Synopsis

mkpic [options] [picfile]

Options:

- -c, --clean
  remove all but the input file and die
- -p, --pdfsample
  create pdf file with sample images
- -f, --font=<font command>
  set default font for labels
  --[no]box
  produce framed boxes
- -V, --version
  report version number and die
- -h, --help
  display help info and die
  --[no]debug
  display debugging information
- -l, --log=<logfile>
  file for warning messages

Without an input file, the DATA section is used.

Command overview

begin name xl yl xmin ymin xmax ymax xlabel ylabel
end
stop
var=value
#
  comment

arccst xcenter ycenter xstart ystart theta
arcset xstart ystart xend yend theta
arccr tt xcenter ycenter radius theta1 theta2
arc3 x1 y1 x2 y2 x3 y3

xmark [label1] x1 [label2] x2 ...
Xmark [label1] x1 [label2] x2 ...
ymark [label1] y1 [label2] y2 ...
Ymark [label1] y1 [label2] y2 ...

xdrop x y
ydrop x y
xydrop x y

arrow x1 y1 x2 y2 label
label YX x y label
xlabels YX x y dx label ...
ylabels YX x y dy label ...

point x1 y1 x2 y2 ...

1
Description

mkpic provides an easy interface for making small pictures with mfpic. To this end you create an input file has to be created consisting of commands, one per line, with space separated parameters (or you modify the DATA section of the mkpic script, which is used if you run it without an input file). For an extensive description see the file mkpicdoc.pdf, which is part of the distribution.

mkpic produces two files. Assuming an input file named picfile defaulting to mkpic these are:

picfile.mac
   a macro file which will contain TeX commands for every picture
picfile.sty
   a style file for latex, defining the same TeX commands for every picture.

With the \texttt{--pdfsample} option, two other files are produced:

picfile.pdf
   a PDF file containing all pictures.
   This lets you easily check the results of your designs.

picfile.tex
   the TeX source used for creating this PDF file.

In \LaTeX{}, you have to include \texttt{\usepackage{picfile}} and to include commands like \texttt{\Fig name} in your source, where \texttt{name} is the name you gave one of your pictures in an \texttt{mkpic begin} command.

In TeX and ConTeXt, you have to \texttt{\input picfile.mac} and to include commands like \texttt{\Fig name} in your source, where \texttt{name} is the name you gave one of your pictures in an
**mkpic** begin command.

In TeX, you must use the \texttt{\bye} command (\textit{not} \texttt{\end}) to finish your TeX source

See the RUNNING section for how to run **mkpic** and TeX, LaTeX, or ConTeXt.

**Commands**

The source is set up so that it is easy to add your own commands, Currently the following commands have been implemented (the arguments are not listed here; for those, refer to the SYNOPSIS section):

\texttt{begin, end}

Every picture begins with the \texttt{begin} command and ends with the \texttt{end} command. The \texttt{begin} command defines a name for the picture and defines a tex command with that name, prefixed with ‘Fig’. The resulting command is written to a \texttt{.mac} file. Thus the command

\begin{verbatim}
begin aa ...
\end{verbatim}

starts writing \texttt{\def\Figaa{}...} to the \texttt{.mac} file, and the picture can be reproduced in a TeX document by importing the \texttt{.mac} file and using the \texttt{\Figaa} command.

\texttt{xl \text{and} yl} are the lengths of the x- and y-axes. xlabel and ylabel are the label that are placed at the ends of those axes. Use a space to suppress labeling, or “.” to suppress drawing the axes at all.

\texttt{stop}

stops further reading of the input. Useful if you have many pictures, but want to see only the first few for testing purposes.

\texttt{var=value}

sets the variable \texttt{var} to \texttt{value}. This variable, or an expression containing it, can be used instead of any numerical parameter. Variable names may contain lower and uppercase letters, digits or underscores, with the restriction that they must start with a letter and may not end in an underscore.

\texttt{#}

denotes a comment

\texttt{xmark, ymark, Xmark, Ymark}

These commands place one or more labels along the x- or y-axes, either below (\texttt{xmark} and \texttt{ymark}) or above (\texttt{Xmark} and \texttt{Ymark}) the axis.

For the \texttt{[xYy]mark} commands a parameter containing any character other than [-0-9] is interpreted as the label (this implies that you cannot use expressions here!) to be placed and its position is expected in the next parameter. If a parameter is just a number, it is placed at that x-position. If you want a number to be interpreted as a label, put it in braces: \{1950\}. 

3
arccst  
(Mnemonic: center start theta.)
Draws an arc with its center in \textit{xcenter, ycenter}, starting in \textit{xstart, ystart} and with an arc length of \textit{theta} degrees.

arcset  
(Mnemonic: start end theta.)
Draws an arc starting in \textit{xstart, ystart} and ending in \textit{xend, yend} and with an arc length of \textit{theta} degrees.

arccrnt  
(Mnemonic: center radius theta1 theta2.)
Draws an arc with its center in \textit{xcenter, ycenter}, a radius \textit{radius} starting at \textit{theta1} degrees and ending at \textit{theta2} degrees.

arc3  
(Mnemonic: 3 points.)
Draws an arc starting at \textit{(x1, y1)}, through \textit{(x2, y2)} and ending in \textit{(x3, y3)}.

\textit{xdrop, ydrop, xydrop}
These commands draw dotted arrows perpendicularly to the x-axis, the y-axis and both axes, respectively, ending on the axes with the arrow head.

\textit{arrow}
draws an arrow from \textit{(x1, y1)} to \textit{(x2, y2)} labeled on its tail with \textit{label}

\textit{label}
draws a label at \textit{(x, y)}. \textit{YX} tells how it will be adjusted: for \textit{Y=t,b,c (x, y)} will be, in the y-direction, on top, bottom or center of the label respectively, for \textit{X=l,r,c} it will be, in the x-direction, left, right or center adjusted on \textit{(x, y)}. Thus

\texttt{label tl 0 0 Hello World!}
will draw the string “Hello World” with its lower left corner at (0,0)

\textit{xlabels}
draws many labels, starting at \textit{(x, y)}, and incrementing \textit{x} with \textit{dx} after every label. \textit{YX}: see \textit{label}. Labels may not contain spaces; if you need spaces, use \textit{-} instead.

\textit{ylabels}
Same as \textit{xlabels}, but incrementing \textit{y} with \textit{dy} instead.

\textit{point}
draws points (dots) at \textit{(x1, y1)}, \textit{(x2, y2)} et cetera.
dpoint
draws points (dots) starting at \((x_1,y_1)\) and then moving by \((dx_1,dy_1), (dx_2,dy_2)\) et cetera.

lines
draws line segments from \((x_1,y_1)\) to \((x_2,y_2), (x_3,y_3)\) et cetera.

dlines
draws line segments starting at \((x_1,y_1)\) and then moving by \((dx_1,dy_1), (dx_2,dy_2)\) et cetera.

curve
draws a bezier curve from \((x_1,y_1)\) to \((x_2,y_2), (x_3,y_3)\) et cetera.

dcurve
draws a bezier curve starting at \((x_1,y_1)\) and then moving by \((dx_1,dy_1), (dx_2,dy_2)\) et cetera.

rect
draws a rectangle with diagonal points at \((x_1,y_1)\) and \((x_2,y_2)\).

drect
draws a rectangle with diagonal points at \((x,y)\) and \((x+dx,y+dy)\).

crect
clears a rectangle with diagonal points at \((x_1,y_1)\) and \((x_2,y_2)\).

dcrect
clears a rectangle with diagonal points at \((x,y)\) and \((x+dx,y+dy)\).

arect
draws a rectangle with a width \text{width} and a height \text{height};
the middle of the bottom is at \((xc,yc)\) and the centerline through
\((xc,yc)\) makes an angle \text{theta} with the x-axis.

bar
draws a equivalent with \text{rect\_x-xdev\_0\_x+xdev\_height}

func
draws the function given by \text{expression-in-x} between \text{xmin} and \text{xmax}, stepping with \text{step} units in the x-direction.
Note that the \text{expression-in-x} will be evaluated by \text{Metafont}, so
you will have to use metafont syntax.

grid
draw dotted grid lines at distances \text{dx} and \text{dy} in the x- and y
directions; the gaps between the dots are set to \text{xgap} an \text{ygap}
respectively. For an esthetic appearance, be sure to use integer\text{dx/xgap} and \text{dy/ygap} ratios.
hatch
  hatch the closed curve that follows.

bhat
  starts a path that will eventually be closed, and then hatched.

ehat
  ends a path started with bhat, closes it and then hatches it.

anything else
  will be inserted as is in the macro file, and therefore should be a valid \mfpic statement. You use this when you need such a statement only once, or a few times and therefore see no need to define a proper command for it.

Running mkpic/TeX
The effect of running
  mkpic picfile
is the creation of picfile.mac, which you can \\input into a TeX or ConTeXt source, and picfile.sty which can be input into a LaTeX source using the \usepackage command.
After running TeX (or LaTeX or ConTeXt), you will find a file picfile.mf and you will have to run Metafont on it, which (assuming you configured TeX for 600 dpi) produces picfile.600gf. This file will have to be converted to a pk file with gftopk. Finally, you need to run TeX, normally at least twice, again. So for pdfLaTeX the sequence is:
  mkpic picfile
  pdflatex file.tex
  mf picfile
  gftopk picfile.600gf
  pdflatex file
  pdflatex file

Bug
Currently only up to 256 pictures can be generated. In the future this problem will probably be solved by introducing more than one font and generating tex-command names that have the font name in front.

Changes
Changes with respect to version 0.1:
  • added mkpicdoc.tex to the distribution

Author and copyright
<table>
<thead>
<tr>
<th>Author</th>
<th>Wybo Dekker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td><a href="mailto:wybo@dekkerdocumenten.nl">wybo@dekkerdocumenten.nl</a></td>
</tr>
<tr>
<td>License</td>
<td>Released under the GNU General Public License</td>
</tr>
</tbody>
</table>