The pdfTeX 0.12+ user manual

Sebastian Rahtz

Hàn Thế Thành
thanh@informatics.muni.cz
s.rahtz@elsevier.co.uk

June 1998

1 Introduction

The main purpose of the pdfTeX project was to create an extension of TeX that can create PDF directly from TeX source files and improve/enhance the result of TeX typesetting with the help of PDF. pdfTeX contains TeX as a subset: when PDF output is not selected, pdfTeX produces normal DVI output, otherwise it produces PDF output that looks identical to the DVI output. The next stage of the project, apart from fixing any errors in the program, is to investigate alternative justification algorithms, possibly making use of multiple master fonts.

pdfTeX is based on the original TeX sources and web2c, and has been successfully compiled on Unix, Amiga, Win32 and DOS systems. It is still under beta development and all features are liable to change.

This manual was compiled by Sebastian Rahtz from notes and examples by Hàn Thế Thành. Many thanks are due to members of the pdfTeX mailing list (most notably Hans Hagen), whose questions and answers have contributed much to this manual.

2 Getting started

This section describes steps needed to get pdfTeX run on a system where pdfTeX is not installed. Some TeX packages have already contained pdfTeX as a part, such as MikTeX, Web2c for Win32 or teTeX, where we need not to bother with pdfTeX instalation. Note that installation description in this manual is web2c-specific.

2.1 Getting sources and binaries

Latest sources of pdfTeX are distributed together with precompiled binaries of pdfTeX for some platforms, including Linux[1], SGI IRIX, Sun SPARC Solaris and DOS

[1] The Linux binary is apparently compiled for the new libc-6 (aka GNU glibc-2.0), which will not run for users of older Linux installation still based on libc-5

(DJGPP)[2]. The primary location is ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/pdftex-testing/latest

For Win32 systems (Windows 95, Windows NT) there are two packages that contain pdfTEX, both in CTAN:systems/win32. Web2c for Win32 is maintained by Fabrice Popineau (mailto:popineau@ese-metz.fr), and MikTeX by Christian Schenk (mailto:cschenk@berlin.snafu.de).

A binary version of pdfTEX (both 0.12f and 0.12h) for the Amiga is coming with the AmiWeb2c distribution (CTAN:systems/amiga/amiweb2c/) by Andreas Scherer (mailto:andreas.scherer@pobox.com).

2.2 Compiling

If there is no precompiled binary of pdfTEX for our system, we need to build pdfTEX from sources. The compilation is expected to be easy on UNIX-like systems and can be described best by example. Assuming that all needed files are downloaded to `$HOME/pdftex`, the following steps are needed to compile pdfTEX (based on web2c-7.2) on a UNIX system:

```
cd $HOME/pdftex
gunzip < web-7.2.tar.gz | tar xvf -
gunzip < web2c-7.2.tar.gz | tar xvf -
gunzip < pdftex.tar.gz | tar xvf -
mv pdftexdir web2c-7.2/web2c
cd ./web2c-7.2
./configure
cd ./web2c
make pdftex
```

If we happen to have a previously configured source tree and just install a new version of pdfTEX, we can avoid running configure from the top-level directory. It's quicker to run config.status, which will just regenerate the Makefiles based on config.cache:

```
cd web2c-7.2/web2c
sh config.status
make pdftex
```

Apart from the binary of pdfTEX the compilation also produces several other files which are needed for running pdfTEX:

`pdftex.pool` – so-called pool file, needed for creating formats, located in `web2c-7.2/web2c`

`texmf.cnf` – web2c run-time configuration file, located in `web2c-7.2/kpathsea`

`ttf2afm` – an external program to generate AFM file from TrueType fonts, located in `web2c-7.2/web2c/pdftexdir`

[2]The DJGPP version is built by DJGPP cross-compiler on Linux

For precompiled binaries these files are included in the zip archive containing the binary (`pdftex.zip`).

2.3 Getting pdfTEX-specific platform-indenpendent files

Apart from above-mentioned files, there is another zip archive (`pdftexlib-0.12.zip`) in pdfTEX distribution which contains platform-indenpendent files required for running pdftex: pdftex config file (`pdftex.cfg`), encoding vectors (`*.enc`), map files (`*.map`), macros (`*.tex`). Unpacking this archive (don't forget `-d` option when using pkunzip) will create a `texmf` tree containing pdfTEX-specific files.

2.4 Placing files

The next step is to place the binaries somewhere in `PATH`. We also need to make a copy (or symbolic link) of `pdftex` and name it as `pdflatex` if we want to use LATEX. `texmf.cnf`, `pdftex.pool` and directory `texmf/` (created by unpacking pdftexlib-0.12.zip) should be also moved to "appropriate" place (see below).

2.5 Setting search paths

web2c-based programs, including pdfTEX, use a `web2c` run-time configuration file called `texmf.cnf`. This file can be found via the user-set environment variable `TEXMFCNF` or via the compile-time default value if the former is not set. It is strongly recommended to use the first way. Then we need to edit `texmf.cnf` so pdfTEX can find all necessary files. Usually one has to edit `TEXMFS` and maybe some next. When runing pdfTEX, some extra search paths are used beyond those normally requested by TEX itself:

`VFFONTS` — the path where pdfTEX looks for virtual fonts.

`T1FONTS` — the path where pdfTEX looks for Type1 fonts.

`TTFONTS` — the path where pdfTEX looks for TrueType fonts.

`PKFONTS` — the path where pdfTEX looks for PK fonts.

`TEXPSHEADERS` — the path where pdfTEX looks for the configuration file `pdftex.cfg`, font mapping files (`*.map`), encoding files (`*.enc`), and PNG pictures.

2.6 The pdfTEX configuration file

When pdfTEX starts, apart from `web2c` configuration file it reads a *pdfTEX configuration file* called `pdftex.cfg`, searched for in the `TEXPSHEADERS` path. As web2c systems commonly specify a 'private' tree for pdfTEX where configuration and map files are located, this allows individual users or projects to maintain customized versions of the configuration file, and means that specific TEX input files need not set any pdfTEX-specific macros.

The configuration file is used to set default values for the following parameters (all of which can be over-ridden in the TEX source file):

output_format Integer parameter specifying whether the output format should be DVI or PDF. Positive value means PDF output, otherwise DVI output.

compress_level Integer parameter specifying the level of text compression via `zlib`. Zero means no compression, 1 means fastest, 9 means best, 2..8 means something in between.

decimal_digits Integer parameter specifying the preciseness of of real numbers in PDF page description. It gives the maximal number of decimal digits after the decimal point of real numbers. Valid values are in range 0..5. Higher value means more precise output, but also much larger size and more time to display or print. In most cases the optimal value is 2.

page_width, page_height Dimension parameters specifying the page width and page height of PDF output. If not specified then page width is calculated as **width of the box being shiped out** $+ 2 \times$ (*horigin* $+$ `\hoffset`). The page height is similar.

horigin, vorigin Dimension parameters specifying the offset of the TEX output box from the top left corner of the 'paper'

map The name of the font mapping file (similar to those used by many DVI to PostScript drivers); more than one map file can be specified, using multiple `map` lines. If the name of the map file is prefixed with a `+`, its values are appended to the existing set, otherwise they replace it. If no map files are given, the default value `psfonts.map` is used.

A typical `pdftex.cfg` file looks like this, setting up output for A4 paper size and the standard TEX offset of 1 inch, and loading two map files for fonts:

```
output_format 1
compress_level 0
decimal_digits 2
page_width 210mm
page_height 297mm
horigin 1in
vorigin 1in
map standard.map
map +cm.map
```

2.7 Creating formats

Formats for pdfTEX are created in the same way as for TEX. For plain TEX and LATEX it looks like:

```
pdftex -ini -fmt=pdftex plain \\dump
pdftex -ini -fmt=pdflatex latex.ltx
```

The formats (`pdftex`, `pdflatex` or other formats) should be placed in the `TEXFORMATS` path.

2.8 Testing the installation

Now we can test the installation:

```
pdftex example
```

If the installation is ok, this run should produce a file called `example.pdf`. The file `example.tex` is also a good place to look how pdfTEX new primitives are used.

2.9 Common problems

The most common problem with installation is pdfTEX complains that something cannot be found. In such cases make sure that `TEXMFCNF` is set correctly, so pdfTEX can find `texmf.cnf`. The next best place to look/edit is the file `texmf.cnf`. Also setting `KPATHSEA_DEBUG=255` before running pdfTEX or running pdfTEX with option `-k 255` will cause pdfTEX writes a lot of debugging information, which may be useful to find out the troubles. Variables in `texmf.cnf` can be overwritten by enviroment variables. Here are some most common problems with getting started:

- `"! I can't read tex.pool; bad path?"`

  `TEXMFCNF` is not set correctly so pdfTEX cannot find `texmf.cnf`, or `TEXPOOL` (in `texmf.cnf`) doesn't contain path to pool file (`pdftex.pool`).

- `"! You have to increase POOLSIZE."`

  pdfTEX cannot find `texmf.cnf`, or the value of `pool_size` specified in `texmf.cnf` is not large enough and must be increased. If `pool_size` is not specified in `texmf.cnf` then we can add something like

  ```
  "pool_size        =  500000"
  ```

- `"I can't find the format file ‘pdftex.fmt’!"`

  or

  `"I can't find the format file ‘pdflatex.fmt’!"`

  Format is not created (see above how to do that) or is not properly placed. Make sure that `TEXFORMATS` in `texmf.cnf` contains path to `pdftex.fmt` or `pdflatex.fmt`.

- `"! Fatal format file error; I'm stymied."`

This appears if we forgot to regenerate the `.fmt` files after installing a new version of the pdfTeX binary and `pdftex.pool`.

- `"! TEX.POOL doesn't match; TANGLE me again!"`

  or

  `"! tex.pool doesn't match; tangle me again (or fix the path)."`

  This might appear if you forgot to install the proper `pdftex.pool` when installing a new version of the pdfTeX binary.

- pdfTeX cannot find: config file (`pdftex.cfg`), map files (`*.map`), encoding vectors (`*.enc`), virtual fonts, Type1 fonts, TrueType fonts or image files.

  Make sure that the required file exists and the correspoding variable in `texmf.cnf` contains path to the file. See above which variables pdfTeX needs apart from the ones TeX uses.

3 Setting up fonts

pdfTeX can work with Type 1 and TrueType fonts, and a source must be available for all fonts used in the document, except for the 14 base fonts supplied by Acrobat Reader (Times, Helvetica, Courier, Symbol and Dingbats). It is possible to use METAFONT-generated fonts in pdfTeX—but it is strongly recommended not to use METAFONT-fonts if an equivalent is available in Type 1 or TrueType format, as the resulting Type 3 fonts render very poorly in Acrobat Reader. Given the free availability of Type 1 versions of all the Computer Modern fonts, and the ability to use standard PostScript fonts without further ado, most existing TeX users should be able to experiment with pdfTeX.

3.1 Map files

pdfTeX reads *map files* (specified in the *configuration file* (see above, section 2.6), in which reencoding and partial downloading for each font are specified. Every font needed must be listed, each on a separate line, apart from PK fonts. The syntax of each line is similiar to dvips map files[3] (but may be changed later), and can contain up to the following (some are optional) fields: *texname*, *basename*, *fontflags*, *fontfile*, *encodingfile* and *special*. The only mandatory is *texname* and must be the first field. The rest is optional, but if *basename* is given, it must be the second field. Similarly if *fontflags* is given it must be the third field (if *basename* is present) or the second field (if *basename* is left out). It is possible to mix the positions of *fontfile*, *encodingfile* and *special*, however the first three fields must be given in fixed order.

texname: the name of the TFM file. This must be given for each line.

[3] dvips map files can be used with pdfTeX without problems.

basename: the base font name (PostScript font name). If not given then it will be taken from font file. Specifying name that doesn't match the name in the font file will cause pdfTEX write a warning message about that, so it would be best not to have this field specified if the font resouce is availble (which is the most common case). This option is primarly intended for use of base fonts and for compability with dvips map files.

fontflags: the flags specifying some chracteristics of the font. Its description has been taken (with a slightly modification) from PDF Manual, 7.9.2 Font descriptor flags.

> The value of the Flags key in a font descriptor is a 32-bit integer that contains a collection of Boolean attributes. These attributes are true if the corresponding bit is set to 1 in the integer. The following specifies the meanings of the bits, with bit 1 being the least significant. Reserved bits must be set to zero.
>
> | Bit position | Semantics |
> |---|---|
> | 1 | Fixed-width font |
> | 2 | Serif font |
> | 3 | Symbolic font |
> | 4 | Script font |
> | 5 | Reserved |
> | 6 | Uses the Adobe Standard Roman Character Set |
> | 7 | Italic |
> | 8–16 | Reserved |
> | 17 | All-cap font |
> | 18 | Small-cap font |
> | 19 | Force bold at small text sizes |
> | 20–32 | Reserved |
>
> All characters in a fixed-width font have the same width, while characters in a proportional font have different widths. Characters in a serif font have short strokes drawn at an angle on the top and bottom of character stems, while sans serif fonts do not have such strokes. A symbolic font contains symbols rather than letters and numbers. Characters in a script font resemble cursive handwriting. An all-cap font, which is typically used for display purposes such as titles or headlines, contains no lowercase letters. It differs from a small-cap font in that characters in the latter, while also capital letters, have been sized and their proportions adjusted so that they have the same size and stroke weight as lowercase characters in the same typeface family.
>
> Bit 6 in the flags field indicates that the font's character set is the Adobe Standard Roman Character Set, or a subset of that, and that it uses the standard names for those characters.

Finally, bit 19 is used to determine whether or not bold characters are drawn with extra pixels even at very small text sizes. Typically, when characters are drawn at small sizes on very low resolution devices such as display screens, features of bold characters may appear only one pixel wide. Because this is the minimum feature width on a pixel-based device, ordinary non-bold characters also appear with one-pixel wide features, and cannot be distinguished from bold characters. If bit 19 is set, features of bold characters may be thickened at small text sizes.

If the font flags is not given, pdfTeX treates it as 4, which stands for Symbolic font. If we do not know the correct value, it would be best not to have it given, as specifying bad value of font flags may cause Acrobat Reader some troubles. On the other hand this option is not absolutely useless—apart from back compability with pdfTeX old map files we may need to give this number in some cases (see the *fontfile* description below).

fontfile: the name of the font source file. This must be a Type 1 or TrueType font file. The font file name can be preceded by one or two special characters, which says how the font file should be handled.

- If it is preceded by a < then the font file will be partly downloaded, which means that only used glyphs (characters) are embedded to the font. This is the most common use and is *strongly recommended* for any font, as it ensures the portability and reduces the size of the PDF output.
- In case the font file name is preceded by a double <<, the whole font file will be included entirely—all glyphs of the font are embedded, including the ones that re not used in the document. Apart from causing large size of PDF output, this option may cause troubles with TrueType fonts too, so it is not recommended. It might be useful in case the font is untypical and can not be subsetted well by pdfTeX.
- In case nothing preceded the font file name, the font file is read but nothing is embedded, only the font parameters are extracted to generated so-called font descriptor, which is used by Acrobat Reader to simulate the font if needed. This option is useful only when we do not want to embed the font (i.e. to reduce the output size), but wish to use the the font metrics and let Acrobat Reader generate instance that look closed to the used font in case the font resource is not installed on system where the PDF output will be viewed or printed. To use this feature the font flags *must* be specify, and it must have the bit 6 set on, which means that only fonts with Adobe Standard Roman Character Set can be simulated. The only exception is case of Symbolic font, which is not very useful.
- If the font file name is preceded by a !, the font is not read at all, and is assumed to be available on system. This option can be used to create PDF files which do not contain embedded fonts. The PDF output then works only on systems where resource of the used font is available. It's not useful

very much for document exchange, as the PDF is not "portable" at all. On the other hand it is very useful when we wish to speed up running of pdfTEX during interactive work, and in final version we embed all used fonts. This feature requires Acrobat Reader to has access to installed fonts on system. This has been tested on Win95 and UNIX (Solaris).

Note that the standard 14 fonts are never downloaded, even they are specified to be downloaded in map files.

encoding: name of the file containing the external encoding vector to be used for the font. The file name may be preceded by a <, but the effect is the same. The format of the encoding vector is identical to that used by dvips. If no encoding is specified, the font's built-in default encoding is used. It may be omitted if we are sure that the font resource has the correct built-in encoding. In general this option is highly prefered, and is *required* to subset TrueType font.

special: special instruction for font handling as for dvips. Only specification of `SlantFont` and `ExtendFont` is read, other instructions are just skipped.

If a used font is not present in map files, pdfTEX will look first for a source with suffix `.pgc`, which is so-called *PGC* source (PDF Glyph Container)[4]. If no PGC source is available, pdfTEX will try to use PK fonts in a normal way as DVI drivers do, including on-the-fly creating PK fonts if needed.

Lines containing nothing apart from *texname* stand for scalable Type 3 fonts. For scaleable fonts as Type 1, TrueType and scaleable Type 3 font, all the fonts loaded from a TFM at various size will be treated as an only font in the PDF output. Thus if a font let's say csr10 is present in map files, then it will be treated as scaleable. As the result the font source for csr10 will be downloaded only one for `csr10`, `csr10 at 12pt` etc. This is to avoid multiple downloading identical font sources. Thus vector PGC fonts should be specified as scaleable Type 3 in map files like:

```
csr10
```

It doesn't hurt much if a scaleable Type 3 font is not given in map files, except that the font source will be downloaded multiple for various sizes, which causes the PDF output larger. On the other hand if a font is given in map files as scaleable Type 3 font and its PGC source is not scaleable or not available (in this case pdfTEX will use PK font instead), the PDF output is still valid but some fonts may look ugly because of scaling bitmap.

Here are some sample lines:

Using a built-in font with font-specific encoding, i.e. neither a download font nor an external encoding is given. SlantFont is specified similiarly as for dvips.

[4]This is a text file containing PDF Type 3 font, often created some MetaPost with some utilities by Hans Hagen. In general PGC files can contain whatever allowed in PDF page description, which may be used to support fonts that are not available in METAFONT. At the moment PGC fonts are not very uselful, as vector Type 3 fonts are not displayed very well in Acrobat Reader, but it may be more useful when Type 3 font handling gets better.

```
psyr        Symbol
psyro       Symbol          ".167 SlantFont"
pzdr        ZapfDingbats
```

Using a built-in font with an external encoding (`8r.enc`). The `<` preceded encoding file may be left out.

```
ptmr8r      Times-Roman     <8r.enc
ptmri8r     Times-Italic    <8r.enc
ptmro8r     Times-Roman     <8r.enc   ".167 SlantFont"
```

Using a partially downloaded font with an external encoding:

```
putr8r      Utopia-Regular  <8r.enc <putr8a.pfb
putri8r     Utopia-Italic   <8r.enc <putri8a.pfb
putro8r     Utopia-Regular  <8r.enc <putr8a.pfb     ".167 SlantFont"
```

Using some faked font map entries:

```
logo8      <logo8.pfb
logo9      <logo9.pfb
logo10     <logo10.pfb
logosl8    <logo8.pfb      ".25 SlantFont"
logosl9    <logo9.pfb      ".25 SlantFont"
logosl10   <logosl10.pfb
logobf10   <logobf10.pfb
```

Width adjusted, but not the stroke thickness

```
logod10         logobf10        <logobf10.pfb   ".913 ExtendFont"
```

Will work for ASCII-subset of OT1 and T1:

```
ectt1000        cmtt10          <cmtt10.map     <tex256.enc
```

Using entirely downloaded font without reencoding:

```
pgsr8r GillSans <<pgsr8a.pfb
```

Using partially downloaded font without reencoding:

```
pgsr8r GillSans <pgsr8a.pfb
```

Do not read the font at all—the font is supposed to be installed on the system:

```
pgsr8r GillSans !pgsr8a.pfb
```

Using entirely downloaded font with reencoding:

```
pgsr8r GillSans <<pgsr8a.pfb 8r.enc
```

Using partially downloaded font with reencoding:

```
pgsr8r GillSans <pgsr8a.pfb 8r.enc
```

Do not include font but extract parameters from font file and reencode—only works for font with Adobe Standard Encoding. The font flags says how this font looks like so Acrobat Reader can generate similiar instance if the font resource is not available on the target system.

```
pgsr8r GillSans 32 pgsr8a.pfb 8r.enc
```

A TrueType font can be used in the same way as a Type 1 font:

```
verdana8r Verdana  <verdana.ttf 8r.enc
```

3.2 TrueType fonts

As mentioned above, pdfTEX can work with TrueType fonts. Adding TrueType into map files is similar to Type 1 font. The only extra thing to do with TrueType is to create TFM. There is a program ttf2afm in pdfTEX distribution which can be used to extract AFM from TrueType fonts. Usage is simple:

```
ttf2afm <ttf> [<encoding>]
```

ttf is the TrueType font file, the optinal *encoding* specifies the encoding, which is the same as encoding vector used in map files for pdfTEX and dvips. If the encoding is not given, all the glyphs of the AFM output will be mapped to `/.notdef`. ttf2afm writes the output AFM to standard output. The rest is easy, as we have AFM already. If we need to know which glyphs are available in the font, we can run ttf2afm without encoding to get all glyph names.

To use a new TrueType font the minimal steps may look lile below (suppose that `test.map` is included in `pdftex.cfg`):

```
ttf2afm times.ttf 8r.enc >times.afm
afm2tfm times.afm -T 8r.enc
echo "times TimesNewRomanPSMT <times.ttf <8r.enc" >>test.map
```

The PostScript font name (`TimesNewRomanPSMT`) is reported by afm2tfm, but from version 0.12l it may be left out.

ExtendFont and SlantFont also work for TrueType fonts.

4 New primitives

There follows a short description of new primitives added by pdfTEX. One way to learn more about how to use these primitives is to have a look at the file `example.tex` in the pdfTEX distribution.

4.1 Document setup

`\pdfoutput=`*n*

Integer parameter specifying whether the output format should be DVI or PDF. Positive value means PDF output, otherwise DVI output. This parameter cannot be specified *after* shipping out the first page. In other words, this parameter must be set before pdfTEX ships out the first page if we want PDF output. This is the only one parameter that must be set to produce PDF output. All others are optional.

`\pdfcompresslevel=`*n*

Integer parameter specifying the level of text compression via `zlib`. Zero means no compression, 1 means fastest, 9 means best, 2..8 means something in between. A value out of this range will be adjusted to the nearest meaningfull value.

`\pdfpagewidth=`*dimen*

`\pdfpageheight=`*dimen*

Dimension parameters specifying the page width and page height of PDF output. If not given then the page dimensions will be calculated as mentioned above.

`\pdfpagesattr{`*text*`}`

Token list parameter specifying optional attributes common for all pages of PDF output file. These attributes can be `MediaBox` (rectangle specifying the natural size of the page), `CropBox` (rectangle specifying the region of the page being displayed and printed), `Rotate` (number of degrees the page should be rotated clockwise when it is displayed or printed; must be 0 or a multiple of 90).

`\pdfpageattr{`*text*`}`

this is similiar to `\pdfpageattr`, but it takes priority to the former one. It can be used to overwrite any attributes given by `\pdfpagesattr` for individual pages.

4.2 The document info and catalog

`\pdfinfo{`*info keys*`}`

This allows the user to add information to the document info section; if this is provided, it can be seen in Acrobat Reader with the menu option Document Information, General. The *info keys* parameter is a set of data pairs, a key and a value. The key names are preceded by a `/`, and the values are in parentheses; all keys are optional. The possible keys are `/Author`, `/CreationDate` (defaults to current date), `/ModDate`, `/Creator` (defaults to `TeX`), `/Producer` (defaults to `pdfTeX`), `/Title`, `/Subject`, and `/Keywords`.

`/CreationDate` and `/ModDate` are expressed in the form `D:YYYYMMDDhhmmss)`, where YYYY is the year, MM is the month, DD is the day, hh is the hour, mm is the minutes, and ss is the seconds.

Multiple apperances of `\pdfinfo` will be concatenated to the only one. If a key is given more than once, then the first appearance will take priority. An example of use of `\pdfinfo` may look like:

```
\pdfinfo{
    /Title (example.pdf)
    /Creator (TeX)
    /Producer (pdfTeX)
    /Author (Tom and Jerry)
    /CreationDate (D:19980212201000)
    /ModDate (D:19980212201000)
    /Subject (Example)
    /Keywords (pdfTeX)
}
```

`\pdfcatalog{`*catalog keys*`}` *openaction goto* ⟨*num num* | *name*`{`*name*`}`⟩ *appearance*

Similar to the document info section is the document catalog, where the available keys are `/URI`, which provides the base URL of the document, and */PageMode* determines how Acrobat displays the document on startup. The possibilities for the latter are:

`/UseNone`	Open document with neither outline nor thumbnails visible.
`/UseOutlines`	Open document with outline visible.
`/UseThumbs`	Open document with thumbnails visible.
`/FullScreen`	Open document in full-screen mode. In full-screen mode, there is no menu bar, window controls, nor any other window present.

The default is `/UseNone`.

The *openaction* is the action provided when opening the document and is specified in the same way as for internal links (see below, section 4.6), e.g. `goto page 3 {/Fit}`.

4.3 Graphics inclusion

`\pdfimage` *width width height height depth depth filename*

Insert an image, optionally changing width, height, depth or any combination of them. Default values are zero for depth and "running" for height and width. If all of them are given, the image will be scaled to fit the specified values. If some of them (but not all) are given, the rest will be set to a value corresponding to the remaining ones so as to make the image size to yield the same proportion of $width : (height + depth)$ as the natural image size, where depth is treated as zero. If non of them is given then the image will take the natural size of it. An image inserted at natural size often has resolution 72 DPI in output file, but some images may contain data specifying image resolution, and in such a case the image will be scaled to the intended resolution. The filename of the image must appear after the optional dimension parameters. The dimension of the image can be accessed by enclosing the `\pdfimage` command to a box and checking the dimensions of the box.

The image type is specified by the extension of the given file name. `.png` stands for PNG image, `.pdf` for PDF file, otherwise the image is treated as JPEG.

4.4 XObject Forms

`\pdfform` *num*

Write out the TEX box *num* as a XObject Form to the PDF file.

`\pdflastform`

Returns the object number of the last XObject Form written to the PDF file

`\pdfrefform \name`

Put in a reference to the XObject Form called `\name`.

These macros support a feature called “object reuse” in pdfTEX. The idea is to create a Form object in PDF. The content of the XObject Form object corresponds to the content of a TEX box, which can also contain pictures and references to other XObject Form objects as well. After that the XObject Form can be used by simply referring to its object number. This feature can be useful for large document with a lot of similiar elements, as it can reduce the duplication of identical objects.

4.5 Annotations

`\pdfannot` *`width`* *width* *`height`* *height* *`depth`* *depth* {*text*}

Attach an annotation at the current point in the text. The text is inserted as raw PDF code to the contents of annotation.

`\pdflastannot`

Returns the object number of last annotation created by `\pdfannot`. These two primitives allow user to create any annotation that cannot be created by `\pdfannotlink` (see below).

4.6 Destinations and links

`\pdfdest` ⟨ *num num* | *name*{*name*} ⟩ *appearance*

Establish a destination for links and bookmark outlines; the link must be identified by either a number or a symbolic name, and the way Acrobat is to display the page must be specified; *appearance* must be one of

`fit`	fit whole page in window
`fith`	fit whole width of page
`fitv`	fit whole height of page
`fitb`	fit whole 'Bounding Box' page
`fitbh`	fit whole width of 'Bounding Box' of page
`fitbv`	fit whole height of 'Bounding Box; of page
`xyz`	keep current zoom factor

`xyz` can optionally be followed by `zoom` *`factor`* to provide a fixed zoom-in. The *factor* is like TEX magnification, ie 1000 is the 'normal' page view.

`\pdfannotlink` *`height`* *height* *`depth`* *depth* *`attr`*{*attr*} *action*

Start a hypertext link; if the optional dimensions are not specified, they will be calculated from the box containing the link. The *attributes* (explained in great detail in section 6.6 of the PDF manual) determine the appearance of the link. Typically, this is used to specify the color and thickness of any border around the link. Thus `/C [0.9 0 0] /Border [0 0 2]` specifies a color (in RGB) of dark red, and a border thickness of 2 points.

The *action* can do many things; some possibilities are

page *n*	Jump to page *n*
goto num *n*	
goto name {*refname*}	Jump to a point established as *name* with `\pdfdest`
goto file {*filename*}	Open a local file; this can be used *with* a *name* or *num* specification, to point to a specific location on the file. Thus `goto file{foo.pdf} name{intro}`
thread num {*n*}	
thread name {*refname*}	Jump to thread identified by *n* or *refname*
user {*spec*}	Perform user-specified action. Section 6.9 of the PDF manual explains the possibilities. A typical use of this is to specify a URL, e.g. `/S /URI /URI (http://www.tug.org/)`.

`\pdfendlink`

Ends link; all text between `\pdfannotlink` and `\pdfendlink` will be treated as part of this link. pdfTEX may break the result across lines (or pages), in which case it will make several links with the same content.

4.7 Bookmarks

`\pdfoutline` *action* `count` *count* {*text*}

Create a outline (or bookmark) entry. The first parameter specificies the action to be taken, and is the same as that allowed for `\pdfannotlink`. The *count* specifies the number of direct subentries under this entry, 0 if this entry has no subentries (in this case it may be omitted). If the number is negative, then all subentries will be closed and the absolute value of this number specifies the number of subentries. The *text* is what will be shown in the outline window (note that this is limited to characters in the PDFEncoding vector).

4.8 Article threads

`\pdfthread` *num* *num* *name*{*name*}

Start an article thread; the corresponding `\pdfendthread` must be in the box in the same depth as the box containing `\pdfthread`. All boxes in this depth level will be treated as part of this thread. An identifier (*n* or *refname*) must be specified; threads with same identifier will be joined together.

`\pdfendthread`

Finish the current thread.

`\pdfthreadhoffset`=*dimen*

`\pdfthreadvoffset`=*dimen*

Specify thread margins.

4.9 Miscellaneous

`\pdfliteral`{*pdf text*}

Like `\special` in normal TEX, this command inserts raw PDF code into the output. This allows support of color, and text transformation.

`\pdfobj` *stream* {*text*}

Similar to `\pdfliteral`, but the text is inserted as contents of an object. If the optional keyword *stream* is given then the contents will be inserted as a stream.

`\pdflastobj`

Returns the object number of the last object created by `\pdfobj`. These primitives provide a mechanism allowing inserting a user-defined object to PDF output.

`\pdftexversion`

Returns the version of pdfTeX multiple by 100, e.g. for version `0.12x` it returns 12.

`\pdftexrevision`

Returns the revision of pdfTeX, e.g. for version `0.12x` it returns `x`.

5 Graphics and color

pdfTeX supports inclusion of pictures in PNG, JPEG and PDF format. The most common technique — the inclusion of Encapsulated PostScript figures, is replaced by PDF inclusion. EPS files can be converted to PDF by GhostScript, Acrobat Distiller or other PS-to-PDF convertors. The BoundingBox of PDF file is taken from CropBox if available, otherwise from MediaBox. To get the right BoundingBox from EPS file, before converting to PDF it is neccesarry to transform the EPS file so that the start point is at the (0,0) coordinate and the page size is set exactly corresponding to the BoundingBox. A Perl script (fitps) for this purpose has been written by Sebastian Rahtz.

Other alternatives for graphics in pdfTeX are:

1. LaTeX picture mode: since this is implemented simply in terms of font characters, it works in exactly the same way as usual;
2. Xy-pic: If the PostScript backend is not requested, Xy-pic uses its own Type 1 fonts, and needs no special attention;
3. tpic: The 'tpic' `\special` commands (used in some macro packages) can be redefined to produce literal PDF, using macros by Hans Hagen;
4. MetaPost: although the output of MetaPost is PostScript, it is in a highly simplified form, and a MetaPost to PDF conversion (written by Hans Hagen and Tanmoy Bhattacharya) is implemented as a set of macros which read MetaPost output and support all of its features;

5. It is possible to insert a “pure” PDF file (PDF that has only one page without fonts, bitmaps or other resources) using a macro package that reads the external PDF file line by line.

The two latter macro packages are part of ConTeXt (`supp-pdf.tex` and `supp-mis.tex`), but also work with LaTeX and are distributed separately.

For new work, the MetaPost route is highly recommended. For the future, Adobe have announced that they will define a specification for ‘encapsulated PDF’, and this should solve some of the present difficulties.

The inclusion of raw PostScript commands (the techique utilized by the pstricks package) cannot be supported. Although PDF is a direct descendant of PostScript, it lacks any programming language commands, and cannot deal with arbitrary PostScript.

6 Macro packages supporting pdfTeX

- For LaTeX users, Sebastian Rahtz’ hyperref package has substantial support for pdfTeX, and provides access to most of its features. In the simplest case, the user merely needs to load hyperref with a ‘pdftex’ option, and all cross-references will be converted to PDF hypertext links. PDF output is automatically selected, text compression turned on, and the page size is set up correctly. Bookmarks are created to match the table of contents.

- The standard LaTeX graphics and color packages have `pdftex` options, which allow use of normal color, text rotation, and graphics inclusion commands. Only PNG and MetaPost files can be included.

- The ConTeXt macro package by Hans Hagen (`mailto:pragma@pi.net`) has very full support for pdfTeX in its generalized hypertext features.

- Hypertexted PDF from `texinfo` documents can be created with `pdftex-info.tex`, which is a slight modification of the standard `texinfo` macros. This is part of the pdfTeX distribution.

- A similiar modification of the `webmac`, called `pdfwebmac.tex`, allows production of hypertexted PDF versions of program written in WEB. This is part of the pdfTeX distribution.

Some nice samples of pdfTeX output can be found on the TUG Web server, at `http://www.tug.org/applications/pdftex/`.