Gropdf Enhancements

Deri James

ABSTRACT

I have made a number of changes, mainly to gropdf, and they need to be tested before release to the wild. These changes can be tested by checking out the git branch "deri-gropdf-ng" before building.

4 July 2023

Gropdf Enhancements

Deri James

The groff PDF output driver

Gropdf has been enhanced in a number of ways:-

- Fonts can be subsetted to only include glyphs required to render the document, this can substantially reduce the size of the completed pdf file, if very large fonts have been used.
- Gropdf now produces pdfs which conform to the PDF 1.7 standard (ISO32000). This also reduces the size of the pdf file.
- A new macro, ".pdfpagenumbering" allows the page numbers in the overview panel to be controlled. It is common for a document to start with roman numbering and then reset to decimal 1 for the body of the document, this can now be specified for the overview panel as well.
- When multiple man pages are bundled into a collection (such as the groff-man-pages.pdf and the LinuxManPages.pdf) intra page links are clickable.
- Unicode support for bookmarks, has been implemented. If bookmarks include unicode characters they are stored as UTF-16 strings and will be displayed correctly in the overview panel.
- Environment variable GROPDF_OPTIONS.

Font Subsetting

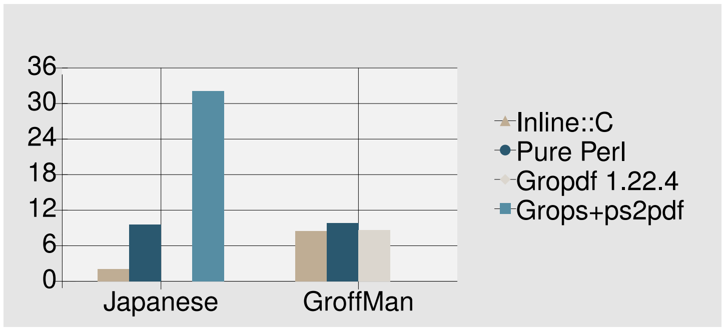
Deconstructing and the subsetting very large fonts is slow in a scripted language so there is an option to use a routine written in C rather than perl. To use the fast version you can install the perl module Inline::C and gropdf will use the faster version. Using fonts with more than 1000 glyphs it is recommended to install Inline::C. The module is widely available packaged usually as perl-Inline-C.rpm or libinline-c-perl.deb.

If you are using a large font without having installed Inline::C, you will see this warning:-

```
./gropdf:groff.7: warning:
Font 'SauceHanSansJP-R (GR)' has 18482 glyphs
You would see a noticeable speedup if you install the perl module Inline::C
```

This can be ignored if you are happy to use the pure perl version

The first test document is 17 pages in Japanese which uses 3 fonts, each of which has over 18,000 glyphs, and the second is the groff-man-pages.pdf document of 375 pages using the 35 standard fonts all of which are embedded. The relative timings (in seconds) are:-

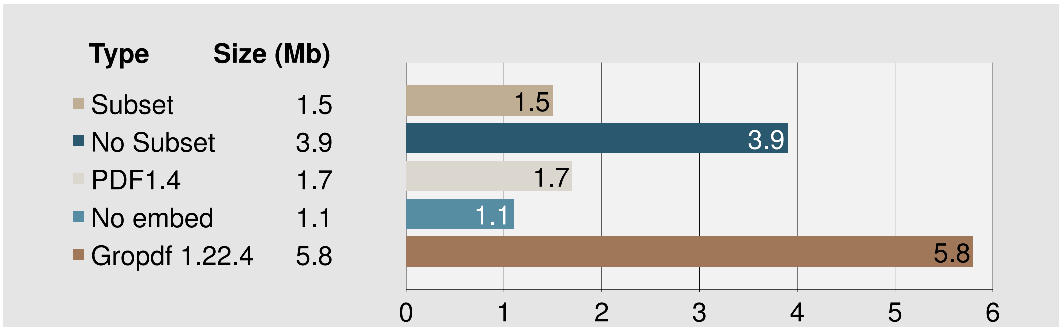


Gropdf 1.22.4 is unable to produce the Japanese document since it could not cope with more than 255 glyphs. Grops+ps2pdf is the total time for creating the pdf (no overview pane - UTF-16 not supported).

Inline::C creates a work directory in your home directory called "_Inline", if you would prefer it used somewhere else then set environment variable PERL_INLINE_DIRECTORY to a writeable directory.

More compact pdf - PDF1.7

The major advantage of using font subsetting is a reduction in the overall size of the generated pdf file, this shows sizes for the 375 page groff-man-pages.pdf.



Starting from the top, bars 2 and 5 did not use font subsetting. The difference between bars 1 and 3 is the more compact format when using PDF 1.7. Bar 4 produces the smallest file but no fonts are embedded at all so the pdf viewer will choose whichever font it considers is best, which can produce strange results. The third bar is using this gropdf but setting a flag to request it output in PDF1.4 format.

The default for gropdf is to subset, use PDF 1.7 format, and not embed the standard fonts. It is recommended to always embed all fonts with the -P-e flag to ensure faithful reproduction of the document.

New macro .pdfpagenumbering

\X'pdf: pagenumbering *type prefix start*'

This is used to control the page numbering shown in the pdf reader's outline pane which contains your bookmarks. Normally the page numbers shown against the bookmark is the physical page number in the file, But this may not match the different page number styles within the document.

In a single document there may be a cover sheet (which has no page number), a TOC (which uses lower case roman numbers), and the main body of the document (which has decimal page numbers). Use this command somewhere on the page where the numbering system changes, once changed the numbers will automatically increment until the number system changes again, so don't call for every page, just when you want to change the numbering.

The parameters are:-

type This specifies the type of numbering to use for this page onward. It should be one of **“Decimal | Roman | roman | Alpha | alpha”**. Only the initial letter is relevant. The alphabetic number systems use A-Z (then AA-AZ ... ZA-ZZ). The *type* may also be ”**“**. which means no numbering system is chosen, but you may still provide a *prefix* to have a custom name (such as "Cover");

prefix Provides a string to insert before the number. If the document has an Appendix with page numbers in the form A-*n*, the *prefix* would be set to “A-” and the *type* would be **Decimal**.

start Gives the start number for the incrementing page numbers in the outline pane. If no value is given for *start* it will default to 1, which is usually correct.

The convenience macro for this command is “ **.pdfpagenumbering** *type prefix start* ” using ’.’ for preceding missing values, or just “ **.pdfpagenumbering**” on its own to have no page numbers shown in the outline pane.

The pdf viewer will be aware of these "virtual" page numbers and should show the virtual page number as well as the physical page number. Also, if you enter a page number to jump to, it will use the virtual page number, so entering "i" may jump to the first page of the TOC if you have used roman numbering for the TOC.

Man page collections

If you bundle multiple man pages into a single groff run, the output will be a single file containing the man pages and any intra-page links to other pages in the collection will be clickable.

Unicode bookmarks

Preconv (which is called by including -k on the groff command) converts UTF-8 characters to \[uXXXX] symbols. When these are passed to .pdfbookmark or .pdfinfo they are converted to UTF-16 characters.

GROPDF_OPTIONS

A new environment variable GROPDF_OPTIONS is read before parsing command line options. So you could instruct gropdf to only produce to the PDF1.4 format by including export GROPDF_OPTIONS="--pdfver=1.4" in your shell login script. Do not include -P as part of the flags.

Other Changes

New file "GMPfront.t", is the front page of the groff_man_pages.pdf, also control the overview page numbering so the front page has no page number and the first man page is on page 1. It is only used in the build of groff_man_pages.pdf so should not be installed.

Changes to "doc/doc.am" to utilise "pdfmom --roff" rather than plain groff to produce groff_man_pages.pdf since this takes care of the forward references in the document.

In "src/devices/gropdf/gropdf.1.man", document that gropdf is no longer restricted by the number of glyphs in a single font, and specify the new feature which can control the appearance of page numbers in the overview panel.

Changes to "src/devices/gropdf/pdfmom.pl" have generalised its use. Previously it was only useful for generating documents which used the mom macros, now, if you pass the --roff flag it can be used with any macro set. To satisfy forward references in an ms file you could use "pdfmom --roff -ms" to create a pdf. Further, if you set up a symbolic link to pdfmom using the template "pdf<macroset name>" (such as "pdfms") when you use this command you don't need to include "-ms" on the command line. Since this command only produces pdfs there is no need to include -Tpdf or -mpdfmark.

An extra column has been added to the output of "src/utils/afmtodit/afmtodit.pl" which is the 4 digit hex code of the character, as discovered from the AGL_to_unicode mapping table. This information is needed to support the inclusion of UTF-16 characters in the overview panel. For certain special characters, such as \(em it is not possible to derive the unicode number for the character from the current information. The extra column has no impact for grops, it ignores the new column. It is also useful when the groff name for the character is a composite, such as the glyph Hcircumflex which groff knows as \[u0048_0302] but the actual unicode number is 0x0124.

Changes to the .MR macro in "tmac/an.tmac" allow them to be treated as clickable links, only if they point to a man page which is part of the same collection. Single man pages never have links in them. This is what enables the intra-page links in groff_man_pages.pdf.

The file "tmac/anmark.tmac" is another support file for creating the groff_man_pages.pdf, the same as GMPfront.t above. It is just used in the build and should not be installed.

If the flag -dpaper= is given to groff the selected papersize is also passed to gropdf if -Tpdf is given. If you also pass the -P-p flag to groff it will take predence. This allows a different media size to be used for the pdf than groff uses for typesetting, should this be what you require.

A new macro .pdfpagenumbering has been added to "tmac/pdf.tmac" which controls the page numbers shown in the overview panel. The text used in bookmarks are no longer "cleaned" by using .asciify, the raw text is now passed to gropdf where it is now cleaned and may be converted to UTF-16.

Preparation for replacing the pseudo slanted lowercase greek characters (used in equations) with real glyphs, i.e. provide a Symbol-Slanted pfb font for gropdf. At the moment it requires the following intervention to make it work with test-groff:-

Copy files SS and StandardSymSL.pfb from the source to the build directory.

Add the following line to the download file:-

```
    Symbol-Slanted  ./StandardSymSL.pfb.
```

This should make the output of equations using -T pdf exactly match the postscript equivalent. It allows the subtle italic corrections to be applied. It requires integration into the groff build system, see the relavent git log entry.

Experimental gropdf flag --opt=

During development I introduced an option flag, which turn on/off some pieces of the new code. The flag can be set by passing -P-opt=n on the command line. Where "n" is a number whose bits control the following:-

1 = SUBSET otherwise include entire font as now. (If someone has a document using a font which produces a bad pdf which can't be viewed, repeating with this bit unset will prove if the problem is with the subsetting, and also if they send this pdf, with a source file, I will be able to extract the actual fonts and use them for diagnosing/fixing the problem).

2 = USESPACE this uses a space character in text to separate words, otherwise use a horizontal motion. The difference is this:-

```
[(This is groff)] Tj # using space.
or
[(This) 250.000 (is) 250.000 (groff)] Tj # using horizontal motion.
```

The first method is preferable since it is more compact, but there are some fonts where it can't be used. Usually this is because there is no space character in the font, or it is called something else (I've seen u0020 and u00a0). If gropdf uses the compact method in this situation it is likely spaces will be shown as the .notdef glyph. Hopefully gropdf detects which fonts can't use the compact format and automatically adjusts, but this bit can be unset to force gropdf to use horizontal motion always. It can also be used to check whether the width of the space glyph as used by the pdf viewer is the same size as groff uses.

4 = COMPRESS when unset the pdf instructions (such as above), are not compressed so can be viewed in a text editor.

8 = NOFILE if set then no fonts are embedded in the pdf, even if the download file says they must be! Not particularly useful but does produce the smallest files.

The default value is 7. (SUBSET | USESPACE | COMPRESS)

I don't intend to release with the --opt=flag, it may be better if we can either drop some or think of separate flags, or add them to the -d flag, since some of them are definitely for debugging!

If, while using the new version gropdf, you notice a problem using a particular font which may manifest itself by the viewer showing an error message (mupdf gives helpful diagnostics), or the document is using the wrong font, please run the document with the flag -P--opt=6, to check if the problem is to do with font subsetting or not. If this "fixes" the issue I would be very grateful for a bug report containing a sample source file plus the two pdfs, with and without -P--opt=6, rather than just the incorrect version.

Known Issues

I am aware of the following issues.

pdfmark destinations

The named destinations, i.e. .HEADING n NAMED name "text" must not contain special characters, e.g. unicode converted by preconv. The "text" may contain unicode and any special characters. If you see warnings of the type " warning: special character 'u0413' not defined" this may not affect the pdf at all, but you may be able to suppress the message by including an -f flag on the command line specifying a font family which includes the missing unicode glyphs.

If you do use unicode for the destination name, such as:-

```
.HEADING 1 NAMED Гуляйпольщина "Гуляйпольщина"
```

It may work, but you will see this error:-

```
toff:<standard in err a special character is not allowed in an identiier
troff:<standard input>:32: error: bad string definition
```

And there is a probability it may not be quite right!

expandos

The use of "expandos" (as explained in the document "Producing PDFs with groff and mom" is not supported if you are using unicode for the text, you have to insert the text yourself. So code such as:-

```
.HEADING 1 NAMED Russian "Гуляйпольщина"
...
.PDF_LINK Russian PREFIX ( SUFFIX ) "see: +"
                                          ^
                                          |
                                          expando
```

Will cause this error:-

```
troff: ../src/roff/troff/input.cpp:523: static int input_stack::finish_get():
  Assertion `level == 0' failed.
groff: error: troff: Aborted (core dumped)
```

Surprisingly not at the point the expando is used, but at the end of the document. To avoid the problem, until it is fixed, use this instead:-

```
.PDF_LINK Russian PREFIX ( SUFFIX ) "see: \[dq]8. Гуляйпольщина\[dq]"
```

Where the text in the link is a copy of the text used when the destination "Russian" was created. If the text does not contain unicode characters expandos work normally.