What's new in the pdfTEX world?

Martin Schröder

pdfTEX Team

BachoTEX 2008, 30th April – 4nd May 2008, Bachotek

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

Looking back: pdfTEX 1.40

- PDF 1.5 compressed object streams
- smaller PDFs
- PDF-X/3
- A new graphic format: JBIG2
- Colour stacks
- Draft mode
- Shell escape
- Better support for unicode fonts
- Highly improved font expansion in autoexpand mode

pdfTEX 1.40.0 → pdfTEX 1.40.8

- Font flags
- LFS
- Page groups and transparent pngs
- Various bug fixes

pdfTEX 1.40.0 → pdfTEX 1.40.8

- Font flags
- LFS
- Page groups and transparent pngs
- Various bug fixes

pdfTeX 1.40.0 → pdfTeX 1.40.8

- Font flags
- LFS
- Page groups and transparent pngs
- Various bug fixes

pdfTEX 1.40.0 → pdfTEX 1.40.8

- Font flags
- LFS
- Page groups and transparent pngs
- Various bug fixes

Font flags

- Font flags is a bitset that controls the replacement of non-embedded fonts, telling the renderer certain properties of the font (serif/sans serif, symbol, italic, script, fixed, . . .)
- Font flags can be set in the map lines, but nobody does that
- pdfTEX tried to use a (mostly wrong) default, which you couldn't set if you wanted to
- Now pdfTEX uses the flags from map entries (if found) or guesses values for the base14 fonts. Warnings about missing font flags have been disabled

Font flags

- Font flags is a bitset that controls the replacement of non-embedded fonts, telling the renderer certain properties of the font (serif/sans serif, symbol, italic, script, fixed, . . .)
- Font flags can be set in the map lines, but nobody does that
- pdfTEX tried to use a (mostly wrong) default, which you couldn't set if you wanted to
- Now pdfTEX uses the flags from map entries (if found) or guesses values for the base14 fonts. Warnings about missing font flags have been disabled

Font flags

- Font flags is a bitset that controls the replacement of non-embedded fonts, telling the renderer certain properties of the font (serif/sans serif, symbol, italic, script, fixed, . . .)
- Font flags can be set in the map lines, but nobody does that
- pdfTEX tried to use a (mostly wrong) default, which you couldn't set if you wanted to
- Now pdfTEX uses the flags from map entries (if found) or guesses values for the base14 fonts. Warnings about missing font flags have been disabled

Font flags

- Font flags is a bitset that controls the replacement of non-embedded fonts, telling the renderer certain properties of the font (serif/sans serif, symbol, italic, script, fixed, . . .)
- Font flags can be set in the map lines, but nobody does that
- pdfTEX tried to use a (mostly wrong) default, which you couldn't set if you wanted to
- Now pdfTEX uses the flags from map entries (if found) or guesses values for the base14 fonts. Warnings about missing font flags have been disabled

LFS: Background

- Old operating or file systems limit the file size to 2 GiB (31 bit)
- A unix standard for larger files exists since 1996 (when 64 bit machines came into use): you normally need only some compiler flags, have to use `off_t` instead of `int` and you have to change some function calls
- PDFs $>$ 2 GiB really exist. pdfTEX created them on 64 bit platforms seemingly without any problems, but because it used only `int` instead of `off_t` these were broken
- Normally PDFs are limited (up to PDF 1.4) to $<$10 GB
- Since PDF1.5 one can use xref streams (`\pdfobjcompresslevel=1`) and create files with an unlimited size

LFS: Background

- Old operating or file systems limit the file size to 2 GiB (31 bit)
- A unix standard for larger files exists since 1996 (when 64 bit machines came into use): you normally need only some compiler flags, have to use `off_t` instead of `int` and you have to change some function calls
- PDFs $>$ 2 GiB really exist. pdfTEX created them on 64 bit platforms seemingly without any problems, but because it used only `int` instead of `off_t` these were broken
- Normally PDFs are limited (up to PDF 1.4) to $<$10 GB
- Since PDF1.5 one can use xref streams (`\pdfobjcompresslevel=1`) and create files with an unlimited size

LFS: Background

- Old operating or file systems limit the file size to 2 GiB (31 bit)
- A unix standard for larger files exists since 1996 (when 64 bit machines came into use): you normally need only some compiler flags, have to use `off_t` instead of `int` and you have to change some function calls
- PDFs $>$ 2 GiB really exist. pdfTEX created them on 64 bit platforms seemingly without any problems, but because it used only `int` instead of `off_t` these were broken
- Normally PDFs are limited (up to PDF 1.4) to $<$10 GB
- Since PDF1.5 one can use xref streams (`\pdfobjcompresslevel=1`) and create files with an unlimited size

LFS: Background

- Old operating or file systems limit the file size to 2 GiB (31 bit)
- A unix standard for larger files exists since 1996 (when 64 bit machines came into use): you normally need only some compiler flags, have to use `off_t` instead of `int` and you have to change some function calls
- PDFs $>$ 2 GiB really exist. pdfTEX created them on 64 bit platforms seemingly without any problems, but because it used only `int` instead of `off_t` these were broken
- Normally PDFs are limited (up to PDF 1.4) to $<$10 GB
- Since PDF1.5 one can use xref streams (`\pdfobjcompresslevel=1`) and create files with an unlimited size

LFS: Background

- Old operating or file systems limit the file size to 2 GiB (31 bit)
- A unix standard for larger files exists since 1996 (when 64 bit machines came into use): you normally need only some compiler flags, have to use `off_t` instead of `int` and you have to change some function calls
- PDFs $>$ 2 GiB really exist. pdfTEX created them on 64 bit platforms seemingly without any problems, but because it used only `int` instead of `off_t` these were broken
- Normally PDFs are limited (up to PDF 1.4) to $<$10 GB
- Since PDF1.5 one can use xref streams (`\pdfobjcompresslevel=1`) and create files with an unlimited size

LFS: 1.40.7

- pdfTEX 1.40.7 creates (with the right compiler flags) also on 32 bit platforms correct large files
- With PDF $\geqslant$ 1.5 pdfTEX 1.40.7 should be able to create correct PDF files up to 256 TeB
- With pdfTEX 1.50 the compiler flags etc. will hopefully be automagically correct (thanks to Peter Breitenlohner's `autoconf` changes)
- Included PDFs are currently limited to <4 GiB (XPDF uses `uint` internally)
- On Linux the Adobe Reader 8 has no LFS (it is limited to <2 GiB), and XPDF/KPDF/... are limited to <4 GiB

LFS: 1.40.7

- pdfTEX 1.40.7 creates (with the right compiler flags) also on 32 bit platforms correct large files
- With PDF $\geqslant$ 1.5 pdfTEX 1.40.7 should be able to create correct PDF files up to 256 TeB
- With pdfTEX 1.50 the compiler flags etc. will hopefully be automagically correct (thanks to Peter Breitenlohner's `autoconf` changes)
- Included PDFs are currently limited to <4 GiB (XPDF uses `uint` internally)
- On Linux the Adobe Reader 8 has no LFS (it is limited to <2 GiB), and XPDF/KPDF/... are limited to <4 GiB

LFS: 1.40.7

- pdfTEX 1.40.7 creates (with the right compiler flags) also on 32 bit platforms correct large files
- With PDF $\geqslant$ 1.5 pdfTEX 1.40.7 should be able to create correct PDF files up to 256 TeB
- With pdfTEX 1.50 the compiler flags etc. will hopefully be automagically correct (thanks to Peter Breitenlohner's `autoconf` changes)
- Included PDFs are currently limited to <4 GiB (XPDF uses `uint` internally)
- On Linux the Adobe Reader 8 has no LFS (it is limited to <2 GiB), and XPDF/KPDF/... are limited to <4 GiB

LFS: 1.40.7

- pdfTEX 1.40.7 creates (with the right compiler flags) also on 32 bit platforms correct large files
- With PDF $\geqslant$ 1.5 pdfTEX 1.40.7 should be able to create correct PDF files up to 256 TeB
- With pdfTEX 1.50 the compiler flags etc. will hopefully be automagically correct (thanks to Peter Breitenlohner's `autoconf` changes)
- Included PDFs are currently limited to $<$4 GiB (XPDF uses `uint` internally)
- On Linux the Adobe Reader 8 has no LFS (it is limited to $<$2 GiB), and XPDF/KPDF/... are limited to $<$4 GiB

LFS: 1.40.7

- pdfTEX 1.40.7 creates (with the right compiler flags) also on 32 bit platforms correct large files
- With PDF $\geqslant$ 1.5 pdfTEX 1.40.7 should be able to create correct PDF files up to 256 TeB
- With pdfTEX 1.50 the compiler flags etc. will hopefully be automagically correct (thanks to Peter Breitenlohner's `autoconf` changes)
- Included PDFs are currently limited to $<$4 GiB (XPDF uses `uint` internally)
- On Linux the Adobe Reader 8 has no LFS (it is limited to $<$2 GiB), and XPDF/KPDF/... are limited to $<$4 GiB

Page groups and transparent pngs

- pdfTEX since 1.10a didn't know how to handle Page Groups (used with transparent PDFs), so in some cases the PDF inclusion lost some content
- Thanks to Leonard Rosenthol from Adobe this has been fixed in 1.40.6
- The same problem happend with transparent PNGs and has also been fixed

Page groups and transparent pngs

- pdfTEX since 1.10a didn't know how to handle Page Groups (used with transparent PDFs), so in some cases the PDF inclusion lost some content
- Thanks to Leonard Rosenthol from Adobe this has been fixed in 1.40.6
- The same problem happend with transparent PNGs and has also been fixed

Page groups and transparent pngs

- pdfTEX since 1.10a didn't know how to handle Page Groups (used with transparent PDFs), so in some cases the PDF inclusion lost some content
- Thanks to Leonard Rosenthol from Adobe this has been fixed in 1.40.6
- The same problem happend with transparent PNGs and has also been fixed

pdfTEX 1.40.8

- pdfTEX 1.40.8 will probably be the pdfTEX on TEXlive 2008
- Some minor bugfixes
- The new TEX 3.1415926
- Currently pdf inclusion is broken on ppc-darwin

pdfTEX 1.40.8

- pdfTEX 1.40.8 will probably be the pdfTEX on TEXlive 2008
- Some minor bugfixes
- The new TEX 3.1415926
- Currently pdf inclusion is broken on ppc-darwin

pdfTEX 1.40.8

- pdfTEX 1.40.8 will probably be the pdfTEX on TEXlive 2008
- Some minor bugfixes
- The new TEX 3.1415926
- Currently pdf inclusion is broken on ppc-darwin

pdfTEX 1.40.8

- pdfTEX 1.40.8 will probably be the pdfTEX on TEXlive 2008
- Some minor bugfixes
- The new TEX 3.1415926
- Currently pdf inclusion is broken on ppc-darwin

pdfTEX 1.50

- Status: early α
- pool file
- Page diversions
- Optional Content
- SyncTEX

pdfTEX 1.50

- Status: early α
- pool file
- Page diversions
- Optional Content
- SyncTEX

pdfTeX 1.50

- Status: early α
- pool file
- Page diversions
- Optional Content
- SyncTeX

pdfTeX 1.50

- Status: early α
- pool file
- Page diversions
- Optional Content
- SyncTeX

pdfTEX 1.50

- Status: early α
- pool file
- Page diversions
- Optional Content
- SyncTEX

pool file

- `tangle` saves the text constants of a Web program in a separate file, the pool file
- The pool file must match the program (and the program the format)
- In theory one can use the pool file to change the messages of the program without recompiling the program and the format, but nobody (but some polish guys?) really does that
- Since pdfTEX 1.50 the pool file is compiled into the program; this was developed by Taco for luaTEX and recently integrated into TEXlive 2008. So future TEX systems have one file less

pool file

- `tangle` saves the text constants of a Web program in a separate file, the pool file
- The pool file must match the program (and the program the format)
- In theory one can use the pool file to change the messages of the program without recompiling the program and the format, but nobody (but some polish guys?) really does that
- Since pdfTEX 1.50 the pool file is compiled into the program; this was developed by Taco for luaTEX and recently integrated into TEXlive 2008. So future TEX systems have one file less

pool file

- `tangle` saves the text constants of a Web program in a separate file, the pool file
- The pool file must match the program (and the program the format)
- In theory one can use the pool file to change the messages of the program without recompiling the program and the format, but nobody (but some polish guys?) really does that
- Since pdfTEX 1.50 the pool file is compiled into the program; this was developed by Taco for luaTEX and recently integrated into TEXlive 2008. So future TEX systems have one file less

pool file

- `tangle` saves the text constants of a Web program in a separate file, the pool file
- The pool file must match the program (and the program the format)
- In theory one can use the pool file to change the messages of the program without recompiling the program and the format, but nobody (but some polish guys?) really does that
- Since pdfTEX 1.50 the pool file is compiled into the program; this was developed by Taco for luaTEX and recently integrated into TEXlive 2008. So future TEX systems have one file less

Page diversions: The idea

- In PDF the logical order of the pages of a document is not given by the sequence in the file, but through a tree on the page objects
- This tree can be rearranged
- The idea for diversions comes from m4, where it is used for text (`divert` and `undivert`)

Page diversions: The idea

- In PDF the logical order of the pages of a document is not given by the sequence in the file, but through a tree on the page objects
- This tree can be rearranged
- The idea for diversions comes from m4, where it is used for text (`divert` and `undivert`)

Page diversions: The idea

- In PDF the logical order of the pages of a document is not given by the sequence in the file, but through a tree on the page objects
- This tree can be rearranged
- The idea for diversions comes from m4, where it is used for text (`divert` and `undivert`)

Page diversions: How it works

- Pages are added to a list, and these lists are selected by positive integers. The default list is 0, and with `\pagedivert` <number> one switches to the list <number>
- A page is added to the list active at `\shipout`
- With `\pdfpageundivert` <number> the pages of the list <number> are added to the active list. The undiverted list is cleared.
- With `\pdfpageundivert 0` all lists are added to the current active list in ascending order
- At the end of the document an automagic `\pdfpagedivert 0` `\pdfpageundivert 0` is added

Page diversions: How it works

- Pages are added to a list, and these lists are selected by positive integers. The default list is 0, and with `\pagedivert` <number> one switches to the list <number>
- A page is added to the list active at `\shipout`
- With `\pdfpageundivert` <number> the pages of the list <number> are added to the active list. The undiverted list is cleared.
- With `\pdfpageundivert 0` all lists are added to the current active list in ascending order
- At the end of the document an automagic `\pdfpagedivert 0` `\pdfpageundivert 0` is added

Page diversions: How it works

- Pages are added to a list, and these lists are selected by positive integers. The default list is 0, and with `\pagedivert` <number> one switches to the list <number>
- A page is added to the list active at `\shipout`
- With `\pdfpageundivert` <number> the pages of the list <number> are added to the active list. The undiverted list is cleared.
- With `\pdfpageundivert 0` all lists are added to the current active list in ascending order
- At the end of the document an automagic `\pdfpagedivert 0` `\pdfpageundivert 0` is added

Page diversions: How it works

- Pages are added to a list, and these lists are selected by positive integers. The default list is 0, and with `\pagedivert` <number> one switches to the list <number>
- A page is added to the list active at `\shipout`
- With `\pdfpageundivert` <number> the pages of the list <number> are added to the active list. The undiverted list is cleared.
- With `\pdfpageundivert 0` all lists are added to the current active list in ascending order
- At the end of the document an automagic `\pdfpagedivert 0` `\pdfpageundivert 0` is added

Page diversions: How it works

- Pages are added to a list, and these lists are selected by positive integers. The default list is 0, and with `\pagedivert` <number> one switches to the list <number>
- A page is added to the list active at `\shipout`
- With `\pdfpageundivert` <number> the pages of the list <number> are added to the active list. The undiverted list is cleared.
- With `\pdfpageundivert 0` all lists are added to the current active list in ascending order
- At the end of the document an automagic `\pdfpagedivert 0` `\pdfpageundivert 0` is added

Page diversions: Applications

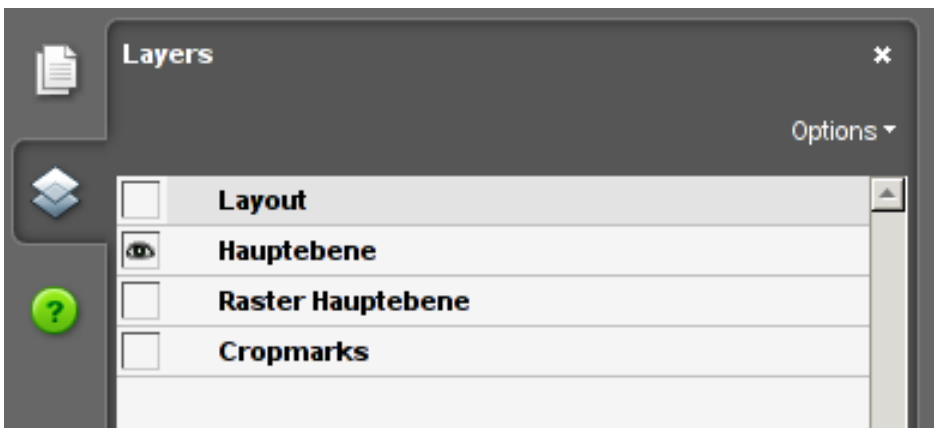
- Getting a correct table of contents in the first run
- Creating a document with reversed page order

Page diversions: Applications

- Getting a correct table of contents in the first run
- Creating a document with reversed page order

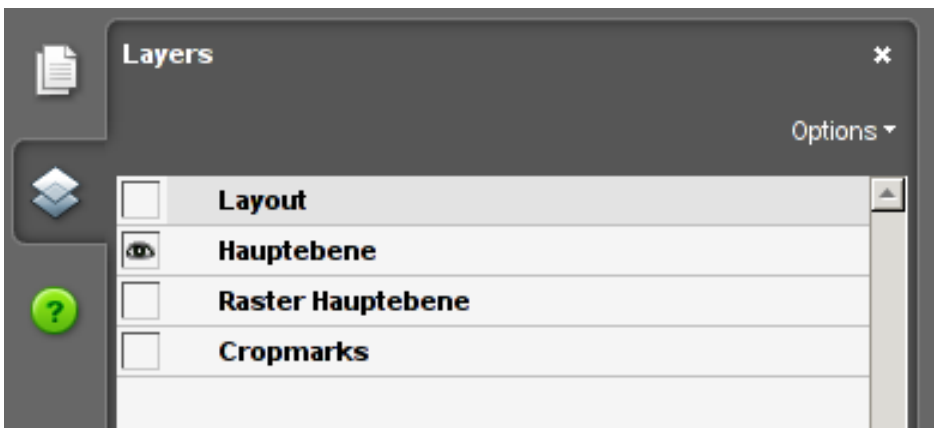
Optional Content

- Optional content (also called “layers”) allow to disable or enable parts of a PDF
- Optional content is in PDF since 1.1, but since PDF 1.5 there’s also the possibility for a GUI for switching between them
- This is a GUI in AR8:



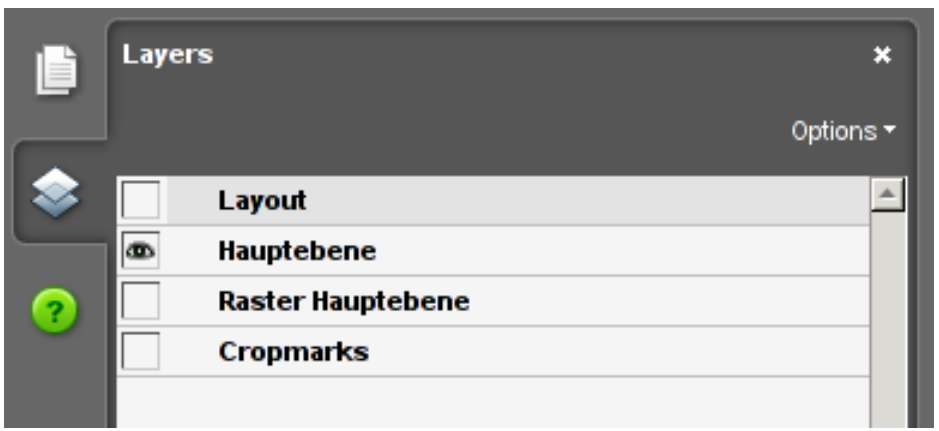
Optional Content

- Optional content (also called “layers”) allow to disable or enable parts of a PDF
- Optional content is in PDF since 1.1, but since PDF 1.5 there’s also the possibility for a GUI for switching between them
- This is a GUI in AR8:



Optional Content

- Optional content (also called “layers”) allow to disable or enable parts of a PDF
- Optional content is in PDF since 1.1, but since PDF 1.5 there’s also the possibility for a GUI for switching between them
- This is a GUI in AR8:



Optional Content in pdfTeX

- Optional content can be created with `\pdfliteral`
- Including PDFs with optional content needs support from pdfTeX, which will be added in 1.50
- There will be primitives to find out the number of layers and their names
- To recreate the gui for the layers one needs access to the object numbers of the layers; this will be possible
- It will also be possible to unify layers between included PDFs and the main document

Optional Content in pdfTeX

- Optional content can be created with `\pdfliteral`
- Including PDFs with optional content needs support from pdfTeX, which will be added in 1.50
- There will be primitives to find out the number of layers and their names
- To recreate the gui for the layers one needs access to the object numbers of the layers; this will be possible
- It will also be possible to unify layers between included PDFs and the main document

Optional Content in pdfTeX

- Optional content can be created with `\pdfliteral`
- Including PDFs with optional content needs support from pdfTeX, which will be added in 1.50
- There will be primitives to find out the number of layers and their names
- To recreate the gui for the layers one needs access to the object numbers of the layers; this will be possible
- It will also be possible to unify layers between included PDFs and the main document

Optional Content in pdfTEX

- Optional content can be created with `\pdfliteral`
- Including PDFs with optional content needs support from pdfTEX, which will be added in 1.50
- There will be primitives to find out the number of layers and their names
- To recreate the gui for the layers one needs access to the object numbers of the layers; this will be possible
- It will also be possible to unify layers between included PDFs and the main document

Optional Content in pdfTeX

- Optional content can be created with `\pdfliteral`
- Including PDFs with optional content needs support from pdfTeX, which will be added in 1.50
- There will be primitives to find out the number of layers and their names
- To recreate the gui for the layers one needs access to the object numbers of the layers; this will be possible
- It will also be possible to unify layers between included PDFs and the main document

SyncTEX

- SyncTEX is a patch by Jérôme Laurens for synchronizing PDF output and input files, which has previously been done (badly) with `src-specials`.
- With `\synctex=1` pdfTEX writes the name of the input file, line number and column and the position in the output (page, position on the page) in a separate file whenever these change
- With this information a viewer can instruct an editor to jump to the right position in the input file
- SyncTEX will also be supported in XETEX

SyncTeX

- SyncTeX is a patch by Jérôme Laurens for synchronizing PDF output and input files, which has previously been done (badly) with `src-specials`.
- With `\synctex=1` pdfTeX writes the name of the input file, line number and column and the position in the output (page, position on the page) in a separate file whenever these change
- With this information a viewer can instruct an editor to jump to the right position in the input file
- SyncTeX will also be supported in XeTeX

SyncTeX

- SyncTeX is a patch by Jérôme Laurens for synchronizing PDF output and input files, which has previously been done (badly) with `src-specials`.
- With `\synctex=1` pdfTeX writes the name of the input file, line number and column and the position in the output (page, position on the page) in a separate file whenever these change
- With this information a viewer can instruct an editor to jump to the right position in the input file
- SyncTeX will also be supported in XeTeX

SyncTeX

- SyncTeX is a patch by Jérôme Laurens for synchronizing PDF output and input files, which has previously been done (badly) with `src-specials`.
- With `\synctex=1` pdfTeX writes the name of the input file, line number and column and the position in the output (page, position on the page) in a separate file whenever these change
- With this information a viewer can instruct an editor to jump to the right position in the input file
- SyncTeX will also be supported in XeTeX

pdfTeX and XPDF

- pdfTeX uses XPDF for PDF inclusion
- XPDF is written in C++ and used only in one source file (`pdftoepdf.cc`) of pdfTeX (which is C otherwise)
- There is an additional layer of abstraction between pdfTeX and XPDF in pdfTeX 1.50
- XPDF is statically compiled into pdfTeX

pdfTeX and XPDF

- pdfTeX uses XPDF for PDF inclusion
- XPDF is written in C++ and used only in one source file (`pdftoepdf.cc`) of pdfTeX (which is C otherwise)
- There is an additional layer of abstraction between pdfTeX and XPDF in pdfTeX 1.50
- XPDF is statically compiled into pdfTeX

pdfTEX and XPDF

- pdfTEX uses XPDF for PDF inclusion
- XPDF is written in C++ and used only in one source file (`pdftoepdf.cc`) of pdfTEX (which is C otherwise)
- There is an additional layer of abstraction between pdfTEX and XPDF in pdfTEX 1.50
- XPDF is statically compiled into pdfTEX

pdfTEX and XPDF

- pdfTEX uses XPDF for PDF inclusion
- XPDF is written in C++ and used only in one source file (`pdftoepdf.cc`) of pdfTEX (which is C otherwise)
- There is an additional layer of abstraction between pdfTEX and XPDF in pdfTEX 1.50
- XPDF is statically compiled into pdfTEX

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF

Other PDF libs

- Most PDF libraries only support writing PFDs; some support reading and very few support reading and writing (manipulating)
- poppler is a fork of XPDF focused on providing a shared library; recently some support for writing PDFs has been added
- podofo is a C++ library for manipulating PDFs that offers the podofobrowser, a very useful tool for analyzing PDFs
- GNU PDF has recently been started to create a free C library for manipulating PDF; development is still in the very early stages
- iText is a Java library for manipulating PDF used in pdftk
- MuPDF is a C library from the GhostScript guys for manipulating PDF