mk a TeX and LaTeX maker

doc generated from the script with gendoc — bash script, version=3.10

Synopsis

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

General options:

```
-h,--help     print this help and exit
-H,--Help     print full documentation via less and exit
-V,--version  print version and exit
-v,--verbose  be verbose
 --noverbose  be quiet (this is the default)
-r,--rc=X     set rc file to X, but this must be the first option!
              If X is absent, don't read any rc file.
-n,--nocolor  no coloring instead of ANSI
```

mk specific options:

-f,–formatter=X Use X as the formatter: tex, latex, pdflatex,

```
  et cetera
```

```
-e,--edit=X  use X as the file to be edited; by default, the file
             file.
-C,--Clean   Remove all files generated by the compilation
-c,--clean   Same, except for the pdf or postscript files
```

vpp-related options:

```
-b,--batch=X    run in batch using X as a printing command for vpp.
                information.
-p,--printer=X  print to printer named X
 --view         view the document after compilation
 --noview       do not view
 --print        offer printing interaction after viewing
 --noprint      do not offer printing interaction
```

Defaults:

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

Arguments for short options are given without a separator, so you can write either `--rc=myrc` or `-rmyrc`.

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 tex document. Essentially, **mk** uses texi2dvi for compilation, vpp for viewing and printing. Any TeX formatter can be handled, with the exception of those starting with `ht` (such as httex and htlatex) and context (handle those with texexec).

Having an existing TeX 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 TeX 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 *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.

If the compilation results in an error, the eror wil be displayed by the `texlog_extract` script, and you will be prompted with:

```
=====> e(dit) c(ompile) q(uit)
```

giving you the opportunity to e(dit) the source, after which it will be recompiled, or to c(ompile) it again (because you may have editied it in another window), or just to quit.

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 a file read with `\input`, a style file, or any other file on which the target depends. However, files in the TEXMFMAIN tree are excluded.

Editor

vpp uses the contents of environment variable EDITOR to find your editor. 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 `--nofork`.

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
    h       display this help
    ?       display this help
```

```
q       quit
```

With these examples, no further explanation should be necessary, except that, when two sided (`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 directory is 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 `/Docs`) if you have defined one (see the section 'RC files').
- If the source was not found in `/Docs`, **mk** thinks that you may have a subdirectory *myfile* in `/Docs` 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 formatter to be used

mk determines the TeX formatter needed to compile the source as follows:

- if the `--formatter` option was used, its argument will be used;
- if the first line of the source starts with `%!`, the rest of that line specifies the formatter; for example:

  ```
  %!xelatex
  ```

- if the `--formatter` option was not used and the first line does not start with `%!`, **mk** looks for a `\usepackage{fontspec}` or `\RequirePackage{fontspec}` and, if found, uses *lualatex*. Those calls 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*;
- finally, if no match was found, *pdftex* is assumed.

Running in batch mode

If you don´t need to edit the source and to view the output, but just want to compile and maybe print the result, the `--batch` option is useful.

The `--batch=command` option prevents **mk** display the output and to interrogate the user about pages to be printed. Instead the document is printed according to the mandatory *command*, which mus be quoted if it contains spaces. Thus the command

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

prints 3 copies of pages 2 and 3 of `test.tex`, without viewing. If you just want to compile without printing anything, use the `q` command:

```
mk --batch=q test
```

or

```
mk -bq test
```

RC file and customization

Unless the option `--rc` 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:

`EDITOR=`
: sets your editor; You can also set your EDITOR in your system startup files. Note that your editor should not fork; so if you set it to gvim, do:
`EDITOR='gvim --nofork'`

`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 will 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 `/Docs/myfile.tex` you can set `altdir` to `/Docs` and run **mk** from any directory with:

```
$ mk myfile
```

However, a directory like `/Docs` 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 files read with `\include` or with `\input` such as graphics. You will then probably prefer to have a subdirectory in `/Docs` 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 regular expressions in this array will be skipped, however. For example, one could enter here:

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

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:
  `--preview`
  `--nocolor`
  `$basename`
- Deselect ”Auto-hide output panel unless errors occur”
- If you need an other formatter than the default, pdflatex, use the `%!` line in your source, or define separate tools in TeXWorks with an extra `--formatter=...` parameter.

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

Bugs

If, during the compile-edit-cycle, the `%!` line is changed, **mk** should respond to it by changing the formatter.

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.)

Author

Wybo Dekker

Copyright

Functions used:

excheck

```
synopsis:     excheck executable1 [executable2...]
description:  check if all needed execs are there and getopt is GNU
```

findsource

```
synopsis:      findsource [file]
description:   find the file to be compiled. If the argument has no extension,
               look for tex, ltx, drv, dtx, in that order.  If the argument is nil, look for main If the
               argument is xxx, look for xxx or xxx/main, in that order, in the current or the alternate
               directory. If the argument is xxx.ext, look for xxx.ext in the current or in the alternate
               directory.
globals set:   base dir ext
globals used:  IFS PWD RED Nor altdir default
```

run

```
synopsis:      run command [arg...]
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 used:  Yel Gre Blu Red Gre Nor
returns:       the exit value of the command
```

---

newtexdeps

```
description:   Scans $base.fls for tex dependencies and returns 1 if any of
               these is newer than target.
globals used:  base skip_pattern target
returns:       1: if any tex dependency is newer than target
```

---

newbibdeps

```
description:   Scan aux file ($base.aux) for bib-files needed and places


               those in the variable bibdeps. If there are bibdeps newer
               than target, remove .bbl, forcing texi2dvi to run bibtex

globals set:   bibdeps
globals used:  base target bibdeps
returns:       1: if any tex dependency is newer than target
```

---

compile

```
description:   runs the command in texcommand
globals used:  base bibdeps target targetext dvips texcommand
returns:       0 on success, else 1
```

---

show_error_and_edit

```
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:  color verbose Gre base bibdeps edit target warnings_to_skip
```

---

edit

```
synopsis:      edit edfile lineno ask
description:   Start the user´s editor to edit the edfile in argument 1
               (if empty: use $edit); start the edit at the line number in argument 2, if empty: 1). If
               the call was induced by the


               detection of an error in edfile, the third argument is
               true and the user will be asked if he wants to edit edfile,
to compile it again (in case he edited it externally) or to quit.
```

```
globals used:  edit EDITOR
returns:       1 if the file was edited, else 0
```

---

handle_options

```
synopsis:      handle_options "$@"
description:   Handles the options
globals set:   Clean batch clean dvips edit formatter input print printer
               rc verbose view
globals used:  Myname Version first HOME verbose formatter
```

---

setformatter

```
description:   set $formatter to the tex formatter to be used
globals set:   formatter targetext vpptargetext
globals used:  base ext
```

---

set_vpp_options

```
description:   set options for vpp call
globals set:   vppoptions
globals used:  batch color print printer pwd verbose view
```