Producing PDFs with groff and mom

Deri James
and
Peter Schaffter

Contents

Producing PDFs with groff and mom

Deri James
and
Peter Schaffter

1. Introduction

PDF documents are intended to be "electronic paper," and, as such, take advantage of the digital medium in ways that PostScript documents do not. Chief amongst these are clickable links that point to named destinations, either within the documents themselves (internal links) or to remote web pages (external links), and the generation of a clickable document outline that appears in the Contents panel of most PDF viewers.

Using **groff** and **mom** to produce PDF documents results in the automatic generation of clickable document outlines (discussed below, "4. The PDF Outline"), and, if the **TOC** macro is included in the source file, entries in the printable table of contents can be clicked on as well when the document is viewed at the screen (see "5. Tables of Contents").

2. Using groff to generate PDF files

Groff provides more than one way to generate PDF documents from files formatted with the **mom** macros. One is to call **groff** directly, either with

```
groff [-Tps] -mom -m pdfmark doc.mom | ps2pdf - doc.pdf
```

which pipes output from the **grops** PostScript driver through **ps2pdf**, or

```
groff -Tpdf -mom doc.mom > doc.pdf
```

which uses the native PDF driver, **gropdf**. Alternatively, one may call the wrapper

```
pdfroff -mom -mpdfmark --no-toc doc.mom > doc.pdf
```

A fourth, preferred method is to use **pdfmom**, which is strongly recommended since it implements the full range of PDF features available in **mom**.

```
pdfmom doc.mom > doc.pdf
```

One reason to prefer using the native PDF driver (via **pdfmom** or **-Tpdf**) is that papersizes set within mom source files (see paper and page setup macros) do not require a corresponding **-P-p<papersize>** flag on the command line.

There are other minor differences between the methods, discussed here.

3. Creating PDF links with mom

Often, but not always, links in the body of a PDF document point to headings elsewhere in the same document. Creating these links is a simple process. First, identify the places to link to ("destinations"), then link to them from any place in the document.

3.1. Creating destination points at headings

The first step in creating links to a heading is to give the heading a unique destination name. With mom, this is done by adding **NAMED <id>** to the HEADING macro, where **<id>** is a unique identifier for the heading. For example,

```
.HEADING 1 NAMED intro "Introduction"
```

would, in addition to printing the head in the body of the document, identify the introduction by the unique id, "intro". This id, or name, can then be used to create links to the introduction from any part of the document.

Furthermore, **NAMED <id>** stores the text of the heading for use later on when linking to it (see "3.3. Creating internal links"). If headings are being numbered, the heading number is prepended.

3.2. Creating destination points at arbitrary locations

Any part of a document can be a link destination, not just headings. For example, say you create a table that needs to be referred to from other parts of the document. You'd identify the location of the table by placing

```
.PDF_TARGET <id> "<text>"
```

just above the table in the source file. As with **HEADING**, **<id>** is any unique name. **<text>** is optional. **<id>** can now be linked to from anywhere in the document.

3.3. Creating internal links

Internal links are clickable text areas that allow you to jump to named destinations within a document. (See here for a description of external links.)

Internal links are created with the macro **PDF_LINK**, which takes the form

```
.PDF_LINK <id> [PREFIX <text>] [SUFFIX <text>] "<hotlink text>"
```

where **<id>** is a named destination point elsewhere in the document (see "3.1. Creating destination points at headings" and "3.2. Creating destination points at arbitrary locations").

PREFIX <text> and **SUFFIX <text>**, both or either of which are optional, are printed around the clickable area but do not form part of the link itself.

<hotlink text> is the text that should be clickable, identifiable in the PDF document by the colour assigned to links (see "3.5. Assigning a colour to links").

If the hotlink text ends in **"*"**, the asterisk is replaced by the text of the destination point, assuming it's a heading. If the hotlink text ends in **"+"**, the replacement text is surrounded by quotes.

Using our HEADING example, above, the following invocation of **PDF_LINK** would produce a clickable link to the introduction:

```
.PDF_LINK intro PREFIX ( SUFFIX ). "see: +"
```

In the text, the link would look like this: (see: "1. Introduction").

3.4. Creating external links

External links are clickable text areas whose destination is a URL. Clicking on them causes a browser window to pop up with the destination address.

The format of the macro to create external links is similar to the one for creating internal links:

```
.PDF_WWW_LINK <url> [PREFIX <text>] [SUFFIX <text>] ["<hotlink text>"]
```

<url> is any valid URL, usually a web address; **PREFIX<text>** and **SUFFIX<text>** have exactly the same meaning, as does **<hotlink text>**, which furthermore accepts the same expandos, **"+"** and **"*"**.

If no hotlink text is given, then **<url>** is used as the text. If hotlink text is given and ends in **"*"**, the asterisk is replaced by the URL. If it ends in **"+"**, the URL is surrounded by quotes. As an example,

```
.PDF_WWW_LINK http://www.schaffter.ca/mom/momdoc/toc.html
```

would open mom's online documentation at http://www.schaffter.ca/mom/momdoc/toc.html. The same, with **"here"** supplied as hotlink text, lets you click here instead.

3.5. Assigning a colour to links

The colour of links is set with

```
.PDF_LINK_COLOR <xcolor> | <newcolor> | <r g b> | <#rrggbb>
```

where **<xcolor>** or **<newcolor>** are the names of colours already initialized with XCOLOR or NEWCOLOR. If you prefer to define a new colour (using the RGB colour scheme), enter it either as 3 numbers between 0.0 → 1.0 or as a 6 character hex string. Thus

.PDF_LINK_COLOR #ff0000 and **.PDF_LINK_COLOR 1.0 0 0**

both lead to mom using red links.

The default colour can be restored by calling **PDF_LINK_COLOR** with no parameter.

> **Note:** The decimal scheme for creating colours must be used if a file is to be processed with '**groff −Tps −mpdfmark**', '**pdfroff**', or '**pdfmom −Tps**'.

4. The PDF Outline

Most PDF viewers provide a panel that displays a document's outline, similar to a table of contents. Clicking on an entry navigates directly to the appropriate place in the document.

Mom generates PDF outlines the same way she populates her own table of contents: by intercepting calls to the **HEADING** macro, as well as to the various title and chapter macros used in namimg documents, and allocating each a hierarchic level.

Covers, titles/chapters, and the table of contents are all assigned to level 1. Subsequent headings are assigned to n+1, where n is the level given to **HEADING**.

The PDF outline can sensibly recover from skipped or omitted heading levels; the printed table of contents cannot. Users are therefore advised to use headings in logical order, not for typographic effects.

4.1. Opening and closing levels

A level is said to be open if one or more levels beneath it is visible in the PDF outline. Closed levels have at least one level beneath them that is not visible unless the closed link is clicked. It is common for only the first two levels to be open so the outline doesn't look cluttered.

To establish which levels should be open by default when a document loads, use

```
.PDF_BOOKMARKS_OPEN n
```

where **n** is a number specifying at which level all subsequent ones should be closed.

If, at any point in the document, you specify

```
.PDF_BOOKMARKS_OPEN NO  \" or any other text argument
```

then all subsequent bookmarks will be closed until **PDF_BOOKMARKS_OPEN** opens them again.

4.2. Suspending/disabling collection of outline entries

Suspending the collection of entries for the PDF outline is accomplished with

```
.PDF_BOOKMARKS OFF
```

Mom's default is to collect entries, so if the command is placed at the start of a document, it disables entry collection completely. Elsewhere, it suspends collection until you re-enable it with

```
.PDF_BOOKMARKS  \" i.e. with no parameter
```

4.3. The PDF window title

While not strictly part of the PDF outline, the title of a document can be displayed as the document viewer's window title. The macro to accomplish this is

```
.PDF_TITLE "<window title>"
```

It can take any text, so the viewer window title need not be the same as the document's title.

> **Note:** The macro, **DOC_TITLE**, always invokes **PDF_TITLE**. If this is not what you want, you can remove the window title by issuing
>
> ```
> .PDF_TITLE "" \" ie. with a blank argument
> ```

5. Tables of Contents

5.1. Generating a Table of Contents

To generate a printable Table of Contents for any document, simply insert the macro, **TOC**, as the last line of the source file. (Formatting of the printable Table of Contents is discussed in detail in the mom documentation). When the file is processed and loaded in a viewer, entries in the Table of Contents will be clickable links.

Whichever link colour is active at the end of the document, prior to **TOC**, will be used for the Table of Contents links.

5.2. Positioning the Table of Contents

If **groff**'s PostScript device (**−Tps**) is used to process a mom file, the Table of Contents is printed at the end of the document. When this is not desirable, the PostScript output from **groff** must be processed with **psselect** in order to place the TOC in the preferred location.

When using mom and **groff**'s native pdf device (via **pdfmom** or **groff −Tpdf**), positioning of the Table of Contents can be done within the source file.

The command to control the placement of the TOC is

```
.AUTO_RELOCATE_TOC [<position>]
```

where the optional **<position>** can be one of these keywords:

TOP**(ie. at the very start of the document)*
BEFORE_DOCCOVER
AFTER_DOCCOVER
BEFORE_COVER
AFTER_COVER

*__Note:__ Documents without a COVER or DOC_COVER require the **TOP** argument.

It is normally not necessary to supply a keyword, since **AUTO_RELOCATE_TOC** places the TOC after the DOC_COVER, if there is one, or the first COVER when no DOC_COVER is present. In rare instances where it is desirable to place the TOC somewhere else in the document, there are two low-level commands, **.TOC_BEFORE_HERE** and **.TOC_AFTER_HERE** which place the TOC either before or after the current page.

These last two commands have a small catch: although the TOC will appear where specified, the "Contents" entry in the PDF outline, which observes a hierarchy of levels, will assign the TOC to level 1, possibly disrupting the visual ordering of levels in the outline.

6. pdfmom: Simplifying PDF output

As explained in the section 2. Using groff to generate PDF files, there are two established methods for creating PDF files with **groff**: the original method, ie. passing the **−Tps** and **−mpdfmark** options to **groff** (or using **pdfroff**, which does this for you); or the newer **−Tpdf**, which produces PDF files natively.

6.1. The problem of forward references

Both methods encounter difficulties when dealing with forward references; that is, when a link *earlier* in a document refers to a destination *later* in the document and the link text terminates with one of the expandos, **"*"** or **"+"** (explained here). Mom doesn't know what text to put in the expando because it has not yet been defined. This means that **groff** must be run multiple times to find the unknown text.

The program **pdfroff** exists to handle these multiple runs, but it imposes some limitations on the PDF features available with **mom**.

6.2. pdfmom

pdfmom performs the same function as **pdfroff**, and is the preferred, trouble-free way to generate PDF documents from a mom source file. Like **pdfroff**, it is a frontend to **groff** and accepts all the same options (see **man groff**).

Called as-is, **pdfmom** accepts all the same options as **groff**, and requires no additional flags. PDF generation is performed by **gropdf**, **groff**'s native PDF driver:

```
pdfmom doc.mom [groff opts] > doc.pdf
```

If a **−Tps** option is supplied, **pdfmom** hands control over to **pdfroff**, and both **groff** and **pdfroff** options may given. The resulting PDF is produced from PostScript output fed into **ghostscript**.

```
pdfmom −Tps [pdfroff opts [groff opts]] doc.mom > doc.pdf
```

For either invocation, it is not necessary to add **−mom** or **−mpdfmark**, as these are implied.

If Encapsulated PostScript or plain PostScript images have been embedded in a document with PSPIC, the **−Tps** option must be used. In most other cases, **pdfmom** with no **−T** flag is preferable.

6.3. Setting papersize within a source file

A significant convenience afforded by using **pdfmom** (or **groff** with the **−Tpdf** flag) is that papersizes or page dimensions set within mom source files (see paper and page setup macros) do not require a corresponding **−P−p<papersize>** option on the command line. It is even possible to create documents with unequal-sized pages.

6.4. Differences between pdfmom and pdfroff

Several features described in this manual are not available when using **pdfmom** with the **−Tps** option, or when using **pdfroff**, or **groff −Tps −mpdfmark**:

- Relocation of the Table of Contents is not supported. The TOC appears at the end of the document; **psselect** must be used to re-order pages.
- If a link crosses a page boundary, it will stop being a clickable hotspot on subsequent pages.
- When establishing whether PDF outline levels are open or closed, only the numerical parameter to **PDF_BOOKMARKS_OPEN** has any effect.
- PDF_LINK_COLOR only accepts colour definitions in decimal notation.

7. Comparison of −Tps/−mpdfmark with −Tpdf/−mom

−Tps/−mpdfmark

- does not support all the features described here
- accepts images and graphics embedded with PSPIC
- is mature and well-tested code

−Tpdf/−mom

- facilitates embedding fonts directly in the PDF file (if the **−P−e** flag is given on the command line)
- sets papersize from within the source file, circumventing the need for the papersize flag (**−P−p<papersize>**) on the command line
- is not compatible with PRINTSTYLE TYPEWRITE underlining (e.g., of italics)
- generally produces larger files; these can be reduced by piping the output through **ps2pdf**

> **Note:** Owing to a known bug, PDF files piped through **ps2pdf** lose some of their metadata, notably the window title set with **PDF_TITLE**.

- is newer code with less testing