

Hans Hagen  
Karel H Wesseling

The authors are indebted to Wybo Dekker for his most valuable suggestions, for proofreading and for numerous acts of additional help.

# context

## TEXEXEC User's Guide

### abstract

This guide describes the uses and options of the texexec program that is available in the CONTEXT distribution. The options are invoked by calls on a command line, which are words preceded by two hyphens as in `--make`. There are options for running CONTEXT on your  $\TeX$  file to produce printable output, options to specify languages, an option to make listings of (software program) files word for word, options for conditional execution, for selecting pages to print, for printing on differently sized paper, for directing your output to a particular file, for conversion of SGML and XML to  $\TeX$ . If it is no problem to you to use a command line and to occasionally look things up in the help file or in this user's guide, you will find texexec to be a useful, even indispensable tool for CONTEXT.

### Samenvatting

Dit is een gids voor de gebruiker van het "texexec" programma dat met CONTEXT samen gebruikt wordt voor een hele serie handigheidjes en acties, en bijna onmisbaar is. Texexec leest een commando regel waarop woorden staan voorafgegaan door twee mintekens zoals in `--make`. Zo'n commando heet optie en er zijn opties voor het compileren van een CONTEXT brontekst, opties voor het specificeren van talen, voor het maken van listings, voor voorwaardelijke compilatie, voor het selectief compileren van pagina's, voor het printen op andere formaten papier of scherm, voor het in vreemde volgordes plaatsen van pagina's zoals dat nodig is voor het vouwen van boekjes uit grote vellen. U mag niet schrikken van commandoregels en moet soms dingen opzoeken in een help bestand of in dit stuk. Maar dan heeft U ook een gereedschapskist van belang in handen.

## Kick start texexec

It takes until page 40 before we actually reach the parts where your  $\TeX$  file will be processed. For the impatient we provide the ultimate quick start here, under the assumption that you are using the English language, which is, after all the "lingua franca" of modern science, and under the further assumption that your  $\TeX$  file name is `lingua.tex`.

Prepare the necessary files for  $\TeX$  and for CONTEXT to work with the English language:

```
texexec --make en
```

Process your file with CONTEXT to produce a PDF file for viewing with Acrobat or Ghostview:

```
texexec --pdf lingua
```

## What is texexec used for?

Texexec is a so-called "command line" interface for CONTEXT, directing it. CONTEXT is a typesetting computer system that extends  $\TeX$  with easy to understand operations activated by typing an `\instruction[with] [parameters]{and text}` such as `\inframed[height=fit,width=fit]{which looks boxed} [which looks boxed]`. In the next sections you will see more examples of such instructions. The instructions are programmed in  $\TeX$  by the author of CONTEXT and form — as it were — a higher level "shell" around  $\TeX$ . With CONTEXT you can do very complex things with relative ease.

To carry out the instructions placed between the normal text in your  $\TeX$  file CONTEXT processes that file to produce typeset output as a DVI or a PDF file. Through texexec

can it be controlled to do the right thing by supplying it with commands.<sup>1</sup> In addition, a series of very useful functions or ‘options’<sup>2</sup> are also built-in, as you will see later, and in directing the processing of your  $\TeX$  file it also shows substantial intelligence. Texexec has become a veritable workbench with so many options that a user's guide is needed.

But you need access to a command line. Under Windows that means back to DOS. Move your mouse pointer to the task bar and click “Start”, move up to “Programs” and follow the arrow, in the next window go up to “MS-DOS Prompt” and double click. A dark window opens showing a DOS prompt where you can type your command lines and have them executed. To close the window type `exit`. A nicer solution is to install the shareware program Windows Commander. This system gives you a command line with a simultaneous overview of your files and offers many features. Download it from [www.ghisler.com](http://www.ghisler.com). Installation is almost automatic.

Texexec started out as a stand alone program that worked as DOS software. In later years demand for the program to work on other platforms such as Linux and Mac and Windows rose and it was decided to rely on a language to develop texexec, that is almost universally available: perl. Thus, you must have perl installed on your computer. If you don't have perl go to the perl site to obtain it. Installation is simple and reliable.<sup>3</sup> Once perl is there you can run texexec and direct CONTEXT and enjoy its power and its many features.

## Many languages, fewer interfaces

When you are a  $\TeX$  user you were perhaps attracted to it because it supports so many languages with near perfect hyphenation. In this guide and texexec we call a text language “language” as you'd expect. There is also an instruction language. CONTEXT, when processing your  $\TeX$  file for typesetting, searches for type setting instructions embedded in your text. They are almost always preceded by a backslash, `\`, and easily recognized. Instructions are words or short phrases typed in English, but CONTEXT can be told to work in one of several instruction languages, called “interface”. For example in English you'd start a new chapter in your text with the command:

```
\chapter{Installing perl}
```

However, if you were Dutch but writing an English user's guide, and CONTEXT was prepared to listen to Dutch commands you might have typed:

```
\hoofdstuk{Installing perl}
```

where the Dutch ‘hoofdstuk’ and the English ‘chapter’ have the same meaning and invoke the same actions. CONTEXT, when just installed, thus has to be taught an instruction language or interface such as Czech, Dutch, or English.

To teach CONTEXT more than one interface and generate the appropriate configuration files execute the command:

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

The `make` command is the first command you use with texexec and before its execution CONTEXT will not work. Note that we are talking about the CONTEXT interfaces and not about the text language. The corresponding language's so-called “format files” for  $\TeX$

---

1. The word ‘instruction’ is used throughout when we refer to an instruction to CONTEXT proper, ‘command’ is used for instructions to texexec as given on the command line.

2. An option is a word preceded by `--` that causes texexec to change from default processing to optional processing.

3. Installation of  $\TeX$ , CONTEXT, texexec, or perl is not treated in this document.

to use are automatically generated with the interfaces. Available interfaces at present are coded as:

```
nl-Dutch,
en-English,
de-German,
cz-Czech,
ro-Romanian,
it-Italian.
```

CONTEXT is multilingual with more than 20 languages supported. This topic is treated in the CONTEXT manual. Clearly, there are more languages than interfaces. It is possible to prepare your system for several more languages simultaneously with a limited number of interfaces, and add special fonts at the same time. For example, some Slavic text languages require special fonts. With english as the instruction language you type:

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

which gives you two interfaces (English and German) and five languages (English, German, Polish, Czech, and Slovak), with `pl,cz,sk` signifying Polish, Czech and Slovak languages; `plr` is the Polish alternative for `cmr`, Knuth's Computer Modern Roman font; `csr` is the `cmr` for Czech & Slovak languages. When one or more additional languages are specified then the first after the `language=` sign becomes automatically the default language for hyphenation and for T<sub>E</sub>X. In the above example, thus, Polish.

```
--interface=languagecode
```

Your T<sub>E</sub>X file contains pure text, mixed with the necessary typeset instructions to CONTEXT. Do not mix CONTEXT instructions of different languages. Texexec, normally, can find out which interface language you used, but it can also be told which one it is with the command `--interface`. In that case CONTEXT will only react to instructions in that particular language and report errors for unrecognized ones. Specify this on the texexec command line as:

```
texexec --interface=de test
```

The accepted language codes are mentioned in section 3 on page 37. It is more appropriate, however, to specify the CONTEXT interface language in the first line of your T<sub>E</sub>X file with:

```
% interface=en
```

To establish an interface language texexec first checks your command line, then the first comment line of your text (which takes priority), then — but only if no interface language was found — it checks the language of the first several instructions (such as `\starttext`) in a final effort to establish the interface. If still unsuccessful the english interface is assumed. To avoid any error or ambiguity: just specify it.

```
--response=languagecode
```

CONTEXT not only understands several interface languages but can respond with messages in the same languages but only after it has been prepared for that. Use the `--make` command and add on the command line:

```
texexec --make --response=de
```

In your texexec and CONTEXT runs from then on you get a log file with messages in German, although the messages of T<sub>E</sub>X itself are still in English.

Even though with the `--make` command there has been a default language established it is better to specify the text language of your document with a CONTEXT instruction, if only to ascertain a proper hyphenation. To set English as the principal language for your

text you would use `\language[en]` somewhere before the start of your text. To have the hyphenation rules and exceptions available for the language of your specification, they must have been “made” with the `--make` command. Since `CONTEXT` is multilingual you can switch languages within the document. The easiest way is to isolate with curly braces the section with the deviant language and specify the local language in shorthand notation as follows: `{\de Die Schwefelmeiler f\"{u}llten Berg und Tal mit giftigen D\"{u}nsten.}` A clearer way for longer sections of deviant language text to switch to another language is offered by the `\start ... \stop` mechanism of `CONTEXT`:

```
\start \nl
In Siddeburen was een bok die machtsverhief en worteltrok. Die
bok heeft onlangs onverschrokken de wortel uit zichzelf
getrokken, waarna hij zonder ongerief zich weer in het kwadraat
verhief. Maar 't feit waardoor hij voort zal leven, is dat hij
achteraf nog even, de massa die hem huldigde met vijf
vermenigvuldigde. Een "Trijntje Fop".
\par      % or add an empty line before \stop as below:
\stop %nl
```

`--alone`

With this option `texexec` must do its work on its own and not call external functions or utilities such as `fmtutil`. Such callings may — on occasion — lead to errors if such a called system is not prepared for `CONTEXT`. This is basically an installation and error tracing option. A typical application is:

```
texexec --make --alone metafun
```

## ASCII and other text encodings

In central Europe, in countries as Poland or the Czech Republic many special characters are used perhaps with so-called diacritical marks. For example a Czech name is `Peňáz` which may be typed as `Pe\v{n}\'{a}z`. If most of the words you type are like this `TEX` quickly becomes a nuisance: the possibility for special characters is there but the facility is difficult to use, a contradiction in terms.

Originally `TEX` used 7-bit ASCII. ASCII is the American Standard Code for Information Interchange. It needs only 7 bit for each of the 128 characters that it supports. But modern computers are byte oriented and a byte has 8 bits. This leaves one bit and thus half of the possible number of characters per byte unused. Thus so-called ‘code-pages’ have been defined and each code page has an identifying number and special meanings for the other 128 positions in each byte. Switching code-pages is not a very nice process and it may lead to silly mistakes and strange characters printed on screen or on paper. A more recent solution has been to design a Unicode which uses two byte per character and thereby has possibilities for 65536 characters, special symbols and glyphs.

`CONTEXT` supports the more commonly used encodings as specified in files in the `CONTEXT` directory tree beginning with `enco-`, `lang-`, or `font-`. If you are interested please print these files. In recent `TEX` versions generically called `WEB2C` it is possible to specify a conversion or translation called mapping from the non-ASCII document encoding that you might use to a standard encoding. This mapping also takes care of fonts and hyphenation patterns. Use `texexec` to command such a mapping on its command line but preferably in the first line of your file as it is then permanently documented as non-standard:

```
% translate=cp1250pl
```

or

```
texexec --translate=cp1250pl nontest
```

nontest is a file, unlike our test file (see chapter 5), which has a non-ASCII encoding.

## CONTEXT and your test file

Assume that you have prepared a file with an ASCII oriented editor which contains a text and some typesetting instructions in CONTEXT. This file is called `test.tex`. A short but complete one is shown below:

```
% output=pdfTeX
\setupoutput[pdfTeX]
\starttext
\title{A very simple test file for \CONTEXT}
```

This file is used initially as a test file to verify that `\type{texexec}`, `perl`, and `\CONTEXT\` work as expected. We could easily add more words but there is no need.

```
\stoptext
```

Now type on a command line:

```
texexec --pdf test
```

This starts `texexec`, tells CONTEXT to work on `test.tex`, and to produce PDF output, readable and printable with the Acrobat viewer, and usually also Ghostview. `Texexec` first checks its command line (one command only in this case, except the input file) and next reads the first line of your test file. If it begins with a `%` symbol it is read and checked for commands that it might understand. In this case there is a command that the PDF output format must be generated, again. By default, however, a DVI output format is generated. DVI is a proprietary  $\TeX$  format that is device independent and can be viewed and printed with DVI viewers though not with Acrobat. The Linux operating system often comes with a DVI viewer already present. There are two other ways of overriding the DVI default and they can be used all at the same time although just one form suffices.

The `texexec` command line option for PDF output is:

```
texexec --output=pdfTeX test
```

or shorter:

```
texexec --pdf test
```

In your  $\TeX$  file you might also type a line:

```
\setupoutput[pdfTeX]
```

which was already included in the test file as shown above. We now know that one of the solutions would have been sufficient but all three together does no harm.

*Importantly, the last output specification that is found rules the form.* Thus, if you have `--pdf` on the command line but `\setupoutput[dvips]` in the setup area of your document the output will be DVI.

## Keeping CONTEXT under control

### Intelligence slows down

As mentioned, `texexec` has a certain intelligence built-in. For example, assume that you work on a very substantial text which includes a table of contents and an index of terms. These items require a certain space, perhaps even a number of pages. But beforehand the exact space required is not known. After your  $\text{\TeX}$  file is processed once, the contents and number of index terms is known. In a second run these tables can be typeset. Then, finally, in a third run all page numbers are known and can be entered in both tables. `Texexec` observes what is going on during the typesetting and has ways to determine how many `CONTEXT` runs are needed.

But assume that each run takes 1 minute, and assume that you are impatient, and further assume that you are only interested in checking that you have entered your `CONTEXT` typesetting instructions correctly. In that case a single run suffices. Type a command line as follows:

```
texexec --once test
```

or

```
texexec --runs=1 test
```

The latter command suggests that `--runs=2` is also an option, which it is.

### Finally fast

Some fast thinkers complain that `CONTEXT` takes its time when processing. If considering how much you get in return does not give sufficient relief of the symptoms then use the `--fast` option. Certain typesetting actions are quite time consuming. The inclusion of graphics files as illustrations is notorious, but the output file without the graphics does not look nice. But you are impatient and willing to pay the price. Then use this option and such time consuming actions will be skipped. A graphic will be replaced by a box outline in the proper size, so that the overall layout is not affected. Your last run, however, should contain the `--final` command to ensure a complete final result, skipping nothing, and doing an adequate number of runs automatically.

### Complexity options

Even more complex: `texexec` commands `CONTEXT` and `CONTEXT` calls  $\text{\TeX}$ . Assume that there is a reason to specify one of several  $\text{\TeX}$  text processor programs. Then specify it on the `texexec` command line:

```
texexec --program=tex --format=plain test
```

There may be a conflict with a specification for PDF output that plain  $\text{\TeX}$  cannot generate. But the conflict is resolved by `texexec`, and DVI output is produced instead. An alternative is to use the first line in your  $\text{\TeX}$  file as the command line for `texexec` (without the `--`). An example of such a pseudo command line is:

```
% interface=en program=pdfetex output=pdfetex
```

or

```
% interface=en program=pdfetex output=dvips
```

depending on your desired output format.

`--batch`

Batch processing means that `CONTEXT` or `TEX` (see page 36) does not halt to ask for user input when encountering an error. It places any errors and questions together with line numbers in the log file while continuing the processing.

`--convert=fileformat`

`CONTEXT` expects its own typesetting instructions in the text file. But `TEX` and `CONTEXT` are not the only typesetting systems. Other world wide standards are `SGML` and `XML`. With use of `texexec XML` and `SGML` mark up can be translated to genuine `CONTEXT` and `TEX`. For example you may place `XML` instructions in your text and then use the `--convert` option to translate:

```
texexec --convert=xml testxml
```

`--format=formatfile`

When compiling, `texexec` uses the `CONTEXT` format files (files beginning with `cont-`). It is possible to use other format files, when testing. This is a rather specialized option that you might not use often. An example is:

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

## Fabulous flexibility

`--color`

This turns on typesetting in color. The alternative is to use the `CONTEXT` instruction `\setupcolor[state=start]` in the setup area of your `TEX` file, before the `\starttext` instruction. Such instructions in the `TEX` file override the command line option since they are processed later. Assume, however, that you have a document in which you have emphasized portions using colors. Assume you have a black ink LaserJet on which a color output file would cause undesirable gray scales from dark to unreadably light perhaps. In that case, don't turn color processing on in your text but in a final run, before going to an outside color printer, produce a result file with the `--color` option. You may also use the `--mode` command on page 45.

`--environment=listofnames`

An environment is usually a separate file with `CONTEXT` instructions which is placed up front in your `TEX` file to specify the make up of your text pages. For example, you might make an environment file named `e-scr.tex` which contains the single `CONTEXT` instruction `\setuppapersize[S6][S6]` which would typeset your document with sizes best suited for viewing on screen. Default, your output would be for A4 paper. Normally you would place reading of the format file in the setup area of your document. This time don't, but put it on the `texexec` command line, and increase your flexibility in processing:

```
texexec --environment=e-scr test
```

Your output result file will be optimal for screen viewing. But with a different environment file it could be made optimal for A5 size paper, yet your document itself would be identical to the comma. Note: this particular effect can be achieved more easily with the `--mode=screen` command explained on page 45 but the general principle holds, and environment files may attain great complexity.

`--output=driver`

The use of this command is already explained in section 5 on page 40 where it was used to produce PDF output (with the option `--output=pdfTeX`, or its abbreviated form `--pdf`).

Further possibilities are `--output=dvips` which produces the default DVI output and need not be specified, `dvipsone`, `dviview` which is an experimental viewer, and `dviwind`.

`--result=outfile`

This option is used a number of times in this guide, for example in combination with the `--mode` option, see page 45. With it you set the name of the output file to which the result of the processing is written. Addition of the proper filename extension is automatic.

### METAPOST measures

METAPOST is a language to draw vector graphics, and its output can be Encapsulated Postscript (`.eps`) which can be incorporated in a CONTEXT document.<sup>4</sup>

METAPOST instructions can be embedded in a CONTEXT, and with a little more trouble in a LATEX document. These METAPOST instructions do not start with a backslash like T<sub>E</sub>X's and would be typeset as text instead of compiled as METAPOST graphics. So, therefore, in a CONTEXT document you must surround them with instructions such as:

```
\startuseMPgraphic
...
any number of MetaPost instructions
...
\stopuseMPgraphic
```

When CONTEXT reads this file and meets the METAPOST code it writes the code to a file. There are now two possibilities or options, the direct option and the indirect option. In the direct option CONTEXT calls METAPOST, has the graphic made, and upon return places the graphic in the document. This process is repeated for any METAPOST graphic it discovers. Thus, the option is relatively slow, yet it is also the default option. In the indirect option, the METAPOST is similarly extracted but not compiled until after the CONTEXT run finishes. Texexec detects that METAPOST files have been generated and calls METAPOST to produce the graphics which are placed in yet another file. (This processing can be blocked with the texexec command `--nomp`.) When not blocked texexec calls CONTEXT a second time when the prepared graphics is included. The indirect option is not default, but it can be forced with the texexec command `--automp`. Thus, when you have embedded METAPOST in your document with many single graphics use the fast indirect option as follows:

```
texexec --pdf --automp --result=oneforal metatest
```

### Passing the word

`--passon=string`

Not only texexec uses options but so do some T<sub>E</sub>X implementations. M<sub>I</sub>K<sub>T</sub>E<sub>X</sub>, for example, uses the option `--src` so that editors can set their cursor position to a location in the typeset DVI file. You may pass this option to M<sub>I</sub>K<sub>T</sub>E<sub>X</sub> as follows:

```
texexec --passon="--src" mikttest
```

The quotation marks are mandatory.

---

4. See: Fabrice Popineau: Practical METAPOST in this issue.



## Optionally list

`--help`

This option produces a rather terse list of all `texexec` options with some further comment which can be useful for experienced users or when you remember that a function is available but don't recall the magic word exactly:

```
texexec --help
```

This produces a list of options that on a Windows systems flashes by so fast that there is no time to read. Use DOS commands to get some control over this process. For example, use the DOS redirect symbol `>` to put the list to a file `help.txt` or use the `| more` command to get it in pages. More help on a particular option can be evoked with:

```
texexec --help pdfselect > helpsel.txt
```

which produces a short list of sub-options for `pdfselect` and places it in the file `helpsel.txt`. There is no extra help on help.

`--listing`

This useful option produces a typewriter-look output of your file as it is shown in your text editor. You get a DVI file unless `--pdf` is placed *after* the `--listing` option. The file to be listed must be given with its extension, or the action is cancelled. Listing is not limited to `CONTEXT` files with extension `.tex` and the files do not have to contain any `CONTEXT` instruction, and when they do they are shown verbatim:

```
texexec --listing --pdf detect.mod
```

produces a file `texexec.pdf`. You may specify your desired output file name with the result option:

```
texexec --listing --pdf --result=testlist test.tex
```

produces the output and its log file as `testlist.pdf` and `testlist.log`. It is not wise to name this listed file `test.pdf` since its normally typeset output would also have that name and overwrite your listed file.

Naturally, you may apply `--listing` to `TEX`, `perl`, or `METAPOST` software program files. By commanding `--pretty` on these files you invoke “pretty” printing, meaning that language items become highlighted in the program text, even highlighted in color if you specify `--color` as well:

```
texexec --listing --pretty --color --pdf softwa.pl
```

with the resultant output in `softwa.pdf`. Finally, `--backspace` and `--topspace` may be used as shown on page 47.

`--module`

`CONTEXT` as a software program is written in the language `TEX` designed by Donald E Knuth many years ago, and it was set up in modules. Some of these modules can be found in subdirectories such as, for example:

```
c:\4tex5.0\texmf\tex\context\base\
```

unless your `CONTEXT` system was not placed in `c:\4tex5.0\` but somewhere else. Now go the `\base\` subdirectory and then type on your command line:

```
texexec --module --mode=color --pdf colo-ini.tex
```

The result of this processing is found in `colo-ini.pdf` and in color it allows you to view (and print) a `CONTEXT` module and how it is programmed. You may like to view this one even as a user, non-programmer.

`--silent` and `-verbose`

Texexec produces a so-called 'log' file in which it places phrases that express what it is doing, problems that it encounters, and errors that may occur. The same information is also shown in a text box on screen but this often passes by so fast that it cannot be understood. The logging of `CONTEXT` can be normal, which is the default, or it can be switched to verbose thereby producing log files which also contain information possibly loaded from the `texexec.ini` file, or to silent which produces a smaller log file. This is done as follows:

```
texexec --verbose test
```

or

```
texexec --silent test
```

For the smaller size  $\TeX$  files the differences between these log files is insignificant.

## Those powerful modes

With mode is not meant that which occurs most frequently or which is fashionable but *modus operandi*, a way to operate, a manner of doing something. Mode gives you power. Please read on.

`--mode=mysize`

Default, `CONTEXT` prepares your typeset document for printing on A4 paper, but other paper sizes can be accommodated simply with a `\setup...` instruction such as `\setuppapersize[S6][S6]` (`S6` is just a code), which prepares your document for display on a computer screen. If `\setuppapersize[ ][ ]` is absent in your document you may specify a "paper" size as a `texexec` command as follows:

```
texexec --mode=screen test
```

which produces the desired screen sized output. But note that a `\setuppapersize[ ][ ]` instruction in your  $\TeX$  file has precedence since it will be read later. Given the `--mode` option it is now simple to produce two or more differently sized output files:

```
texexec --mode=screen --result=test-scr test
texexec --mode=A4 --result=test-a4p test
```

which produces the files `test-scr.dvi` and `test-a4p.dvi`, unless you also added the `--pdf` option in which case you obtain `.pdf` files. Predefined modes are `A4`, `letter`, `legal`, `color` and `screen`. As far as European paper size are concerned, all sizes from `A0` down to `A9` are options.

Once you have set the mode you can use the `\doifmode{truetest}{true-instructions and/or text}` instruction in your text to accomplish special things, as follows:

```
\doifmode{screen}{\setupcolor[state=start]}
```

which seems useful if you have a color screen display and a black ink printer. The same can be achieved with the `--color` option and with `--mode=color`.

`--mode=mymode`

The `--mode` option, however, is more powerful than this since you can define your own mode by specifying one in your text, as follows:

```
\startmode[myspecial]
  \environment e-special
\stopmode
```

With:

```
texexec --mode=myspecial test
```

CONTEXT, when examining your text file and seeing a `\startmode` instruction, will compare bracketed names and execute the instructions between `\startmode[]` and `\stopmode` only when the names match. Thus, you now have the option to conditionally have CONTEXT execute instructions to your taste. It is not limited to loading an environment, as shown above, which can be done equally conveniently with the `--environment` option, see page 42, and it is not limited to one single mode but you may specify many modes as a comma separated list:

```
texexec --mode=myspecial,mymode,screen --pdf test
```

Similarly, you may put a conditional CONTEXT instruction in your text as follows:

```
\doifmode{myspecial}{\page}
```

which forces a new page at that point if and only if the mode is `myspecial`. Any series of instructions and text to be printed can thus be made conditional. For an IF there is an ELSE. You may also have in your text:

```
\doifnotmode{myspecial}{\page}
```

## Weird arrangement of pages

```
--pages=listofpagenumbers
```

For double sided printing on printers that do not support this feature automatically the odd numbered pages should be printed first, then the even numbered pages. This can be achieved by making two output files and printing them one after the other with this option as follows:

```
texexec --pdf --pages=odd --result=odtest test
texexec --pdf --pages=even --result=evtest test
```

This effect, however, can more easily be achieved by using Acrobat as a printer controller.

You may also specify a comma-separated list of page numbers such as `5,2,99,125`. The specification list can be in any order but the resultant output is in increasing order of page number. A range of pages can be given as `2:6`. Finally, both forms can be mixed at will, thus `5,7:17` is also valid.

```
--noarrange
```

CONTEXT has many possibilities to arrange pages on a sheet of paper. This process is called “page imposition” and often quite complex. It is, for example, possible to arrange A5 pages on A4 sheets in such a way that after printing and folding they form a booklet. The following CONTEXT instructions:<sup>5</sup>

```
\setuppapersize      [A5] [A4]
\setuparranging       [2UP,rotated,doublesided]
\setuppagenumbering  [alternative=doublesided]
\setuplayout         [margin=0pt,width=fit]
```

---

5. This example is taken from a CONTEXT manual.

```
\starttext
....
\stoptext
```

produce such a booklet. Such arranging is something to delay to the final run. For initial runs the arranging can be suppressed with the `--noarrange` option so that the result can be more easily viewed on screen, with pages in normal order. Similar arranging can be achieved by passing texexec the `--pdfarrange` option, see page 47. With the instructions above, however, typesetting is done with specified font sizes. Thus, if you specified a bodyfont at size 10pt your text would come out rather large for an A5 sized paper. To achieve an effect more similar to `--pdfarrange`, which photographically reduces the size of a page with the type size, you should specify `\setupbodyfont[7pt]`. The command `--pdfarrange` performs no typesetting.

`--arrange`

This option assures that CONTEXT arranges pages on sheets in an order proper for folding to a booklet only in the last pass when everything else has been properly set, such as referenced page numbers. For example, you may have specified in your T<sub>E</sub>X file that CONTEXT should arrange output as two A5-sized pages per A4-sized sheet printed double sided. For such booklets to fold properly the order of the pages has to change from normally increasing to weird.

## Shuffling PDF pages

### Arrange with `--pdfarrange`

Once a PDF file has resulted from CONTEXT processing, the pages can be rearranged in that file in such a way that the rearranged file can be printed as a booklet. The next command, for example, rearranges the pages of the well known T<sub>E</sub>XLive manual. Note that the command works on a .pdf, not a .tex file, and the extension must be specified:

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

With `--pdfarrange` and `--paper=a5a4` the pages are reduced in size and the font size with it. This option works as if you did a photographic size reduction. Without `--print=up` the result of the above is A5 sized pages nicely centered on A4 sized paper sheets, one per sheet. With `--print=up` you get two A5 sized pages per sheet ready for double sided printing which Acrobat can do for you. The output can be found in file `texexec.pdf`. The output can be directed to a filename of your choice with the `--result` option. The pdf extension is automatic and does not have to be given.

The `--pdfarrange` option can be combined with a number of other options to fine tune positioning of the pages on the sheets:

```
--backspace   inner margin on paper sheet
--paperoffset  room left over at paper border
--textwidth   width of original text page
--topspace    top and bottom margin on sheet
```

They have to be specified in a dimension known to T<sub>E</sub>X such as pt, cm, or in and values can be positive or negative, for example: `--paperoffset=-1cm` is valid. It reduces white spaces at the border which leaves room for a wider and higher printed text page. These options require some experimenting and it is suggested that once a pleasing set is found you save them in a file for future referencing.

But this is not all. With `--addempty` you may add an empty page after a series of specified page numbers in the original numbering scheme. For example `--addempty=3,9,55`

is valid and adds an empty page after page 3, page 9 and page 55. The page numbers do not change. These pages may appear on sheets numbered 1 and 17, for example. Single sided printing is commanded with `--noduplex`, and cut marks may be added with `--markings` on the command line.

`--paper=keyname`

This option and the `--print` and `--pdfarrange` option act together to do things similar to what can be achieved as explained under `--noarrange` on page 46. Paper options are `a5a4` and `a4a3`, the former for A5 size pages printed on A4 sheets, the latter for A4 sized pages printed on A3 sheets. Any other European paper size combination can be used from A0 down to A9.

`--print=keyname`

This option and the `--paper` and `--pdfarrange` option act together to do things similar to what can be achieved as explained under `--noarrange` on page 46. For `--print=up` you get two pages per sheet, doublesided, whereas `--print=down` causes two 90 degree rotated pages per sheet, doublesided.

### Combine with `--pdfcombine`

This option allows you to print several pages on one sheet. If your CONTEXT setup specified printing on A4 paper and the paper format to print on is also A4, the size of each page is reduced or there would not fit more than one page on a sheet. Specify the arrangement and number of pages on one sheet with `--combination=2*2` for example. The number of columns in a row comes first, the number rows of pages comes second. Although `--combination=1*2` or `2*1` are certainly allowed they leave much empty space on the sheet with room to fit two more pages. The `--print=up` option (see page 48) does not work here. Thus, combinations from `2*2` onwards are useful. With further compression, however, the print quickly becomes unreadable.

The `--paperoffset` may help in keeping combined printed pages readable. The paper offset specifies a rim of white space around the printed sheet. The smaller the rim the more space is available for printing. Thus `--paperoffset=-1cm` compared to the default `+1cm` helps.

### Copy with `--pdfcopy`

On photocopiers you can make reduced and also enlarged copies of originals. You only have to make sure that the copy fits the paper size and if necessary adjust the position of your original on the glass plate. Such copying is available with `texexec`:

```
texexec --pdfcopy --scale=.707 --result=test07 test
```

produces a  $\sqrt{0.5}$  sized photocopy in file `test07.pdf` neatly centered on the page, and note that keeping pages parallel on the sheet is automatic unlike with most photocopiers.

```
texexec --pdfcopy --scale=1.414 --result=test14 test
```

enlarges the printed area with, at this scale, sections falling off the paper. The remaining text is upper right adjusted on the sheet.

```
texexec --pdfcopy --scale=1.5 --paperoffset=-10cm test
```

enlarges your page 50% and removes any empty borders but the page is still right adjusted on the sheet and no option is available to preferentially keep another area of the page on the sheet.

**Select with** `--pdfselect`

This useful option lets you cut one or more pages from a `.pdf` file and collect them to another file. Use it in combination with `--selection=listofpagenumbers`. It is even possible to place the selected pages on a different page size by using the `--paper=keyname` option. Note that the option works directly on any `.pdf` file but not on the `.tex` file which possibly was the source of the `.pdf` file. Here follow four examples of command lines:

```
texexec --pdfselect --selection=1,2,5 test
texexec --pdfselect --paper=S6 --selection=1,2,5 test
texexec --pdfselect --selection=1:5 test
texexec --pdfselect --selection=1,2,5:11,17 --result=subrange test
```

The first command line cuts pages 1, 2 and 5 from the `test.pdf` original and places them (by default) in `texexec.pdf` with the original page numbers retained. The second command line cuts the requested pages from `test.pdf` and places them resized to fit and left adjusted on screen sized “paper” in `texexec.pdf`. You may make reduced size booklets by specifying `--paper=A6`, and enlarged ones by specifying `--paper=A2`, which is probably beyond the possibilities of your printer. The third command line cuts pages from 1 to 5 inclusive. The fourth command line cuts pages in more complex specification to direct its output to `subrange.pdf`.

Thus, whenever in the future, you'd like to extract a table or a figure or the summary of a paper you have written, cut the appropriate page(s) from your electronic PDF type document and have them printed, just like you used to do by selectively photocopying such pages from the printed document.

## Option

### a

addempty 47  
alone 39  
arrange 47  
automp 43

### b

batch 42  
bodyfont 38

### c

color 42  
convert 42

### e

environment 42

### f

fast 41  
final 41  
format 41, 42

### h

help 44

### i

interface 38

### l

language 38  
listing 44

### m

make 37  
markings 47  
mode 45  
module 44

### n

noarrange 46  
noduplex 47  
nomp 43

### o

once 41  
output 42

### p

pages 46  
paper 48  
passon 43  
pdfarrange 47  
pdfcombine 48  
pdfcopy 48  
pdfselect 49  
positioning paper 47  
print 48  
program 41, 42

### r

response 38  
result 43  
runs 41

### s

scale 48  
silent 45

### t

translate 39

### v

verbose 45

## Index

### a

Addempty option 47  
 Alone option 39  
 Arrange option 47  
 ASCII and other encodings 39  
 Automp option 43

### b

Batch option 42  
 Bodyfont option 38

### c

Chapter  
 ASCII and other encodings 39  
 kick start 36  
 languages and interfaces 37  
 mode option 45  
 optionally list 44  
 PDF options 47  
 run a test file 40  
 run control 41  
 uses 36  
 weird arrangements 46  
 Color option 42  
 Command 36  
 Command line 37  
 CONTEXT  
 instructions 36  
 interface language 37  
 multilingual 38  
 shell 36  
 special font embedding 38  
 text language 37  
 Convert option 42

### e

Environment option 42

### f

Fast option 41  
 Final option 41  
 Format option 41, 42

### h

Help option 44

### i

Instruction 36  
 Interface option 38

### k

Kick start 36

### l

Language option 38  
 Languages and interfaces 37  
 Listing option 44

### m

Make option 37  
 Markings option 47  
 METAPost embedded 43  
 Mode option 45  
 Module option 44  
 MS-DOS Prompt 37

### n

Noarrange option 46  
 Noduplex option 47  
 Nomp option 43

### o

Once option 41  
 Options to list 44  
 Output option 42

### p

Pages option 46  
 Paper option 48  
 Passon option 43  
 Pdfarrange option 47  
 Pdfcombine option 48  
 Pdfcopy option 48  
 PDF options 47  
 PDF output file 40  
 Pdfselect option 49  
 perl 36  
 Positioning options 47  
 Print option 48  
 Program option 41, 42

### r

Response option 38  
 Result option 43  
 Run CONTEXT 40  
 Run a test file 40  
 Run control 41  
 Runs option 41



**s**

Scale option 48  
Silent option 45

**t**

Test file 40  
texexec  
  command 36  
  functions 36  
  interface 36  
  perl 37  
  uses 36

Translate option 39

**u**

Uses 36

**v**

Verbose option 45

**w**

Weird arrangements 46  
Windows Commander 37