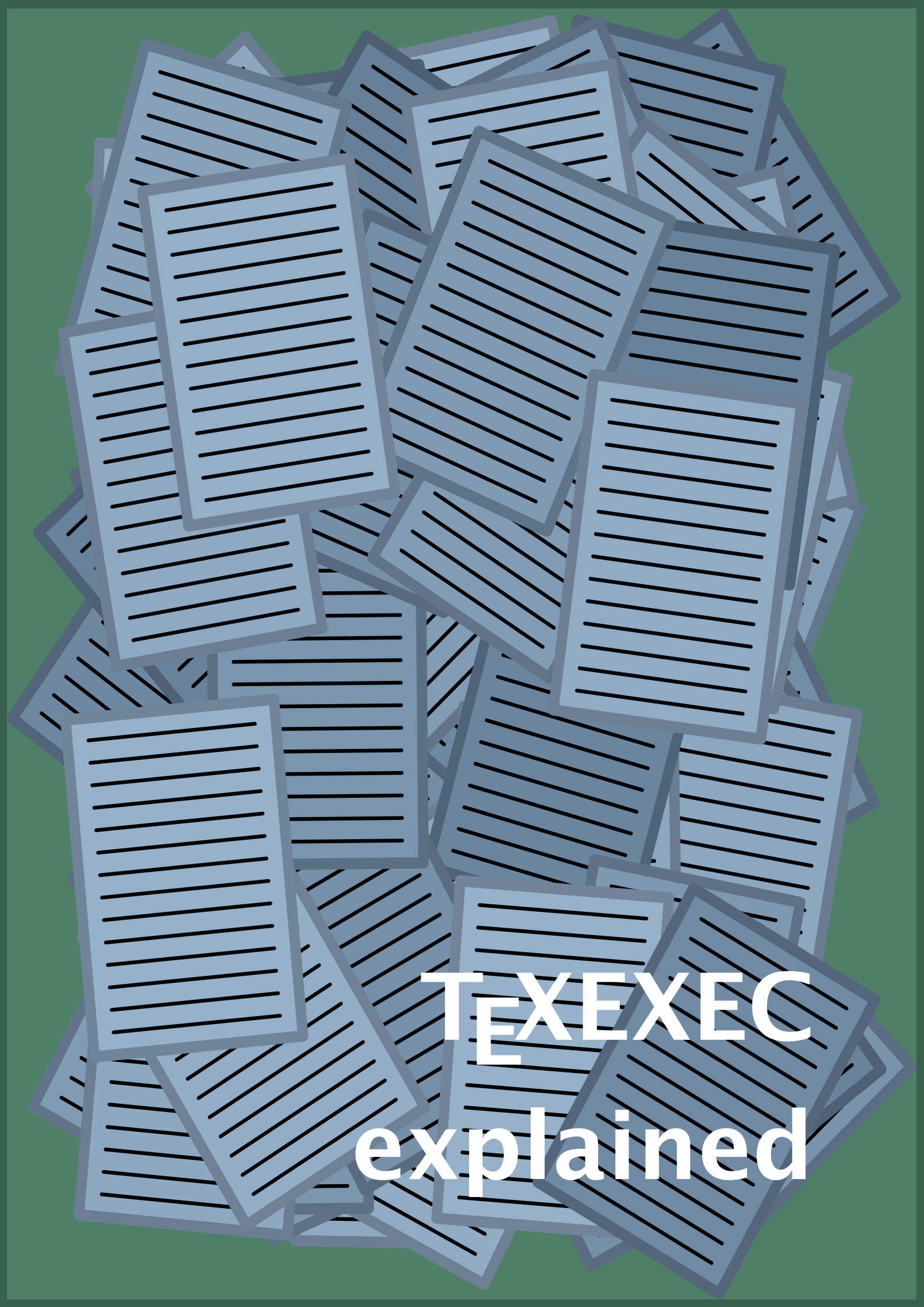

TEXEXEC
explained

Introduction

TEX the program is a batch text processer. You key in a document source file in ASCII, and TEX takes this file and interprets the TEX specific typesetting instructions embedded in the text (usually preceded by \).

When we launch TEX it first loads a so called format file, a pre–compiled collection of macro definitions, hyphenation patterns, and sometimes font metric data. In our case, we use the CONTEXT format.

Each time we install a new version of CONTEXT, we have to regenerate this format, and each time we run TEX we have to specify the format to use. To free the user from this burden of specifying and generating, we have written TEXEXEC, a PERL script that, like TEX, is not bound to specific hard- and software platform.

Just in case you wonder why the TEXEXEC logo (on this cover) looks like it is: one of the main objectives of TEXEXEC is to enable users to produce his/her pages as efficient as possible. So, TEXEXEC is dealing with pages.

Running

Each CONTEXT user interface has its own format. The next command generates two formats, one with the English interface, also defaulting to English typesetting, and one for the Dutch language:

```
texexec  --make  nl en
```

By default, the main language matches the user interface, which is normally what we expect from a Dutch (nl), English (en) or German (de) version. You can default to another main language and/or font by changing the file cont-usr.tex, but the next alternative works well too:

```
texexec  --make  --language=pl,cz,sk  --bodyfont=plr  en
```

This will generate a format file with an english user interface, while the main language is Polish (pl) and the default body font is the Polish alternative plr of the Computer Modern Roman (cmr). In addition, Czech and Slovak patterns are loaded. In a similar way you can generate a Czech version (cz and csr).

When the appropriate formats are present, we can process a file, say test.tex, and typeset it by calling TEXEXEC:

```
texexec  test
```

TEXEXEC tries to determine what interface is used. It does so by analyzing the file. Later we will see how we can force an interface.

When, instead of DVI output, we want to produce PDF code, we say:

```
texexec --pdf  test
```

Often one processing cycle is not enough. When no errors are encountered, TEXEXEC will call its relative TEXUTIL. When changes are found in the number of pages, references and/or lists, TEXEXEC will process the file again. You can prevent this by:

```
texexec  --once  test
```

or

```
texexec  --runs=2  test
```

We already mentioned PDF production, that can be forced by `--pdf`. Specific output can be forced by:

```
texexec  --output=dvips,acrobat  test
```

This means that the DVIPS special driver is used (which happens to be default anyway) combined with PDF instructions written in POSTSCRIPT.

You can force an interface on the command line by using the `--interface` switch.

```
texexec  --interface=en test
```

This specification is seldom needed because TEXEXEC has its own way of finding out what interface to use. A rather safe way of forcing an interface is adding a specification to the file:

```
% interface=en tex=pdfetex output=pdftex
```

Beware: this line should be the first line in the file! For TEX this line is simply a comment line, but TEXEXEC will read it as an instruction. The binary to be used can also be forced on the command line:

```
texexec  --tex=pdftex  --format=plain
```

Encodings

In the Eastern European Countries like Poland and Czech, the normal ASCII encodings are no longer sufficient. Although course, you can access the non–latin glyphs in a font by using for instance `\.z`, it makes more sense to use the slots from 127 and onwards. This not only permits a more convenient way of keying in a document —in those countries people use word processors that show the appropriate glyphs on the screen— but also permits proper hyphenation.

Instead of supporting all kind of (sometimes very rare and weird) encodings, the default ConTeXt distribution sticks to the most commonly used ones. More details can be found in the files prefixed by `enco-`, `lang-` and `font-`.

When using the web2c distribution, you can use a mapping mechanism to map from the document encoding to an encoding that matches the ConTeXt internals, the fonts, and the hyphenation patterns. For this reason you can inform TeX on the mapping to be applied, either on the command line, or on the first line of the file. The latter method is to be preferred, because that way the encoding used is also properly documented in the file. Both:

```
%& --translate-file=cp1250pl
```

```
% --translate=cp1250pl
```

are understood. The other way of informing TeX is:

```
texexec  --translate=il2pl  somefile
```

In both cases, users should be aware of the fact that their files are not on forehand portable!

Installation

At start–up TeXexec tries to locate the file `texexec.ini`. This file tells TeXexec where to find things and what binaries to use. When fed with `--verbose`, TeXexec will report the settings found there.

In order to run TeXexec you need to have Perl installed. Installing Perl makes sense anyway. On unix running Perl scripts is natural, but on for instance MS Windows you must often launch the scripts in an indirect way. One way is simply calling Perl: `perl texexec.pl`. The fpTeX distribution comes with a program called `runperl.exe`

that, when copied to texexec.exe, takes care of the loading. Of course you also have to copy this program to texutil.exe. In the teTeX and fpTeX distributions, the installation programs will take care of most issues involved here.

Although TeXexec tries hard to satisfy its hunger for additional information, sometimes you have to provide a little help. In the ConTeXt distribution there is a file called texexec.rme. This file should be copied into texexec.ini, unless of course such a file already exists. This file is self documented, so we only list some possible settings here:

variable	meaning	example settings
UsedInterfaces	the formats generated/used	en,nl,de,uk
UserInterface	the default user interface	en
TeXExecutable	the TeX binary to use	pdfetex
MpExecutable	the MetaPost binary to use	mpost
MpToTeXExecutable	the MetaPost to TeX converter	mpto
DviToMpExecutable	the DVI to MetaPost converter	dvitomp
TeXFormatFlag	the format introducer	&
TeXVirginFlag	the format generation switch	-ini
TeXFormatPath	fmt files	texmf/...
ConTeXtPath	sources	texmf/tex/context/base
SetupPath	cont-sys.tex cont-usr.tex	texmf/tex/base/user
TeXScriptsPath	scripts	texmf/context/perltk

An example of a setting is:

```
set  TeXExecutable  to  tex
```

You can group settings, like

```
for  tetex  set  TeXExecutable  to  pdfetex
```

Such grouped settings are only taken into account when the shell variable is set:

```
set  TeXShell  to  tetex
```

Switches

The most important switches are mentioned already, but there are some more. Next we will describe them all in more detail. All switches are mentioned in full, but when issued, you can stick to a representative first part. So, --ver is the same as --verbose.

`alone` When issued, this switch makes sure that TEXEXEC does as much on its own as possible. It can for instance be used to disable `fmtutil` being used (which can be handy when tracing installation problems).

`arrange` When you specify in a file that the resulting output should be arranged, for instance in a booklet, you must make sure that this only happens in the last pass. This switch runs CONTEXT as many times as needed to get the references right, and then applies the last, arranging, pass.

`batch` Process the file in batchmode, that is, don't pause at errors. More on batchmode can be found in the TEXBook.

`color` Turn on color mode. Color setting commands in the file being processed, take preference.

`convert=fileformat` Convert the input file before processing. Normally the conversion results in a TEX file. Currently supported input formats are `xml` and `sgml`.

`environment=listofnames` Load environment before processing the file. This option can be handy in combination with input file conversion, where no environment and/or layout settings are present in the file.

`fast` Run as fast as reasonable and possible without for instance spoiling time on including graphic data.

`figures=alternative` This option can be used to generate a document with figures. The alternatives `a`, `b`, and `c` produce different output:

a. a proof/correction sheet with additional information about the figure
b. a compact, paper saving sheet of figures
c. one figure per page, with the page clipped to the boundingbox

The next example shows that you can also pass an offset to be added to the page (only important in alternative `c`).

```
texexec  --figures=c  --paperoffset=.5cm  *.pdf *.png *.jpg
```

In the process, TEXEXEC uses TEXUTIL to provide the list of figures.

`final` Add a final run without skipping anything. This option is typically used with fast.

`format=formatfile` Normally TEXEXEC prepends the `cont-` prefix to the format filename. This switch can be used to specify the full name, like `plain`.

```
texexec  --format=plain  --program=pdftex  somefile
```

`interface=languagecode` When a file is processed, you have to use the right interface version of ConTeXt. This switch can be used to force ConTeXt to use a certain interface, like `en` (English), `nl` (Dutch), `de` (German), `uk` (Brittish) etc.

`language=languagecode` This switch sets the main hyphenation language, just in case this is not done in the file itself.

`listing` A maybe unexpected switch is the one that forces TeXexec to produce a listing. When we have a file, say `readme.now`, we can typeset a listing by invoking:

```
texexec --listing --pdf readme.now
```

In this case, the result is available in the file `texexec.pdf`, but without `--pdf`, you will get a DVI file.

`make` In order to typeset a file, TeX needs a so called format file, a pre-compiled bunch of macros. Such a format is generated when this switch is passed.

`mode=modelist` Why not typeset more than one version of one document source? The pdfTeX manuals are typeset by issuing the next few commands. Watch the renaming of the output.

```
texexec  --pdf  --mode=A4      --result=pdftex-a  pdftex-t
texexec  --pdf  --mode=letter  --result=pdftex-l  pdftex-t
texexec  --pdf  --mode=screen  --result=pdftex-s  pdftex-t
```

Here the `mode` switch tells ConTeXt to obey the mode directives in the layout specifications.

`module` Documentation of ConTeXt modules and related METAPOST and Perl modules comes down to converting the file in a suitable format and typesetting this source. This switch does is all.

`mptex` When TeX code is embedded in a METAPOST file, a sequence of pre- and postprocessing steps must take place. When set up correctly, TeXexec does this annoying job for you.

`noarrange` This switch suppresses the arrangement as set up in the source file.

`nomp` Embedding METAPOST code and calling for METAPOST to process these files run-time takes some time. This switch suppresses the METAPOST runs.

once Normally TEXEXEC keeps on processing a file till all references are sorted out. It therefore analyzes the files produced by TEXUTIL as well as METAPOST source code generated by CONTEXT. This option forces TEXEXEC to process a file only once.

output=driver One of the reasons why TEX can adapt itself so easily to new developments, is its channel to the outside world: the \special primitive. Because each DVI and PDF driver has its own set of specials, we have to tell CONTEXT explicitly what specials to use. However, normally you will load the drivers needed in the local cont-sys.tex file.

Valid drivers directives are pdftex for native PDFTEX code, dvips (the default), dvipsone and dviwindo (the oldest CONTEXT drivers), dviview (an experimental driver) and some more.

pages=pagenumberlist When PDFTEX came around, postprocessing became TEX's job. This switch tells CONTEXT to output only the pages as specified. Instead of a list, you can use the keywords odd and even.

passon=string You can pass additional switches to the TEX program used by using --passon, like for MIKTEX:

```
texexec  --passon="--src"  somefile
```

This tells MIKTEX that you want to enrich the DVI file so that editors can relate their cursor position to a location in the DVI file. Don't forget the '', because otherwise the --src will be seen as a switch to TEXEXEC.

paper=key Like the previous one, this switch is used in postprocessing. Valid keys a4a3 for A4 printed on A3, and a5a4 for A5 printed on A4. The pages are arranged as specified by the print switch.

pdf This is a shortcut for --output=pdftex. Because we like PDFTEX and its output and use it quite often, we introduced this dedicated switch.

pdfarrange Although CONTEXT is pretty well able to arrange pages itself —think of making A5 size booklets and those big 8 page composes pages meant for printing books— it is possible to let CONTEXT rearrange pages in PDF files.

```
texexec --pdfarrange --paper=a5a4 --print=up live.pdf
```

This command makes an A5 booklet out of the famous TEX live manual as produced by LATEX. You have to set up TEX rather large (a few meg's of memory) and the fonts

should be present on the system in order to let PDFTEX include them as efficient as possible. You can influence the page with the following keys:

`--paperoffset`	gap around the page
`--backspace`	inner margin
`--topspace`	top/bottom margin
`--markings`	add cutmarks
`--addempty`	add empty pages after
`--textwidth`	the original text width

The `--addempty` switch takes a list of page numbers after which to insert an empty page. This can be handy when a single sided document is processed and you want the backside of the title page to be empty.

When you manipulate a text that originally is typeset in single sided mode, passing the original text width will improve the appearance of the result a lot.

You can pass a list of files. All files are combined into one document. This enables the user to add for instance a title page.

Doing more complex things is also possible, like adding additional text to the page, when you set up a small file with layout definitions and an simple figure insertion loop.

You can also use this switch to combine PDF files into one file:

```
texexec --pdfarrange --noduplex --paper=S6 file-a file-b
```

The result is available in the file `texexec.pdf`.

`pdfselect` This switch filters pages from a file. It can be used to generate a snapshot or summary, or to isolate a page.

```
texexec  --pdfselect  --paper=S6  --selection=1,9,14  file-1
```

The result is collected in the file `texexec.pdf`. Setting the paper size is optional. Just like with the `--pdfarrange` option, you can influence a bit the layout.

`print=key` Page imposition can be done by CONTEXT, which is normally more robust that using a post processor, if only because CONTEXT knows what (colorful) content it is dealing with. The `up` key results in 2 pages per sheet doublesided, and the `down` key in 2 rotated pages per sheet doublesided. Use this switch together with the `paper` switch.

`result=filename` When producing several typeset alternatives of one file in batch, we don't want to overwrite the same file again and again. This switch specifies the name of the output file.

`runs=number` When you know in advance how many runs are needed, you can overrule the automatic number–of–runs–needed algorithm.

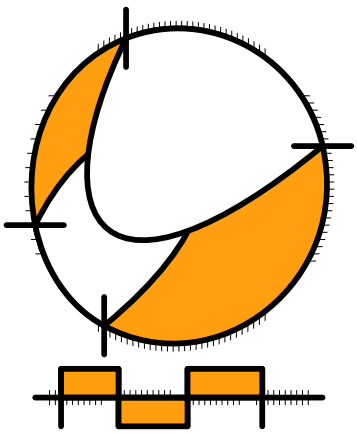
`silent` When you got tired of all the messages ConTeXt sends you, you can silence it by issuing this switch.

`tex=programname` Once a system is set up properly, you never have to use this switch. But occasionally, we want to run an experimental version, so here is the way to specify the binary.

`verbose` Some of the information that TeXExec needs, is fetched from the `texexec.ini` file. This switch tells TeXExec to write to the terminal what it found in there.

`help` This is a rather helpful switch. Try it. You can request more help, like:

```
texexec --help paper
```



PRAGMA

Advanced Document Engineering | Ridderstraat 27 | 8061GH Hasselt NL
tel: +31 (0)38 477 53 69 | email: pragma@wxs.nl | internet: www.pragma-ade.com