Tutorials

Introduction to pdfTEX*

Thomas Feuerstack

Abstract

The following article is aimed at beginners and/or those interested in quickly creating PDF documents using pdfTEX, without having to work intensively to gain background knowledge.

We will point out problems that frequently occur while creating PDF documents, as well as their solutions — of course, we don't claim to solve everything.

1 Introduction

If you look around in TEX newsgroups or discussion lists, you will quickly see that a large number of the questions posed deal with bringing together TEX and Adobe's Portable Document Format (PDF).

Beginners in particular seem to get caught on the same rough edges with amazing regularity. Often the first problem is just how, for example, you can convert a TEX document into PDF. Once this hurdle is jumped, there is likely to be difficulties with using graphics, and fonts in particular.

* This article originally appeared in *Die TEXnische Komödie* 2/2001, in German, with the title "Einführung in pdfTEX"; it was translated by Steve Peter, and is published here with permission.

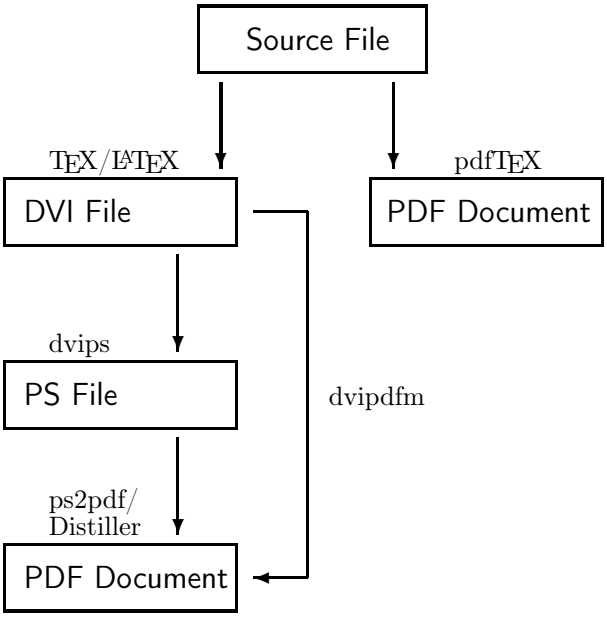

Figure 1: Even if not all roads lead to PDF, at least there are several.

In brief: most of these "rough edges" are easy to master with simple techniques — as long as you know what to do. This article is for those who aren't (yet) in the know.

2 From TEX/LATEX to PDF

It is well-known that all roads lead to Rome, nothing is impossible, and only you can make yourself happy. It therefore shouldn't be surprising that there isn't just one way to put your TEX file on the road to PDF, but more than a half dozen ways, given a bit of creativity; Figure 1 gives a roadmap.

In principle, we can differentiate roughly the two following ways.

The classic way: The document is TEXed as usual and converted to PDF with the aid of a special driver (e.g., `dvipdfm`) or via PostScript (`dvips` followed by `ps2pdf` or Distiller).

The elegant way: pdfTEX creates the PDF format you want directly from your document — and without further detours.

Since the second alternative is usually not only faster to describe, but also in fact faster to use, let's concentrate on it.

3 12 characters to make your document "portable"

Simply put `\pdfoutput=1` in the preamble of your document, and instead of the usual DVI file, you'll get PDF. If however, you get neither PDF nor DVI, but an error message on your screen: `Undefined control sequence...`, then you've tried to run (as you previously did) `latex` instead of `pdflatex`. You should change this right away.

Just by adding `\pdfoutput=1` you got the desired effect; you can find the whole set of specific pdfTEX sequences in [6]. With just a little more effort (29 characters to be exact) you can drastically increase the effectiveness of your PDF output.

Including `\usepackage[pdftex]{hyperref}` as the last package in the preamble of your TEX file not only causes PDF output, but also gives you the following functionality for free:

- All tables, such as table of contents, table of figures, etc., will automatically be linked to the text. Clickable text is shown surrounded by a colored frame.
- A list of PDF bookmarks is created from the structure of the table of contents. Using these bookmarks in a manner similar to the table of contents itself, you can quickly navigate to various parts of the text.
- Cross references within the text, like footnotes, indices, etc., are hyperlinked to the text.

If you now load your freshly-made PDF document into Adobe Reader (or any other PDF reader, for that matter), it might already look like Figure 2.

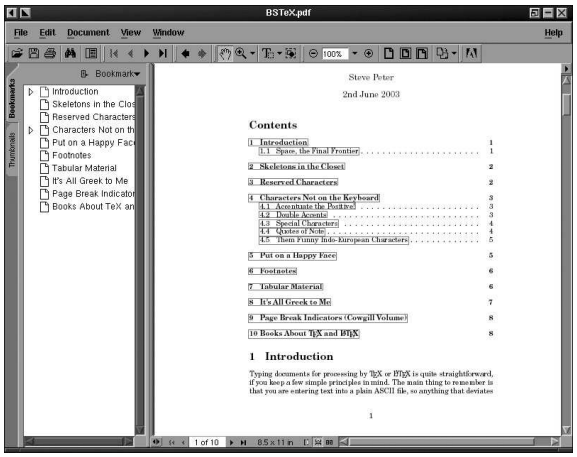

Figure 2: The result of the `hyperref` package.

4 Of course `hyperref` can do much more

By using a selective set of `hyperref` options you can tweak the appearance of your PDF document to almost the smallest detail. The options, as is usual in declaring LATEX packages, are given in square brackets before the actual package name. We can distinguish between parameters that are simply passed

in (as in the example above with the parameter `pdftex`), and those where the parameter requires a value, such as `pdfstartview=Fit`.

Thus, a typical declaration of `hyperref` might look like the following.

```
\documentclass{article}
   ...
\usepackage[pdftex,a5paper,%
            pdftitle={The Ducks},%
            pdfauthor={Mother Goose},%
            colorlinks=true,%
            linkcolor=blue%
           ]{hyperref}
   ...
\begin{document}
   ...
```

The complete range of all specifiable options can be found in the documentation by Sebastian Rahtz [5], with a somewhat more complete description in [4]. The possibilities described in the manuals may seem to be impossibly many, and often just as hard to understand. Therefore, let's now look at a few commonly used options more closely.

4.1 General options

The **'backend driver'**, in our case `pdftex`, informs the `hyperref` package how you want to create your PDF document. If, in spite of this article, you choose some other way to create PDFs (cf. Figure 1), you will need to change this option correspondingly.

Since there may be other LaTeX packages in the document that depend on a backend driver, for example the `graphicx` package, you can specify the driver as an option to `\documentclass`; the option is then passed to all packages that need it, and doesn't have to be given each time you call `\usepackage`.

The **paper size** is set to `a4paper` by default, but you can change it to `a5paper`, `b5paper`, `letterpaper`, `legalpaper` or `executivepaper`.

The two options `draft` and `debug` play a special role. While `draft` turns off all `hyperref` functionality, `debug` places into your log file additional diagnostic information.

4.2 Screen display and document information

Normally your document will be displayed by Acrobat Reader as in Figure 2. You can change this with the help of the following options:

`pdfpagemode` determines how Acrobat Reader is opened. Possible values are `UseOutlines` (default), to show the bookmarks in the left frame; `UseThumbs`, to show the individual pages as thumbnails in the left frame; `None` and `FullScreen` (synonyms), to show the document without menubar, bookmarks, thumbnails, or a left frame in general.

The *zoom factor* of the document can furthermore be configured by the following:

`pdfstartview` sets the size of the page first displayed (normally page 1, but that can be changed with `pdfstartpage`). Possible values are `Fit`, to show the whole page; `FitH`, to fit the width of the page in the window; or `FitB`, to fit the width of the contents to the window.

`pdfview` determines the viewing size of a page that is opened via hyperlink. The possible values are the same as those for `pdfstartview`; by default it is the same viewing size as the previous page.

The *additional information* about the document, accessed in Acrobat Reader with the menu command File – Document Properties – Summary, typically includes the document title, the author's name, as well as keywords to enable automated searches.

You can set these values with options such as `pdftitle`, `pdfauthor`, `pdfkeywords`, ... Our list of options is getting slowly but surely longer.

```
\documentclass{article}
   ...
\usepackage[pdftex,a5paper,%
            pdftitle={The Ducks},%
            pdfauthor={Mother Goose},%
            pdfkeywords={Ducks Fowl},%
            pdfpagemode=FullScreen,%
            pdfstartview=FitB%
           ]{hyperref}
   ...
\begin{document}
   ...
```

4.3 Configuring bookmarks

You can turn off automatic creation of bookmarks with `bookmarks=false`, which (as you learned in the last section) means that the option `pdfpagemode` is set to `None`.

Only top level bookmarks are typically shown; to see lower bookmarks, you have to manually "expand" the hierarchy. You can change that with the `bookmarksopen=true` option, which will cause the entire bookmark tree to be displayed.

If the bookmarks become a bit too prominent when you do that, you can reach a usable compromise with the `bookmarksopenlevel` option.

4.4 Colors

As you will quickly see, links created by `hyperref` are colored. Links are colored differently depending on the type of reference — for example, internal links (like in the table of contents) are red, blue for web links or email addresses, green for links to the bibliography, and so forth.

The options `citebordercolor` (bibliography), `linkbordercolor` ("normal" links), `urlbordercolor` (web links and email), etc., override the default color values. You enter the color you want as three RGB values, which must be $0 \leq \text{RGB} \leq 1$. For example, `linkbordercolor={0 0 1}` will change the standard red of "normal" links to blue.

The width and style of the link box can be configured via `pdfborder`, which also requires a numeric triplet. `pdfborder={0 0 5}` changes the line from 1 point to 5 points, with the result that you can hardly see the text.

Especially interesting effects can be achieved by adding an optional parameter to the numeric triplet. For instance, `pdfborder={0 0 1 [3]}` creates a box with a 1 point border that is broken — the higher the value of the optional parameter, the longer the breaks in the line.

`colorlinks=true` removes the box around the link (which `pdfborder={0 0 0}` will also do), but the text of the link is now colored. The color scheme for boxes is also valid for colored text links, and can be changed with the options `linkcolor`, `citecolor`, `urlcolor`, etc.

What's good about this is that for changing colors used for text, you can use the color names in the `color` package; in other words, a declaration such as `linkcolor=blue` works just fine.

What's not so nice about this is that when you print, colored text is of course printed in color, which means the table of contents is red, or gray on a black-and-white printer.

4.5 Global settings

Settings that are the same for several documents need not be entered each time via the `\usepackage` declaration. They can instead be placed into a systemwide `hyperref.cfg` file.

For example, if all of our documents use `colorlinks` and the "normal" link color should be blue, the following will accomplish it:

```
%% hyperref.cfg: Global Settings
\hypersetup{colorlinks=true,%
        linkcolor=blue,%
        pdfproducer={Birds-of-a-Feather},%
        pdfstartview=FitB}
```

Then the list of options for the `hyperref` package can be reduced to only those needed just for this document.

```
\documentclass{article}
   ...
\usepackage[pdftex,a5paper,%
            pdftitle={The Ducks},%
            pdfauthor={Mother Goose},%
            pdfkeywords={Ducks Fowl}%
           ]{hyperref}
   ...
\begin{document}
   ...
```

Options passed in the `\usepackage` declaration override options from the `hyperref.cfg` file that have the same name.

5 Your own links

Although the `hyperref` package already automatically links for you almost everything in a TEX document that ought to be, you may actually want to add your own links to it. Links to URLs (i.e., to Web or email addresses) and document-internal links should be differentiated.

Links to URLs can be made quite simply with the command `\href`.

```
A new day dawns in
\href{http://www.fburg.com}{Featherburg},
home of the richest
\href{mailto:d.duck@fburg.com}{duck}
in the world.
```

In the example above, the words `Featherburg` and `duck` are colored according to the setting given by `colorlinks`. By clicking on `Featherburg` we will probably land on the home page of the city; by clicking on `duck` you can send D. Duck an email — but don't bother asking for money!

For *hyperlinks within a document* you need to specify both the `\hypertarget` and `\hyperlink`. With `\hypertarget` you define a jump location (also known as an anchor), which `\hyperlink` will then target.

```
In order to protect his financial
interests D. Duck has lived in a
\hypertarget{trove}{safe}for decades.
...
``Let's go'', the safecrackers shouted,
``Today it's the
\hyperlink{trove}{tightwad}'s turn!".
```

In this example, the word `tightwad` is made clickable — using the mouse button will send us directly to the `safe`. The connection between link and target is made by the freely-chosen word `trove`.

6 Inclusion of graphics

Conventional wisdom has it that a picture is worth a thousand words — which has lead many an author to the conclusion that the value of their work would increase according to the number of graphics it contained.

The most complicated graphic that a pdfTEX user puts into their document is generally the first one. So let's start at the beginning.

If you have aready put graphics into your "normal" LaTeX documents, ideally you should find there the following constructs:

```
\documentclass{article}
  ...
\usepackage{graphicx}
  ...
\begin{document}
  ...
If we want to use examples from the world
of comics, we can include pictures.

\begin{figure}
\includegraphics{entenhausen.eps}
\end{figure}
...
```

After you put your document on the road to PDF by including the `hyperref` package, your first `\includegraphics` command [1] will likely cause an error. pdfTEX directly supports the inclusion of JPEG, PDF and PNG graphic files — but not EPS!

I started a new paragraph here to allow you some more time to let the shock sink in. For years, you've struggled to get all the necessary graphics into Encapsulated PostScript, and now this.

Not to try your patience further, but I want to offer here an explanation of why this is, and not unnecessarily; note quite simply that pdfTEX has considerable difficulty (to put it kindly) with PostScript constructs of any sort, including the beloved package `pstricks`.

But keep your chin up, and to close this section, consider the following:

- In the future, you won't have to spend any more time converting to EPS, since most graphics programs directly support the formats given above.
- For those cases where you already have EPS pictures included, but don't have copies in any of the above-mentioned formats, in most cases the `epstopdf` converter, which ought to be included in every TEX installation, will handle the conversion.

7 Fonts and typefaces

Finally! You're using the `hyperref` package, and you've got it all configured just as you like it, all your graphics are cleanly converted for pdfTEX, and your document doesn't have any errors when you TEX it. We're almost to the holy grail.

All too often though, you're coming down the home stretch, and then Acrobat Reader offers something frightful: the typeface looks as if instead of the usual TEX fonts, the font `Horror` is being used.

If you didn't really choose `Horror` as your base font (thereby being quite happy with the typeface), then the cause is probably the following:

TEX normally uses the so-called packed or bitmap fonts (e.g., `cmr10.pk`), which cause Acrobat no small amount of difficulty for screen display. pdfTEX automatically avoids this issue, in that instead of bitmaps, it uses outline-based PostScript varieties of Computer Modern (fonts are the principal case where the PDF needs the PostScript construct). You can recognize this in the log file near the end of a pdfTEX run when you see messages like:
`...<cmr10.pfb><cmbx12.pfb>....`

To come back to the chewed-up screen representation of your document, the cause is usually in the preamble of your document, more specifically in the declaration

```
\usepackage[T1]{fontenc}
```

with which you want, among other things, to improve hyphenation[1], but instead of Computer Modern it embeds the EC fonts which are not included in PostScript form in most TEX installations, by default. A sure confirmation of this hypothesis is when pdfTEX writes to the log file messages like `<...ecrm1000.pk>`.

Considering the question of which is preferable, better hyphenation or beautiful screen appearance, the answer must of course be "both!" We have several possibilities to choose from:[2]

- Instead of the usual TEX fonts, use the standard PostScript fonts (Times, Helvetica, etc.), via `\usepackage{times}` for example. Possibly also use the `txfonts` (Times) or `pxfonts` (Palatino) to get companion symbols.
- If you (as I) are dependent on a typical TEX layout, invest $150 (including shipping) and get the European Modern Fonts from Y&Y.

[1] This is useful for continental hyphenation, but is not necessary for hyphenation in English.

[2] Two other notable possibilities have been developed since this article was originally written: The cm-super and Latin Modern PostScript font collections.

- If that's a bit too rich for you (like one of the frequently-occurring characters in this article), get the (free) `ae` package and include that in your document. With `ae` you get a modern font encoding, but the actual glyphs come from the Computer Modern PostScript fonts.
- Use the freely available Latin Modern or CM-super font collections now available from CTAN.

8 Additional PostScript fonts

Even if using the standard PostScript fonts is quite problem-free, including other, more exotic fonts can present a very labor-intensive task. This may be necessary since many firms and institutions, thanks to corporate design, insist on typefaces such as Frutiger, Futura, Garamond, etc.

Including such Type 1 fonts, for the most part commercially-sold, is certainly not trivial. However, as is well-known, with TEX almost nothing is impossible; consider the following:

- If you've purchased a PostScript font, its distribution should include files with the extensions `.pfb`, `.afm`, and/or `.pfm`.
- For use with pdfTEX you now still need the accompanying files with the extensions `.vf`, `.tfm`, `.map`, and `.fd`, and it is quite useful to have a style file to use the whole lot as a package in your document.

  As a rule, these files are *not* on the disk when you buy fonts! With a bit of luck, you might find them on CTAN in the `fonts/psfonts` directory.
- Look in the directory structure of your TEX installation, and try to find out where other font packages place files with the above-named extensions. Play around with putting the new font files in the same places.
- Look for the `pdftex.cfg` configuration file and add your `.map` file to it. For example, for a map file named `newfont.map`, you would add the line `map +newfont.map`.

If you did everything correctly, the new fonts should now be usable with pdfTEX. If you have difficulties, you might consult [2] and/or [3] for detailed information.

If you want to know a bit more about *what* you've just actually done, and what those files with the weird extension names are good for, I have a tip for you.

Come to the next meeting of Dante and ask Walter Schmidt, or better yet: come to one of Walter Schmidt's talks.

The only thing I have left now is the →

9 Conclusion

← that will close this overly-long article. If this introduction to pdfTEX still seems complicated, don't worry, just jump right in.

When creating new documents, using pdfTEX directly causes no problems in general, since you don't have to do anything extra aside from calling the `hyperref` package. Instead of the usual previewer or print driver, Acrobat Reader can be used without difficulty for both tasks.

It can be somewhat more complicated to repurpose existing TEX files for pdfTEX. Especially in cases where many (EPS) graphics need to be converted, it is frequently more efficient to choose the traditional way and use tools like `pstopdf` for PDF creation.

References

[1] D.P. Carlisle. *Packages in the graphics bundle*, Jan. 1999. `http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide%.ps`.

[2] Peter Flynn. "Installing PostScript fonts" in *Formatting Information*. `http://www.ctan.org/tex-archive/documentation/beginlatex/html/chapter8.%html#instfonts`.

[3] Philipp Lehman. The font installation guide. `http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide.p%df`.

[4] Heiko Oberdiek. PDF information and navigation elements with hyperref, pdfTEX and thumbpdf. In *EuroTEX'99 Proceedings*, 1999. `http://www.tug.org/applications/hyperref/ftp/doc/paper.pdf`.

[5] Sebastian Rahtz. *Hypertext marks in LATEX: the hyperref package*, Jun. 1998. `http://www.tug.org/applications/hyperref/`.

[6] Hàn Thế Thánh, Sebastian Rahtz, and Hans Hagen. *The pdfTEX user manual*, Jan. 2000. `http://www.tug.org/applications/pdftex/pdftex-a.pdf`.

◇ Thomas Feuerstack
FernUniversität in Hagen
58084 Hagen
Germany
`Thomas.Feuerstack@fernuni-hagen.de`