark
```

(in which the **groff**(7) register reference, "`\n(Y?`", represents the computed vertical location of the bookmark on the current page). Although the preferred method for producing such *pdfmark* output is discussed below, under the **Creating a Document Outline** heading, this output, as illustrated above, may be reproduced by calling the **.pdfmark** macro, thus:

```
.pdfmark \
  /Count 2 \
  /Title (A PDF bookmark with two children) \
  /View  [/FitH \n(Y? u] \
  /OUT
```

It may be further noted that, whereas contemporary PDF document production tools may be reasonably tolerant of non-conformity, the **.pdfmark** macro *does* endeavour, insofar as is practicable, to format its output to conform to the Adobe® *Document Structuring Conventions*, so as to preserve compatibility with the broadest possible spectrum of such PDF document production tools.

### Defining Document Meta-Data

Whereas **groff**(1) and **gs**(1) — both of which are called by **pdfroff**(1) — will *automatically* insert *some* meta-data into any PDF document which they produce, there are some meta-data entries, which it may be desired to incorporate, each of which *must* be specified manually. Among those entries which, if it is desired that they should be specified, require an explicit manual specification, are those in the PDF **/DOCINFO** category with key names **/Title**, **/Author**, **/Subject**, and **/Keywords**.

While such meta-data may be specified directly, using the **.pdfmark** macro, for example:

```
.pdfmark /Title (An Example PDF Document) /DOCINFO
```

the **.pdfinfo** macro, which conforms to the syntactic calling convention:

**.pdfinfo** /<*key-name*> *string value . . .*

offers an alternative method for specifying such meta-data; the effect of the preceding example may be re-

produced by the call:

```
.pdfinfo /Title An Example PDF Document
```

It should be noted that, when the **.pdfinfo** macro is used to specify document meta-data, this meta-data will *not* be written immediately to the **grops**(1) output stream; rather, it will be cached internally, in a **groff**(7) diversion, until it is flushed to the output stream, by a subsequent invocation of the **.pdfsync** macro, (which may be called, conveniently, from within an end-of-input macro).

**Establishing an Initial Document View**

By default, when any PDF document is opened for the first time, the viewer application will position the viewport to display the top of the first page, at a magnification which is selected according to the configuration of the particular viewer application which is in use. Furthermore, it is dependent on the particular viewer configuration, whether a navigational side pane is displayed, and what contents may be displayed in such a pane; a common default is to show thumbnail views of the document pages.

The viewer application's default initial view configuration may, or may not, conform to the document author's preference; if not, alternative preferences may be specified by macro calls of the form:

**.pdfview** *view specification* ...

While it is not guaranteed that any particular viewer application will honour every possible *view specification*, this should be specified to conform with the requirements of the **/DOCVIEW** *pdfmark* feature object, as they are described in the Adobe® PDFMark Reference Manual <https://www.adobe.com/go/acrobatsdk_pdfmark>; typical usage might include:

```
.pdfview /PageMode /UseOutlines
.pdfview /Page 1 /View [/FitH \n(.p u]
```

In this example, note that the **grops**(1) "**u**" operator is required, to convert the view specification from **groff**(7) page co-ordinates, to a mapping within the PostScript, or PDF, page co-ordinate system.

**Creating a Document Outline**

A document outline is created by a series of calls to either the **.pdfbookmark** macro, (or the entirely analogous **.pdfhref O** macro); when called in the form:

**.pdfbookmark** [**–T** <*tag*> | **–N** <*refname*>] <*level*> <*context* ...>

this is *exactly* equivalent to:

**.pdfhref O** [**–T** <*tag*> | **–N** <*refname*>] <*level*> <*context* ...>

This is formally documented in the **USAGE** section of **groff_pdfhref**(7), under the subheading **Creating a Document Outline**; please refer to that manual page, for complete documentation of **.pdfhref O** usage, (and analogously, therefore, of **.pdfbookmark** usage).

Please note that, regardless of whether a document outline is created by use of **.pdfbookmark**, or **.pdfhref O**, the constructed outline is initially stored in an internal outline cache; this *must* ultimately be flushed to the PDF output data stream, by calling the **.pdfsync** macro, (typically from within an "end-of-input" processing macro), as described under the subheading **Flushing Internal Data Caches**, which follows immediately below.

**Flushing Internal Data Caches**

The *groff-pdfmark* macros use two internal data caches for intermediate storage of data, which may not yet be ready for committal to the PDF output data stream, namely:

- The meta-data cache, which is implemented as a **groff**(7) diversion, and is used for intermediate storage of meta-data which has been specified by use of the **.pdfinfo** and **.pdfview** macros.
- The outline data cache, which is implemented as a collection of internally named, and dynamically defined, **groff**(7) registers and strings; this stores information relating to the evolving structure of any document outline, until it has become sufficiently developed for committal to the PDF output data stream.

Both of these caches *must* be *explicitly* flushed, to avoid loss of data on completion of document processing; flushing may be achieved by calling the **.pdfsync** macro:

**.pdfsync** [**O** | **M**] ...

When the "**M**" argument is specified, the meta-data cache is flushed. This may be safely requested at any time, *after all* desired meta-data has been defined; typically, the request is deferred until all document input data has been processed.

When the "**O**" argument is specified, the document outline cache is flushed. This may be safely requested *only* when the next bookmark to be placed will be at the topmost level, (i.e. at level *one*), or when *all* document input data has been processed; however, it is unnecessary to *explicitly* request such outline cache flushes, other than at the end of document processing, since the **.pdfbookmark** — or **.pdfhref O** — macros will request them *automatically*, when it is safe to do so.

It is permitted to specify *both* the "**M**" and "**O**" arguments together, as a space-separated pair, in a single **.pdfsync** macro call; this requests flushing of *both* caches.

If the **.pdfsync** macro is called *without* arguments, this is interpreted as being equivalent to specifying *both* the "**M**" and the "**O**" arguments together.

Typically, it is convenient to specify a **.pdfsync** call, *without* arguments, in an "end-of-input" processing macro, to ensure that both caches are appropriately flushed when all document input has been processed; any macro package, which is designed to interoperate with *groff-pdfmark*, should automatically incorporate such an "end-of-input" processing request.

## FILES

*/usr/local/share/groff/site-tmac/pdfmark.tmac*

This implements the core functionality of the entire *groff-pdfmark* macro suite.

## AUTHOR

The **groff_pdfmark** suite of macros is provided by the *groff-pdfmark* package, which was written by Keith Marshall <keith@address.hidden>. This was originally included as a contributed component of the *GNU roff* software distribution, but 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. Contemporary releases of *GNU roff* no longer include this component package; any residual copies, which may be found within earlier *GNU roff* distributions, are likely to be obsolete.

## SEE ALSO

**groff**(1), **grops**(1), **gs**(1), **pdfroff**(1), **groff**(7), **groff_pdfhref**(7), **groff_pdfnote**(7)

More comprehensive documentation, on the use of the **groff_pdfmark** macros, and the *groff-pdfmark* macro suite in general, 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.