mk a TeX and LaTeX maker

doc generated from the script with gendoc — bash script, version=3.05

Synopsis

```
mk [options] [file]
```

mk-specific options:

`-C, --Clean`	Remove all unnecessary files generated by latex and bibtex
`-c, --clean`	Same, except for the pdf or postscript files
`--ps`	Generate postscript version of document (default: pdf)
`-e, --edit=STRING`	use STRING as the file to be edited

vpp-related options:

`-b, --batch=STRING`	run in batch using STRING for print command
`-p, --printer=STRING`	print to printer named STRING
`-d, --doublesided`	printer is double sided
`--[no]view`	do [not] view the document
`--[no]print`	do [not] offer printing interaction

General options:

`-r, --rc=STRING`	use STRING as an rc file, instead of `~/.mkrc`.
`--norc`	don't read the `~/.mkrc` file
`-V, --version`	print version and exit
`-v, --[no]verbose`	be [not] verbose (quiet is the default)
`-h, --help`	print this help and exit
`-H, --Help`	print full documentation via less and exit

Defaults:

```
--print --view --noverbose main
```

Required other programs:

- *bash* >=4.00
- *vpp* >=3.00 (CTAN)
- *texi2dvi* >= 1.152 (CTAN or *texinfo* package)
- *tex* and friends (CTAN or *texlive* package)

Description

mk is a Bash script that, in close collaboration with vpp (short for View and Print PDF/PostScript), is helpful in the cyclic process of editing, compiling, viewing, and printing a latex, xelatex, or plain tex document. Essentially, **mk** uses texi2dvi for compilation, vpp for viewing and printing.

Having an existing LaTeX document, say `main.tex` (see the section *Locating the source*

for the creation of new documents and for other extensions than `.tex`), you run **mk** by typing:

```
$ mk main
```

or, since *main* happens to be mk's default filename:

```
$ mk
```

Now, if `main.tex` is a valid LaTeX source, **mk** compiles it, including any table of contents, indices, bibliography references, included files, and so on, and *vpp* takes over and displays the resulting *PDF* or, with the `--ps` option, *PostScript* output. When you leave the viewer you will see a prompt:

```
vpp command (h for help):
```

If you are satisfied with the displayed output, you can now decide to print all or part of your document (see the section *Page selection*), or you can simply quit by typing 'q'. On the other hand, if you decide that you want to change the source and have another try, you can edit the source by typing 'e' to get back to **mk** and (re)edit your source. After saving your work and leaving your editor, another compilation and display cycle will be performed, based on the new source.

Essentially, **mk** uses *texi2dvi* for compilation. *texi2dvi* always runs TeX at least once, even though this may be unnecessary. Therefore, TeX will be run with the `--recorder` option, which reports all the target's dependencies in a .fls file. In every cycle, **mk** analyzes the .fls and the tex and bibtex .log files to see if a compilation is needed. When errors have occurred, **mk** uses the log files to find out which file has to be edited, and at which line. This can also be an `\input`ed file, a style file, or any other file on which the target depend. However, files in the TEXMFMAIN tree are excluded.

Editor

vpp uses the contents of environment variable EDITOR to find your editer. If that variable is empty, *vim* is used. Note that your editor should not fork off your shell, so if you specify *gvim*, for example, specify it with the option `--force`.

Page selection

As said in the introduction, after a successful compilation and display of the resulting *PDF* or *PostScript* output, the user is prompted with:

```
vpp command (h for help):
```

on typing 'h' *vpp* displays examples of possible commands:

```
Examples of print commands:
    5       to print page 5
    5-      to print pages 5 through the end
    5-7     to print pages 5, 6 and 7
    -7      to print the first 7 pages
    5-7,19- to print pages 5, 6, 7 and 19 through the end
    a       to print the whole document
    -       to print the whole document
    a x3    to print 3 copies of the document
```

```
    x3      the same
    5 x3    to print 3 copies of page 5
    t       print the whole document two sided
    t 2-    print two sided starting at page 2
    b       to print the whole document as an a5 size booklet
    b -12   to print the first 12 pages as an a5 size booklet
Other commands:
    e       (if called by mk) edit the tex source and rerun mk
    c       (if called by mk) rerun mk
    v       (re)view the ps/pdf file
    oxyz    send pdf output to file xyz.pdf instead of printer
    pxyz    print to printer xyz
    dx      tell vpp printer is doublesided (x=t) or singlesided (x=f)
    h       display this help
    ?       display this help
    q       quit
```

With these examples, no further explanation should be necessary, except that, when twosided (`t`) or booklet (`b`) printing is selected for a single-sided printer, printing will be performed in two shifts, one for the front side and one for the backside. Between the shifts, another prompt appears:

```
printer ready? then turn stack and type return
```

You will have to arrange your printer such that, with the printed sides up, the first page printed will be at the bottom of the stack, and the last page printed will be on top. Normally you will then have your output come out the back of your printer. 'Turn the stack' then means: rotate it over the long side of the paper and feed it back into the printer for the other side to be printed.

For further information on *vpp*, look in its manpage by typing

```
$ vpp --help
```

or read the *vpp* documentation.

Locating the source

mk locates the LaTeX source in several steps: (here the source extension `.tex` is supposed, but `.ltx`, `.drv` and `.dtx` will also be tried)

- If you supply no arguments, the file `main.tex` in the current directoryis assumed.
- If you supply an argument (say *myfile*), **mk** adds a `.tex` extension if it isn't there and looks for `myfile.tex` in the current directory.
- If `myfile.tex` is not found in the current directory, **mk** looks in the 'alternate directory' (say `/Documents`) if you have defined one (see the section 'RC files').
- If the source was not found in `/Documents`, **mk** thinks that you may have a subdirectory *myfile* in `/Documents` where the source may live under the name `main.tex`
- If that file is not there, **mk** now concludes that the source does not yet exist and reports this, telling at the same time which files have been tried.
- Finally, if all the above did not lead to a source file, **mk** dies.

The TeX format to be used

mk will try to find out what TeX format is is needed to compile the source. The most straightforward way to tell **mk** what format to use is to insert a starting comment line which starts with `%!`, followed by the name of the tex engine to use; for example:

```
%!xelatex
```

If no such line is found, **mk** looks for a `\usepackage{fontspec}` or `\RequirePackage{fontspec}` and, if found, uses *xelatex*. The `\usepackage` may have options, but they must be on the same line.

If still no decision could be made, **mk** looks for `\documentclass` and chooses *pdflatex* or, with the `--ps` option, *latex*

Finally, if no match was found, *pdftex* is assumed or, with the `--ps` option, `tex`.

Options

mk comes with several options. Before evaluating any options, **mk** will try to read a system rc file, a user rc file, and, finally an rc file in the current directory. The default values for most options can be set in these files. See the section 'RC files' for more information.

You can also set option defaults in an alias. For example:

```
$ alias mk='mk -noverbose'
```

`--help`
: Prints help information and lets you type 'm' to display the complete man page or anything else to quit.

`--version`
: Prints name and version and then quits.

`--quiet`
: Suppresses messages about the progress **mk** is making. This is the default.

`--rc=file`
: execute the specified *file* before processing. The contents of the *file* may override options specified before the `--rc` option, therefore it is a good idea to have the habit of specifying the `--rc` option first.

`--norc`
: do not read the `~/.mkrc` file, even if it exists.

`--batch=string`
: Prevents the `--print` option to interrogate the user about pages to be printed. Instead the document is printed according to the mandatory *string*. Also sets viewing off. Thus the command

```
mk --batch '2-3 x3' test
```

prints 3 copies of pages 2 and 3 of `test.tex`, without viewing.

`--clean`
: Clean up (remove) all unnecessary files generated by *latex* and *bibtex* except for the *PDF* or *PostScript* files.

`--Clean`
: Clean up (remove) all unnecessary files generated by LaTeX and *bibtex*

including the *PDF* or *PostScript* files.

`--print`
: Present the print prompt. This is the default. This option is normally used to suppress the print prompt, for example when using **mk** from other scripts that generate LaTeX documents that have only to be displayed or stored without even being displayed.

`--ps`
: Generate *PostScript* version of document. The default is to generate a *PDF* document.

`--view`
: Run the file viewer. This is the default. This option is normally used to suppress starting the viewer, for example when using **mk** from other scripts that generate LaTeX documents that have only to be printed.

`--edit=file`
: Normally, **mk** lets you edit the main source file, but here you can specify another file to be edited instead. This is useful, for example, if you are are fixing a style file or another input file.

RC file and customization

Unless the option `--norc` has been used, the file `~/.mkrc` will be sourced, if it exists, before reading the command line options.

You can use this rc file to set the default values for the options, by setting the global shell variable named after the long version of the options. For example:

```
verbose=true # run in verbose mode
```

So if you usually like **mk** to work in verbose mode, you can indicate so in your rc file and change your mind in some cases by using the `--noverbose` option.

Other variables, not having a corresponding command line option, that can be set in the rc files, and their default values, are:

`extraoptions=`
: adds one or more extra options to the *tex* (*latex, xelatex* et cetera) command. Example: `extraoptions='-shell-escape␣-quiet'`

`othercleans=`
: can be set to a file regular expression; in the cleaning operation, caused by the `--clean` option, this variable wil be eval'ed, and the resulting files will be removed. This is useful, for example, when the `gnuplottex` package is used; this package generates intermediate files named `$base-gnuplottex-fig*`, where the variable `$base` contains the basename (without extension) of your tex source file. So after adding:
  ```
  othercleans='${base}-gnuplottex-fig*'
  ```
  to your `./mkrc` file, the cleaning operation will get rid of these files, too.

`texi2dviquiet=false`
: Normally, in verbose mode, you also see the complete tex log output, because texi2dvi will be verbose, too. This obscures most other output. You can keep texi2dvi quiet in verbose mode by setting this variable to true:

```
texi2dviquiet=true
```

`skip_pattern=`
: can be set to a file wild card pattern. Files matching this pattern on which the *(la)tex* source file may depend will not be checked for changes. For example, if you use a write-protected TeX-tree in the directory mytextree it makes sense to set `skip_pattern=mytextree` unless you set `skip_pattern` explicitly, it will be set to match the TEXMFMAIN tree.

`altdir=`
: If `altdir` is non-empty and a file to be compiled does not exist in the current directory, it will be given another try after prefixing it with the contents of `altdir`. So if you like to have your LaTeX file in `/Documents/myfile.tex` you can set `altdir` to `/Documents` and run **mk** from any directory with:

```
$ mk myfile
```

However, a directory like `/Documents` does not make much sense if many of your LaTeX documents do not consist of a single file, but are constituted of an ensemble of a main LaTeX source and one or more `\include`d and `\input`ed files such as graphics. You will then probably prefer to have a subdirectory in `/Documents` for every LaTeX document. Therefore, if **mk** does not find `myfile.tex` in the alternate directory, it will assume that *myfile* is a subdirectory with a main LaTeX source in it, called `main.tex`.

`default=main`
: This is the default for the base name of your LaTeX document.

`warnings_to_skip=()`
: Warnings appearing in the log file will be reported after a successful run. Warnings matching any of the rexgexp's in this array will be skipped, however. For example, one could enter here:

```
warnings_to_skip=(
  'Package hyperref Warning: Token not allowed in a PDFDocEncoded string
  'Package array Warning: Column [XY] is already defined on '
)
```

The first message appears when the *hyperref* package is used and section titles contain LaTeX-commands, the second message appears when the *ctable* package is used, because it intentionally changes the X and Y column specifiers.

TeXWorks and mk

mk can be used for one-click typesetting:

- edit -> preferences -> Typesetting
- add a new tool 'mk' and give it three parameters:

```
--noview
--noprint

$basename
```

- Deselect "Auto-hide output panel unless errors occur"

mk runs pdflatex with the `--synctex=1` option, so you will be able to jump betwee source and pdf-ouput.

Bugs

Currently, **mk** is only available for Linux. It depends on *texi2dvi*. Spaces in the basename of TeX sources are not allowed (neither does the texi2dvi script on which **mk** is based.)

Changes

Changes with respect to version 3.01:

- let vpp save its output in the working directory
- removed crappy version testing on texi2dvi - use texi2dvi from the *texinfo* package

Author and copyright

`Author` Wybo Dekker
`Email` wybo@dekkerdocumenten.nl

Functions used:

findsource

```
parameters      the script's first and only argument, maybe nil
description     find the file to be compiled; if the argument is:
                   nil:     main.{tex,ltx,drv.dtx}
                   xxx:     {xxx,xxx/main,$altdir/xxx/main}.{tex,ltx,drv.dtx}
                   xxx.ext: {.,$altdir}/xxx.ext
globals set:       base dir ext fullpath
```

globals used: IFS PWD altdir default

`returns:` 0 on succes, 1 otherwise

run

```
parameters:     command to be run, with its parameters
description:    Run a command; show what's run if $verbose.
                If the command exits with 1, that's considered an error,
                other values have a special meaning and are supposed to be a
                success
globals  set:
globals used:       Com Err Lin Nor
returns:        the exit value of the command
```

settexdeps

```
parameters:     -
description:    Scans $base.fls for tex dependencies and places those in the
                array texdeps. Any dependencies with future timestamps are
                touched in order to prevent mk from looping.
globals  set:       texdeps
globals used:       base PWD skip_pattern
returns:        0
```

---

setbibdeps

```
parameters:     -
description:    Scan aux file ($base.aux) for bib-files needed and places those
                in the array bibdeps. Any dependencies with future timestamps are
                touched in order to prevent mk from looping.
globals  set:       bibdeps
globals used:       base
returns:        -
```

---

compile

```
parameters:     -
description:    runs the command in texcommand
globals  set:   -
globals used:       base bibdeps target texcommand texdeps
returns:        0 on success, else 1
```

---

show_error_and_edit

```
parameters:     -
description:    Show compilation errors via texlog_extract and (unless edit is empty)
                edit the source file where the error is in, opening the editor at the line
                where the error is..
globals  set:       IFS
globals used:       Lin base bibdeps edit target warnings_to_skip
returns:        0
```

---

edit

```
parameters:     1 file to be edited; if empty: contents of edit variable is used
                2 line number where edit should start; if empty, use 1
                3 true if an error was detected in the file and user must decide
```

	if file shall be edited
description:	Start the user's editor to edit the file in argument 1; if the call was induced by the detection of an error in that file, the user will be asked if he want to edit the file, or to quit.
globals set:	-
globals used:	edit
returns:	1 if the file was edited, else 0

---

handle_options

parameters:	uses script's arguments
description:	Handles the options
globals set:	Clean batch clean doublesided dvips edit input norc print printer ps rc verbose view
globals used:	HOME verbose
returns:	0

---

read_rc

parameters:	-
description:	If the `--norc` option was used, does nothing. Otherwise, sources the file in variable `rc`; if that is empty, use the /.mkrc if it exists.
globals set:	-
globals used:	HOME myname rc
returns:	0

---

check_needs

parameters:	-
description:	Checks if all executables neede are available.
globals set:	EDITOR
globals used:	EDITOR print ps view
returns:	1 if there are missing executables, else 0