Special and common aspects of pdf/dvi/xdvi generators

Ernst Reissner (rei3ner@arcor.de)

2024-9-26T22:46

Contents

List of Tables

List of Listings

1 Introduction

This document is created with `lualatex` or that like with output format PDF. The package `tex4ht` is not loaded.

This document is about features the three LaTeX engines, `pdflatex`, `lualatex` and `xelatex` have in common and discusses also aspects under which they are specific. These programs are just underlying TeX engines preloading the LaTeX format. The names of the underlying engines just drop the inner sylable "`la`". For `pdftex` there is a user manual [THHB24], for `luatex` there is a reference manual [HHHS24] and for `xetex` a reference guide [RHB24].

The first aspect we cover are the options (among those to display the version). It turns out, that the options are specific for the distribution. Apart from TeX Live, there is a second important distribution, MiKTeX, which should also be treated.

We treat the options for TeX Live in Section 2 and those for MiKTeX in Section 3.

The second subject is privacy as part of security and reproducible builds of documents, e.g. for tests. The focus is here on PDF files and comprises besids visible data also metadata. The results of our research is collected in Section 4.

2 The options for TeX Live

Note that in fact we use a variant of `luatex`, called `luahbtex`.

This document is valid for versions of the underlying LaTeX engine as given in Table 1. Moreover, our research refers to a specific distribution, TeX Live.

LaTeX engine	version
`pdflatex`	`pdfTeX 3.141592653-2.6-1.40.24`
`xelatex`	`XeTeX 3.141592653-2.6-0.999994`
`lualatex`	`LuaHBTeX, Version 1.15.0`

Table 1: LaTeX engines and the version this document refers to

We start with a synopsis of the options. Table 2 shows options of the LaTeX engines under consideration. Note that in contrast to the other LaTeX engines, `lualatex` defines options starting with `--` but it can also process the options if given with a single dash also. Conversely, LaTeX engines other than `lualatex` can also deal with options starting with single dash. Options unknown to a LaTeX engine never result in an error or even a warning; instead just an info message is displayed. This allows to create a configuration which works for all LaTeX engines.

In Table 2 column "included", each LaTeX engine is represented by the starting letter of its name, so for each option it is known which LaTeX engines know about it and conversely, which options each LaTeX engine has.

The table allows furnishing configurations working for all LaTeX engines. Some options are common to all LaTeX engines

option	included			explanation
`(-)-cnf-line=STRING`	p	x	l	parse `STRING` as a configuration file line
`--credits`	-	-	l	Display credits and exit.
`--debug-format`	-	-	l	enable format debugging
`(-)-draftmode`	p	-	l	switch on draft mode (generates no output PDF)
`-enc`	p	-	-	Enable encTeX extensions such as `\mubyte`
`-etex`	p	x	-	enable e-TeX extensions
`(-)-[no-]file-line-error`	p	x	l	disable/enable file:line:error style messages
`--[no-]file-line-error-style`	-	-	l	aliases of `--[no-]file-line-error`
`-fmt=FMTNAME`	p	x	l	use FMTNAME instead of program name or a %& line[1]
`(-)-ini`	p	x	l	for dumping formats
`-ipc`	p	-	-	send DVI output to a socket as well as the usual output file
`-ipc-start`	p	-	-	as `-ipc`, and also start the server at the other end
`(-)-halt-on-error`	p	x	l	stop processing at the first error
`(-)-help`	p	x	l	display this help and exit
`(-)-version`	p	x	l	output version information and exit
`-8bit`	p	x	-	make all characters printable by default[2]
`(-)-interaction=STRING`	p	x	l	set interaction mode (`STRING=batchmode/nonstopmode/scrollmode/errorstopmode`)
`(-)-jobname=STRING`	p	x	l	set the job name to `STRING`
`(-)-kpathsea-debug=NUMBER`	p	x	l	set path searching debugging flags according to the bits of `NUMBER`
`--lua=FILE`	-	-	l	Load and execute a lua initialization script.
`--luaonly`	-	-	l	run a lua file, then exit
`--luaconly`	-	-	l	byte-compile a lua file, then exit
`--luahashchars`	-	-	l	the bits used by current Lua interpreter for strings hashing
`(-)-[no-]mktex=FMT`	p	x	l	disable/enable mktexFMT generation[3]
`-mltex`	p	x	-	enable MLTeX extensions such as `\charsubdef`
`-no-pdf`	-	x	-	generate XDV (extended DVI) output rather than PDF
`--nosocket`	-	-	l	Disable the Lua socket library.
`(-)-output-comment=STRING`	p	x	l	use STRING for DVI file comment instead of date (no effect for PDF)[4]
`(-)-output-directory=DIR`	p	x	l	use existing DIR as the directory to write files in
`(-)-output-format=FORMAT`	p	-	l	use `FORMAT` for job output; `FORMAT` is 'dvi' or 'pdf'[5]
`-output-driver=CMD`	-	x	-	use `CMD` as the XDV-to-PDF driver instead of `xdvipdfmx`
`-papersize=STRING`	-	x	-	set PDF media size to `STRING`
`-[no-]parse-first-line`	p	x	-	disable/enable parsing of first line of input file
`(-)-progname=STRING`	p	x	l	set program (and fmt) name to `STRING`[6]
`(-)-recorder`	p	x	l	enable filename recorder
`--safer`	-	-	l	Disable easily exploitable Lua commands.
`(-)-[no-]shell-escape`	p	x	l	disable/enable `\write18SHELL COMMAND`[7]
`(-)-shell-restricted`	p	x	l	enable restricted `\write18`[8]
`-src-specials`	p	x	-	insert source specials into the DVI file
`-src-specials=WHERE`	p	x	-	insert source specials in certain places of the DVI/XDV[9] file.
`(-)-synctex=NUMBER`	p	x	l	generate SyncTeX data for previewers[10]
`-translate-file=TCXNAME`	p			use the TCX file `TCXNAME`[11]

[1] in fact for `lualatex` the explanation deviates a bit: `--fmt=FORMAT: load the format file FORMAT`
[2] for `xelatex`: don't use ^^X sequences
[3] `(FMT=tex/tfm/pk)` for `pdflatex`; else `(FMT=tex/tfm)`
[4] For `xelatex` it is XDV instead of DVI and the remark `(no effect for PDF)` is missing
[5] `xelatex` offers option `-no-pdf` instead.
[6] `lualatex` does not mention `(and fmt)`
[7] For `lualatex` the explanation is `disable/enable system commands`
[8] For `lualatex` the explanation is `restrict system commands to a list of commands given in texmf.cnf`
[9] DVI for `pdflatex`; XDV for `xelatex`
[10] Explanation differs for `lualatex`
[11] TCX means TEX character translation

`--utc`	l	Init time to UTC[12]

Table 2: Options of TEX engines in TEX Live

pdflatex:

```
Usage: pdftex [OPTION]... [TEXNAME[.tex]] [COMMANDS]
   or: pdftex [OPTION]... \FIRST-LINE
   or: pdftex [OPTION]... &FMT ARGS
  Run pdfTeX on TEXNAME, usually creating TEXNAME.pdf.
  Any remaining COMMANDS are processed as pdfTeX input, after TEXNAME is read.
  If the first line of TEXNAME is %&FMT, and FMT is an existing .fmt file,
  use it.  Else use `NAME.fmt', where NAME is the program invocation name,
  most commonly `pdftex'.

  Alternatively, if the first non-option argument begins with a backslash,
  interpret all non-option arguments as a line of pdfTeX input.

  Alternatively, if the first non-option argument begins with a &, the
  next word is taken as the FMT to read, overriding all else.  Any
  remaining arguments are processed as above.

  If no arguments or options are specified, prompt for input.
```

3 The options for MiKTeX

Since at the time of this writing, the author has no MiKTeX at hand, the results for MiKTeX are based on documentation, rather than experimentation. The three engines are a bit different, also in their names.

Well this section is preliminary only. It turned out that Section 2 is valid only for distribution TEX Live. So in this section we venture to find out the options for the other big distribution, MiKTeX. We shall also investigate whether there are further distributions.

Whereas the description [KB23] seems not to mention the options explicitly, the MiKTeX manual [Sch22] describes each program in Section II, 6, in particular also the LATEX engines. This is the source of the following tables.

The first observation is that, for MiKTeX all options start with two dashes, whereas for TEX Live this is the case only for luatex. One has to clarify, whether the maven latex plugin under consideration really works for MiKTeX.

option	included			explanation
`--alias=name`	p	x	l	Pretend to be program name, …[13]
`--aux-directory=dir`	p	x	l	Set `dir` as the directory to write auxiliary files to.
`--buf-size=n`	p	x	-	Set the the maximum number of characters …
`--c-style-errors`	p	x	l	Change the way, error messages are printed.
`--credits`	-	-	l	Display credits and exit[14].
`--disable-8bit-chars`	p	x	-	Make only 7-bit characters printable.
`--disable/enable-installer`	p	x	l	Disable/Enable automatic installation of packages.

[12] Coordinated Universal Time, successor of Greenwich Mean Time (GMT)

[13] Using this option is equivalent to copying the program file to name and invoking name.

[14] The same as for TEX Live.

`--disable-write18`	p	x	l	Disable the `\write18{command}` construct.
`--enable-write18`	p	x	l	Fully enable the `\write18{command}` construct[15].
`--restrict-write18`	p	x	l	Partially enable the `\write18command` construct.
`--debug-format`	-	-	l	Enable format debugging[16].
`--[dont-]parse-first-line`	p	x	-	[Dont p|P]arse first line of input file under definite conditions[17]
`--draftmode`	p	-	l	switch on draft mode (generates no output PDF)[18]
`--enable-8bit-chars`	p	x	-	Make all characters printable.
`--enable-enctex`	p	-	-	Enable encTeX extensions such as `\mubyte`[19].
`--enable-etex`	p	x	-	Enable eTeX extensions.
`--enable-installer`	p	x	l	Enable automatic installation of packages.
`--enable-mltex`	p	x	-	Enable MLTeX extensions such as `\charsubdef`.
`--error-line=n`	p	x	-	Set the width of context lines on …error messages.
`--extra-mem-bot=n`	p	x	-	Set the extra size … for large data structures …
`--extra-mem-top=n`	p	x	-	Set the extra size (in memory words) for chars, tokens, ….
`--font-max=n`	p	x	-	Set the maximum internal font number.
`--font-mem-size=n`	p	x	-	Set the size, in TeX memory words, of the font memory.
`--half-error-line=n`	p	x	-	Set the width of first lines of contexts in terminal error messages.
`--halt-on-error`	p	x	l	Quit after the first error.
`--hash-extra=n`	p	x	-	Set the extra space for the hash table of control sequences …
`--help`	p	x	l	Give help and exitt[20]..
`--hhelp`	p	x	-	manual page in an HTML Help window[21]
`--include-directory=dir`	p	x	l	Add the directory `dir` to [those] to be searched for input files.
`--initialize`	p	x	l	Become the INI variant of the program.
`--interaction=mode`	p	x	l	Set the interaction mode (`mode=batchmode/nonstopmode/scrollmode/errorstopmode`).
`--job-name=name`	p	x	l	Set the name of the job (`\jobname`).
`--job-time=file`	p	x	-	Set the time-stamp of all output files equal to file's time-stamp.
`--lua=FILE`	-	-	l	load and execute a lua initialization script[22].
`--luaonly`	-	-	l	Start LuaTeX as a Lua interpreter[23].
`--luaconly`	-	-	l	byte-compile a lua file, then exit[24].
`--luahashchars`	-	-	l	the bits used by current Lua interpreter for strings hashing
`--main-memory=n`	p	x	-	Change the total size … of the main memory array.
`--max-in-open=n`	p	x	-	Set the maximum number of input files …
`--max-print-line=n`	p	x	-	Set the width of longest text lines output.
`--max-strings=n`	p	x	-	Set the maximum number of strings.
`--[no-]mktex=fmt`	-	-	l	Enable/Disable `fmt` generation, where `fmt` must be either `tex` or `tfm`.
`--nest-size=n`	p	x	-	Set the maximum number of semantic levels simultaneously active.
`--no-c-style-errors`	p	x	l	Don't change the way, error messages are printed.
`--no-pdf`	-	x	-	generate XDV (extended DVI) output rather than PDF
`--nosocket`	-	-	l	Disable the Lua socket library.
`--output-comment=string`	-	-	l	Use `string` for DVI file comment instead of date.
`--output-directory=dir`	p	x	l	Write output files in `dir`[25].
`--output-driver=CMD`	-	x	-	use CMD as the XDV-to-PDF driver instead of xdvipdfmx

[15] Corresponds roughly to `(-)-shell-escape` in TeX Live.
[16] The same as for TeX Live.
[17] Similar for TeX Live. Note that there is also a converse option.
[18] Some differences in formulation between the LaTeX engines and also between distributions
[19] Corresponds with `-enc` in TeX Live.
[20] The same as for TeX Live.
[21] This option is only available on Windows systems.
[22] The same as for TeX Live.
[23] Could be the same as for TeX Live.
[24] Could be the same as for TeX Live.
[25] Similar as for TeX Live.

`--output-format=format`	p	-	l	Use format for job output (one of: dvi, pdf)[26].
`--papersize=STRING`	-	x	-	set PDF media size to STRING
`--param-size=n`	p	x	-	Set the the maximum number of simultaneous macro parameters.
`--pool-free=n`	p	x	-	Set the minimum pool space left after loading the format.
`--pool-size=n`	p	x	-	Set the maximum number of characters in strings, …
`--quiet`	p	x	-	Suppress all output, except errors.
`--record-package-usages=file`	p	x	-	Record all package usages and write them into `file`.
`--recorder`	p	x	l	Enable the file name recorder[27].
`--safer`	-	-	l	Disable easily exploitable Lua commands[28].
`--save-size=n`	p	x	-	Set the the amount of space for saving values outside of current group.
`--src-specials`	p	x	-	Embed source file information in the DVI file[29].
`--stack-size=n`	p	x	-	Set the maximum number of simultaneous input sources.
`--string-vacancies=n`	p	x	-	Set the minimum number of characters …
`--synctex=n`	p	x	l	Generate SyncTeX data for previewers[30]
`--tcx=tcxname`	p	-	-	Use the tcxname translation table …
`--time-statistics`	p	x	-	Show processing time statistics.
`--trace[=tracestreams]`	p	x	-	Enable trace messages.
`--trie-size=n`	p	x	-	Set the amount of space for hyphenation patterns.
`--undump=name`	p	x	l	Use name as the name of the format to be used, …
`--utc`	-	-	l	Init time to UTC[31].
`--version`	p	x	l	Show version information and exit[32].

Table 3: Options of TEX engines in MiKTeX

Strange, there are `--enable-etex` and `--enable-mltex` but no way to disable. Maybe disable is the default.

```
  miktex-pdftex [option...] [[file] | [\command...]]

  miktex-luatex [option...] [[command...] | [file]]

The following options are ignored:

--8bit, --etex, --parse-first-line, --no-parse-first-line
These are always on.

--default-translate-file=tcxname, --translate-file=tcxname
These are always off.

  miktex-xetex [option...] [[file] | [\command...]]
```

4 Reproducibility and security for PDF documents created

Of course reproducibility and security are general subjects not tied to a specific format like PDF, but as a first step we undertake a discussion specific for PDF files. This shall be extended step by step.

[26] `pdflatex` and `lualatex` differ a bit in text. Seems similar to TEX Live.
[27] The same as in TEX Live.
[28] The same as for TEX Live.
[29] Similar as in TEX Live.
[30] Explanation with more detail than for TEX Live.
[31] The same as for TEX Live.
[32] The same as for TEX Live.

Reproducible builds are important for tests and for global cooperation. For PDF files besides visible data also so-called metadata must be reproduced. Security is mainly privacy here. Also, besides visible data also metadata shall not expose private data.

To display metadata, we use `exiftool` and `pdfinfo`.

4.1 The tool chains

Although `xelatex` always produces a XDV file ias an intermediate step, when creating a PDF file this is eliminated. In contrast, with the option `-no-pdf` one can eliminate creation of the PDF file and the XDV is not erased. For conversion, of the XDV file to PDF, the option `-output-driver=CMD` is used which defaults to the command `xdvipdfmx`. Besides direct creation of a PDF file, we consider creation via XDV file using `xdvipdfmx`. The XDV format is an extension and in fact a variant of the DVI format.

For the other engines in contrast, the option `-output-format=dvi/pdf` determines the output format which is PDF by default and there is no intermediate format for PDF. When creating DVI files instead, these files can be converted into PDF by explicitly invoking something like `dvipdfmx`, `dvipdfmx` or `xdvipdfmx`. In my current distribution TEX Live, the programs `dvipdfm`, `dvipdfmx` and `xdvipdfmx` are all binary identical.

Nevertheless, they turn out to yield different results. One reason found below is, that the name with which the program is invoked goes into the result. It is likely that this is the only reason.

As a consequence of the workflow of `xelatex`, `\ifpdf` provided by package `iftex` always enters the `\else` branch for `xelatex`.

Although this section focuses on PDF format, of course the intermediate format DVI/XDV must be considered by need. The visual appearance is roughly the same, so visual reproducibility of DVI/XDV is almost equivalent with visual reproducibility of PDF, but for metadata one has to distinguish cases where the LATEX engine already creates metadata and writes it into `\special` commands in a DVI/XDV file, whereas the PDF creator just includes this data from DVI/XDV from cases where it is the PDF creator which creates metadata, like creation time.

4.2 Dates, times and time zones

Clearly, creation times affect reproducibility quite directly, because usually the date is given in the head page of a document, but it goes also into metadata both directly and indirectly.

Before discussing aspects of time in the context of PDF creation from LATEX,

4.2.1 Date/time formats and time zones

This section introduces date/time formats, time zones and conversions between.

The description is started with *epoch time* which is the most important although not well human-readable. It is the number of seconds from start of computer time history, which is agreed to be

```
1970-01-01T00:00:00Z
```

i.e. the point in time, where in timezone UTC (represented by the `Z`, zero deviation from UTC) it was start of year 1970, i.e. date `1970--01--01` and midnight, time `00:00:00`. It is called epoch time, although nothing special happened like Christs birth. Zero and negative time is allowed. Note, that this does not depend on a time zone, but as human-readable time formats depend on the time zone, conversion from and to epoch time must take time zone into account.

In fact, the above date is an example of a representation given by [ISO19]. More general, the timezone is either `Z` representing UTC or some deviation from UTC for example with two digits for hours and two digits for minutes. For example the same date as above in time zone two hours earlier (more east) than UTC writes

```
1970-01-01T02:00:00+0200
```

Among the standard environment variables in POSIX operating systems is also `TZ` specifying the time zone as described in `https://www.gnu.org/software/libc/manual/html_node/Standard-Environment.html`. We only use the forms based on UTC like so:

```
TZ=UTC, TZ=UTC-02
```

CAUTION: The number is understood as to be added to given time to get UTC, so east of UTC has negative numbers. This is the sign opposite to the representation given by ISO 8601 which is the offset to be added to UTC to get time given.

In Unix and similar operating systems, conversion can be done with `date`. If a time is given with option `-d`, this is converted into the format given by the format specifier which starts with `+`. If no time is given, the current time is assumed.

`date`
: yields the current date and time with timezone in current locale representation. The locale is given by the environment variable `LANG`.

`date +%FT%T%z`
: yields the current date and time in form given by ISO 8601 with date given by `%F` followed by literal `T` followed by time `%T` and finally the time zone given by `%z`.

`date -d 'Apr 29 2024' +%s`
: converts human-readable date into epoch time specified by format `%s` (seconds since 1970–01–01, i.e. epoch time). **Caution**: Many formats are recognized, but time zones are *silently ignored*. For example,

```
date -d '2024-06-20T12:03:00+03' +%s
```

silently ignores the timezone `+03` and uses the current one which is `UTC+02`, because me the author is located in Germany. This can be seen, because the result does not depend on the time zone; you can use `+02` or `Z` signifying UTC or drop: the result is always `1718877780`.

To set the time zone, the variable `TZ` must be used like so:

```
TZ=UTC-02 date -d '2024-06-20T12:03:00' +%s
```

Note the sign difference between ISO 8601 representation and `TZ`. Since `UTC+02` is the timezone of the author, the result is still `1718877780`, but if using `TZ=UTC-03`, which is one hour east, it is one hour, i.e. `3600` seconds less which results in epoch time `1718874180`.

`date -d '@1718877780' +%FT%T%z`
: converts from epoch time (signified by leading `@`) into local time zone resulting in

```
2024-06-20T12:03:00+0200
```

Since the time zone of the author is currently `GTM+02`, or with minutes `GTM+0200` this is what the resulting time string ends with.

`TZ=UTC-03 date -d '@1718877780' +%FT%T%z`
converts from epoch time (signified by leading `@`) into time zone given by `TZ` resulting in

```
2024-06-20T13:03:00+0300
```

Compared with the result above, the time which is one hour later and this is reflected by the according time zone. Note again the sign contrary to the value of `TZ`.

Finally, [ISO20], Section 7.9.4 describes the encoding of date/time with timezone in metadata of PDF files. It is quite similar to the format given by ISO 8601–1. Transformation is by dropping separators `--` and `:`, prefixing `D:` and inserting `'` separating hours and minutes in time zone. The above date/time writes as follows:

```
D:20240620130300+03'00
```

4.2.2 Date, time and time zones in PDF files created from LaTeX files

When creating documents from LaTeX, date and time information go into at least in the following aspects: in

- LaTeX macros displaying time and date in the document like the built-in `\today`, `\date` and `\time` accessible with prefixed `\the`, and also `\DTMnow` provided by package `datetime2`
- metadata of PDF documents like `CreationDate` and `ModDate`
- file properties like creation time and modification time.

There are several environment variables affecting treatment of these aspects. First we analyze behavior without them.

Let us start with the last aspect: For Linux, the command `stat` yields both creation time and modification time of a file `xxx.pdf`:

`stat -c +%w xxx.pdf` is the time of file birth, i.e. creation time

`stat -c +%y xxx.pdf` is the modification time. This is the same as given by `ls -l`.

Both times comprise the date are endowed with time zone information and are given in human-readable form, although not precisely conform with ISO 8601–1. The command `stat` assumes the local time zone by default, but is sensitive to `TZ`:

```
TZ=UTC stat -c +%w xxx.pdf
```

displays the result in UTC time zone. The according pieces of information in epoch time are accessed with option in capital letters, i.e. `+%W` and `+%Y`, respectively.

As one would expect, the modification time is the time when the engine finished writing. The time of file birth is really when the file comes into existence, e.g. for the first time or after having been deleted.

The first surprise with metadata `CreationDate` and `ModDate` is, that in fact it is not only date but comprises time and timezone as well, much the same as the according file properties. The form in which they are given in the PDF files is described in Section 4.2.1. This is essentially accessible via

```
  %pdfsource    ={no latex main file}
}
%\usepackage{datetime2}% not with dvi

\ExplSyntaxOn%

\NewExpandableDocumentCommand{\printtime}{}
 {
  \int_compare:nT { \c_sys_hour_int < 10 } { 0 }
  \int_eval:n { \c_sys_hour_int }
  :% chktex 26
  \int_compare:nT { \c_sys_minute_int < 10 } { 0 }
```

Listing 1: Definition of `\printdate`

```
pdfinfo -rawdates xxx.pdf
```

The data is considered raw because it is displayed as is, i.e. as stored in the PDF file. It is close to ISO 8824–1. Dates are also accessible in format specified in ISO 8601–1 via

```
pdfinfo -isodates xxx.pdf
```

The second surprise is, that the creation date and modification date given by metadata do not coincide with the according file properties.

First, `xelatex` is special in that by default, it provides `CreationDate` only, whereas `lualatex` and `pdflatex` provide both. The command `\DocumentMetadata` forces `xelatex` to display both times. This is not really important, because the two times seem to coincide if both are given. The idea behind creation time is, that the LaTeX engine creates a new file even when overwriting an old one. The time when starting to overwrite seems to be the `CreationDate`. Why the `ModDate` is the same and not the later time when writing is finished, well…the reader may find an explanation him/herself. An idea is that this is related to incremental updates described in [ISO20], Section 7.5.6: Instead of creating a PDF anew if a modification occurs, it is advised to just append a description of the difference. This update does not affect creation date but only modification date. LaTeX engines no not support incremental updates and so compilation creates just a new PDF document.

Both, `xelatex` and `pdflatex` represent `CreationDate` and `ModDate` with the local time zone which can be overwritten by `TZ`, and so does `lualatex` by default. In addition, `lualatex` has an option `--utc` both for TeX Live and for MiKTeX described in Sections 2 and 3 which forces metadata given in UTC zone, overwriting internal timezone but also the value of `TZ` if provided.

Finally, we have to consider the visible date and time displayed by commands like `\date` or `\time`, to be and `\DTMnow` provided by package `datetime2`. Unfortunately, `\time`, the number of minutes since midnight is not well human-readable. Thus, for experiments we use `\printdate` defined in Listing 1 which is based on `\time`.

Time and date in the visible document, e.g. by `\DTMnow` behave exactly as the date and time given by the metadata, except that `xelatex` does not display seconds in visible document and does not make the time zone visible, although metadata show that both pieces of information are available.

Note that as `lualatex` represents metadata in UTC time if invoked with option `--utc`, the same is true for visible date and time. The time zone is made explicit. Note that the variable `TZ`

is treated as for metadata: Goes into representation, except for `lualatex` invoked with option `--utc`.

The environment variable `SOURCE_DATE_EPOCH` defined in `https://reproducible-builds.org/docs/source-date-epoch/`, is set with epoch time. It was introduced to obtain reproducible builds in a general context can also be used in TEX systems. Although `SOURCE_DATE_EPOCH` defines the build time in a sense, it is specified, that the given epoch time is exactly what is written into the metadata. It affects each LATEX engine but in slightly different way.

For `pdflatex`, the main description is in [THHB24], Chapter 2. An important detail is, that the time zone written into metadata is UTC. Even if the timezone is given explicitly by `TZ`, it is ignored. The time given by `SOURCE_DATE_EPOCH` affects metadata but also `\DTMnow`. To affect also primitives like `\time`, one has to add `FORCE_SOURCE_DATE=1` in addition. Essentially, `xelatex` behaves the same as described in`xelatex`: [RHB24], Sections 8.3 and 8.5.

In contrast to this, as described in [HHHS24], Section 4.4, `lualatex` does honor the environment variable `SOURCE_DATE_EPOCH` but not `FORCE_SOURCE_DATE`. Instead, it acts as if `FORCE_SOURCE_DATE=1` is set implicitly: Besides metadata, also visible date representations honor `SOURCE_DATE_EPOCH`, be it `\time` or `\DTMnow`. In contrast to `pdflatex` and `xelatex`, which automatically display epoch time given by `SOURCE_DATE_EPOCH` in UTC, `lualatex` does so only if invoked with option `--utc`. Else `lualatex` honors the current time zone which may also be specified by `TZ`.

At time of this writing there seems to be a bug in `lualatex` leading to the wrong time zone when the time is start of a full hour.

Last thing: LATEX engines creating DVI/XDV and going on with `xdvipdvmx`. Of course, visible data are as with direct compilation and also file times are straightforward. To get these data right, LATEX engine must be invoked with the system variables as for direct creation of PDF files.

The only issue are meta-data. Here also they seem to be stored in the intermedate dvi/xdv files, and just passed to PDF by `xdvipdvmx`, except date and time information. These are added by `xdvipdfmx`.

So, `SOURCE_DATE_EPOCH` and `TZ` but not `FORCE_SOURCE_DATE` must be considered when invoking `xdvipdfmx` and that like. If none of these are given, the current time with local time zone are used. If `SOURCE_DATE_EPOCH` is not given, then `TZ` is honored, but if `SOURCE_DATE_EPOCH` is given, then creation time is in UTC time zone, ignoring `TZ`. So `xdvipdfmx` behaves very much like `xelatex` in direct compilation of PDF, which is clear because it is by defaults `xelatex`'s PDF backend. This behavior of `xdvipdfmx` coins time metadata even if used in conjunction with `lualatex`, which is some argument to configure direction computation of PDF files with `lualatex` in a way that it fits the other engines.

4.3 Metadata in the PDF trailer and info dictionary

The root for metadata in PDF file is the trailer dictionary. As described in [ISO20], Section 7.5.5, it contains references to several other dictionaries, the size and the *trailer identifier* with key `ID`. Among the references is that with key `Info` to the *document information dictionary* (DID) described in [ISO20], Section 14.3.3. The most important entries are `CreationDate` and `ModDate`; the rest is deprecated with PDF version 2.0. Still these entries are relevant for PDF versions 1.x. For all entries in the document information dictionary including `CreationDate` and `ModDate` there are replacements in the metadata streams described in [ISO20], Section 14.3.2. To be more precise it is the XMP metadata, which is an XML format, described in [ISO12]. Frequently, the replacements allow a set of locales. The LATEX engine writes this if in the header of the TEX file

```
\DocumentMetadata{...}
```

is specified, even if the argument list is empty, because this signifies default `xmp=true`. The result can be read through

```
pdfinfo -meta xxx.pdf
```

The keys of the document information dictionary (DID) and their replacements as XMP are given in Table 4. Note that the keys for XMP have a prefixed namespace separated by `:`. The table does not list all XMP keys specified in [ISO12], but only those which are observed as created by a LaTeX engine. Observe that the XMP keys with namespace `pdf` are not specified in [ISO12].

DID	XMP	remark
Title	dc:title	same
Author	dc:creator	same
Subject	dc:description	
Keywords	dc:subject	not pdf:Keywords as specified
—	dc:type	
—	dc:language	
—	dc:date	duplicate of CreationDate?
—	dc:format	application/pdf
—	dc:source	the LaTeX main file
Creator	xmp:CreatorTool	should be creator of tex source…
Producer	pdf:Producer	
—	pdf:PDFVersion	
CreationDate	xmp:CreateDate	
ModDate	xmp:ModifyDate	same as CreationDate?
—	xmp:MetadataDate	same as CreationDate?
Trapped	pdf:Trapped	
—	xmpMM:DocumentID	
—	xmpMM:InstanceID	

Table 4: Keys of DID and replacement by XMP

So far, the only piece of data degrading security is `dc:source`. One may think of eliminating it using package `hyperxmp`, described in [Pak23]. This is deferred because it collides with `\DocumentMetadata`.

Besides a reference to the document information dictionary, the trailer dictionary also contains the trailer identifier described in [ISO20], Section 7.5.5. It indicates that not writing the trailer identifier or corrupting it is no good idea, e.g. because it is needed for encryption or for uniquely identifying the document.

Details are given in [ISO20], Section 14.4. First, the identifier consists of two hashes, the first of which depends on the newly created but unmodified document. The notion behind is that of incremental updates described in [ISO20], Section 7.5.6. The first hash shall depend on the content of the file at creation time and on `CreateDate`. The second hash in contrast shall depend on content of the file after last incremental update and on time of last update, i.e. on `ModDate`.

In the LaTeX world incremental update is not supported so that `CreateDate` and `ModDate` coincide and so do the contents. As a consequence, the two hashes of the trailer identifier coincide as well. The last section of [ISO20], Section 14.4 seems to indicate, that the location of the PDF file may go into the trailer ID, which would make it less reproducible.

4.4 Reproducibility

The first observation is, that invocation of `xelatex latexEngines` produces different PDF output for each run. Likewise, `xelatex -no-pdf latexEngines` produces different XDV output for each run. As turns out later, this is because the creation time goes into the result.

Thus, it is plausible that, to obtain reproducibility, we invoke the engine as

```
SOURCE_DATE_EPOCH=0 FORCE_SOURCE_DATE=1 xelatex         latexEngines
SOURCE_DATE_EPOCH=0 FORCE_SOURCE_DATE=1 xelatex -no-pdf latexEngines
```

The second result is, that creating the PDF file and the XDV file that way is reproducible.

It turns out, that this compression setting does not refer to the XDV file, which seems always compressed, but solely to the PDF file. The PDF files differ mainly in the time stamp but also in some hashes which may depend on the time stamp.

Now let us experiment with `xdvipdfmx`. Even if we start with a reproducible XDV file, the PDF file created by `xdvipdfmx` changes with each invocation. This changes when also `xdvipdfmx` is invoked with fixed time.

```
SOURCE_DATE_EPOCH=0 FORCE_SOURCE_DATE=1 xdvipdfmx latexEngines
```

As mentioned above, in the distribution TEX Live current at time of this writing, the programs `dvipdfm`, `dvipdfmx` and `xdvipdfmx` are all binary identical. Nevertheless, they seem to lead to different output. Possibly, the invocation name goes into the result. To find out, we do not allow compression. It turns out that the names go into the result as the producer.

Using the package `hyperref`, one can overwrite a lot of metadata. Details are found in the manual [RO22], Section 5.10. In particular, the producer can be set `unknown`. As a result, the trailer identifier is the only remaining difference. Seemingly, the producer goes into this whether displayed or not. The trailer identifier cannot be overwritten by `hyperref`, but only in a way specific for `xelatex`:

```
\special{pdf:trailerid [
  <00112233445566778899aabbccddeeff>
  <00112233445566778899aabbccddeeff>
]}
```

makes even the XDV to PDF converter transparent.

In [RO22], Section 5.10, also the creator is found, which is `LaTeX with hyperref` independent of the LATEX engine. This shall be overwritten if there are security concerns.

As long as the tool chain and settings remain constant, invocation of LATEX engine `xelatex` and backend XDV to PDF converter specifying `SOURCE_DATE_EPOCH=0 FORCE_SOURCE_DATE=1` suffices to guarantee reproducibility. This setting refers to start of computer time history, although nothing special happened like Christs birth, creation date `1970--01--01 00:00:00Z` and if the file exist, this is the modification date. From the point of view of reproducibility, there is ok, but it is not the truth. Thus, it makes sense to overwrite this with the string `unknown`. If the file is overwritten, the same considerations apply to the modification date. Both can be overwritten with package `hyperref`.

One question remains: how does `hyperref` manipulate the metadata and in a second step, can we do this directly without using `hyperref`.

Now let us switch to the other two LATEX engines. Both write the banner information indicating above all the type of engine and the version. As checked by switching compression of, `xelatex` does not write any banner information. Again, as long as the engine does not change nor changes

its version or its distribution, the banner does not corrupt reproducibility. On the other hand, removing it would stabilize and generalize reproducibility somewhat: stabilize because the banner contains version information and thus breaks reproducibility at version change, and it breaks reproducibility when changing the engine of course. Also, privacy or security is an argument in favor of eliminating the banner. The package `hyperref` offers no way to change the banner; this can be done only in a machine specific way. The details are described in Section 4.6 below.

The same is true for the trailer identifier. Strictly speaking it need not be suppressed for reproducibility, but to make the result independent of the DVI to PDF converter as is explained in the context of `xelatex` above.

Creating DVI files with `SOURCE_DATE_EPOCH=0 FORCE_SOURCE_DATE=1` yields reproducible results. As expected, these settings are also necessary for translating DVI into PDF.

4.5 Suppressing and overwriting meta info in an engine specific way

For `lualatex` [HHHS24], Section 3.2.2 describes the variable `suppressoptionalinfo` which specifies bitwise which pieces of metadata to be suppressed. Among the `PTEX` entries, `PTEX.FullBanner` is unique as it refers to the top level document: it is the first line of output in the log file. For sake of privacy this shall be suppresed. The other `PTEX` entries, `PTEX.FileName`, `PTEX.PageNumber` and `PTEX.InfoDict` refer to embedded PDF files. The `InfoDict` has entries `Producer`, `Creator`, `CreationDate`, `ModDate` and `Trapped` and must be suppressed as well.

Also, `Creator` and `Producer` shall be suppressed. Note that suppression of `Creator` is impossible when using package `hyperref`, because it writes `LaTeX with hyperref` setting itself as creator. To be safe that this piece of information is not exposed, this meta info must be overwritten later. Metadata `Trapped` is suppressed, although there are reasons both for suppression and for leaving as is.

As described elsewhere, neither `CreationDate`, `ModDate` or `ID` shall be suppressed.

As described in [THHB24], Section 4.2.3 and 4.2.4, `pdflatex` can suppress only `PTEX` data which must be done, and `CreationDate`, `ModDate` which must not be suppressed. Finally, `xelatex` cannot suppress any metadata.

Meta info may be suppressed but also overwritten. For `lualatex` [HHHS24], Section 3.2.2 shows how to overwrite the trailer ID. In contrast, `pdflatex` has many options to overwrite metadata collected in [THHB24], Section 4.2, among those also the trailer ID. Neither of these are needed. Finally, there is no official documentation on `xelatex` for overwriting metadata but internet sources show a way to overwrite the trailer ID:

```
\special{pdf:trailerid [
  <00112233445566778899aabbccddeeff>
   <00112233445566778899aabbccddeeff>
]}
```

4.6 Security and Stability of Reproducibility

Security is here privacy. Hiding information makes attacks more difficult. Stability of reproducibility consists in stability as regards new versions of the same tools in the tool chain and the aspect of change of a tool or some other aspect of environment.

One aspect is the time zone. As indicated in Section 4.2.2, reproducible PDF files must have `CreateDate` and `ModDate` both in UTC time zone. This supports both privacy, as the true time zone of the build facility is not accessible and stability of reproducibility as a change of location of

the build facility or change of summer/winter time may not corrupt reproducibility. This feature supports international cooperation.

From Section 4.4 come the recommendation to set the following pieces of information to unknown., This is done with hyperref package ****

Creator This is uniformly `LaTeX with hyperref` as long as `hyperref` is loaded, except for `beamer` class (which loads `hyperref` implicitly) for which it is `LaTeX with Beamer class`. If `hyperref` is not loaded, the creator is TEX except for `xelatex` which shows engine and creation date. Thus, it is advisable in general for security but without `hyperref` for sake of stability of reproducibility.

Producer This is `xdvipdfmx` with version for creating DVIs and in general for `xelatex`. For creating PDF with `pdflatex` or with `lualatex`, it is something like `pdfTeX-1.40.25` or `LuaTeX-1.18.0`. This shall be hidden for sake of security and stability of reproducibility.

PTEX.Fullbanner is not written by `xelatex`, but for both `lualatex` and `pdflatex`. It can be suppressed in an engine specific way, but not through `hyperref`. The banner exposes tools, versions and distributions. Thus, it shall not be exposed for sake of security and stability of reproducibility.

All these pieces of information and a bit more are suppressed by including headerSuppressMetaPDF.tex. Observe that to that end the package `hyperref` is used whenever possible because this technique is not specific for the LATEX engine. Only banner and trailer identifier are suppressed in an engine specific way.

Besides code, `headerSuppressMetaPDF.tex` provides additional info in the comments, but for details which are specific for the individual engines see [HHHS24], Section 14.1.8 for `lualatex`, and [THHB24], Section 4.2 for `pdflatex`. Seemingly, `xelatex` is quite different from the other engines and tends to write fewer pieces of information. It uses an external XDV to PDF converter, `xdvipdfmx` by default. By placing `\special` commands in the TEX file, the user can pass information to the XDV to PDF converter also controlling meta info to some extent. Some details are given in the manual for `xdvipdfmx`, [Tea20], Section 4.1.1.

4.7 Miscellaneous

4.7.1 The command `\DocumentMetadata`

One of the subjects is the command `\DocumentMetadata` described in [MF23], Section 2. We used it as

```
\DocumentMetadata{uncompress}
```

to avoid compression of the created PDF file, mainly for debugging, but it is significant even with empty argument list, e.g. it forces `xelatex` displaying `ModDate` in addition to `CreationDate`. Another effect, which can be considered a bug is, that overwriting of meta info by command `\hypersetup` offered by package `hyperref` does not work anymore. This applies e.g. to dates but also to creator and producer. This bug does not corrupt reproducibility but security.

Another effect comes into the game because `\DocumentMetadata` has default value `xmp=true` which means that XMP metadata described in [ISO12] is written. This is an XML format and can be retrieved through

```
pdfinfo -meta xxx.pdf
```

Seemingly, this does not corrupt reproducibility but may be considered to degrade security. Partially, it is a surrogate for the document information dictionary as described in Section 4.3.

Readers who refuse using `\DocumentMetadata` and use use `xelatex` may take refuge to switching off compression via

```
\special{dvipdfmx:config z 0}
```

4.7.2 Manipulating the trailer identifier

5 To be clarified

Note that `xelatex` works as the other engines for PDF, whereas it has XDV instead of DVI as alternative format. Thus, the format DVI is ignored. See the manual.

6 References

[HHHS24] H. Hagen, H. Henkel, T. Hoekwater, and L. Scarso. *LuaTEX Reference Manual*, 2 2024. Refers to version 1.18. A copy is within the documentation of this software.

[ISO12] International Organization for Standardization (ISO). *Extensible metadata platform (XMP) specification Part 1: Data model, serialization and core properties*, 2012. withdrawn; superseeded 2019.

[ISO19] International Organization for Standardization (ISO). *Date and time — Representations for information interchange Part 1: Basic rules*, 1 edition, 2 2019.

[ISO20] ISO. *Document management – Portable document format – Part 2: PDF 2.0*, 2 edition, 12 2020.

[KB23] editor K. Berry. *The TeX Live Guide—2023*, 2 2023. A copy is within the documentation of this software.

[MF23] F. Mittelbach and U. Fischer. *The documentmetadata-support code*, 3 2023. A copy is within the documentation of this software, in fact two documents, documentmetadata-support-doc.pdf and documentmetadata-support-code.pdf which also comprises the implementation.

[Pak23] Scott Pakin. *The hyperxmp package*, 9 2023.

[RHB24] W. Robertson, K. Hosny, and K. Berry. *The XETEX reference guide*, 3 2024. Refers to version 0.999996. A copy is within the documentation of this software.

[RO22] Sebastian Rahtz and Heiko Oberdiek. *Hypertext marks in LaTeX: a manual for hyperref*, 2 2022.

[Sch22] C. Schenk. MiKTeX Manual. `https://docs.miktex.org/manual/`, 2022.

[Tea20] The Dvipdfmx Project Team. *Dvipdfmx User's Manual*, 6 2020. Version 0.12.4b.

[THHB24] Han The Thanh, H. Hagen, H. Henkel, and K. Berry. *The pdfTEX user manual*, 2 2024. For pdftex 1.40.26. A copy is within the documentation of this software.