Name

pic – compile pictures for *troff* or TeX

Synopsis

pic [**−CnSU**] [*file* ...]

pic −t [**−cCSUz**] [*file* ...]

pic −−help

pic −v
pic −−version

Description

The GNU implementation of *pic* is part of the *groff*(1) document formatting system. *pic* is a *troff*(1) preprocessor that translates descriptions of diagrammatic pictures embedded in *roff*(7) or TEX input files into the language understood by TEX or *troff*. It copies the contents of each *file* to the standard output stream, except that lines between **.PS** and any of **.PE**, **.PF**, or **.PY** are interpreted as picture descriptions in the *pic* language. End a *pic* picture with **.PE** to leave the drawing position at the bottom of the picture, and with **.PF** or **.PY** to leave it at the top. Normally, *pic* is not executed directly by the user, but invoked by specifying the **−p** option to *groff*(1). If no *file* operands are given on the command line, or if *file* is "**−**", the standard input stream is read.

It is the user's responsibility to provide appropriate definitions of the **PS**, **PE**, and one or both of the **PF** and **PY** macros. When a macro package does not supply these, obtain simple definitions with the *groff* option **−mpic**; these will center each picture.

GNU *pic* supports **PY** as a synonym of **PF** to work around a name space collision with the *mm* macro package, which defines **PF** as a page footer management macro. Use **PF** preferentially unless a similar problem faces your document.

Options

−−help displays a usage message, while **−v** and **−−version** show version information; all exit afterward.

−c Be more compatible with *tpic*; implies **−t**. Lines beginning with \ are not passed through transparently. Lines beginning with **.** are passed through with the initial **.** changed to \. A line beginning with **.ps** is given special treatment: it takes an optional integer argument specifying the line thickness (pen size) in milliinches; a missing argument restores the previous line thickness; the default line thickness is 8 milliinches. The line thickness thus specified takes effect only when a non-negative line thickness has not been specified by use of the **thickness** attribute or by setting the **linethick** variable.

−C Recognize **.PS**, **.PE**, **.PF**, and **.PY** even when followed by a character other than space or newline.

−n Don't use *groff* extensions to the *troff* drawing commands. Specify this option if a postprocessor you're using doesn't support these extensions, described in *groff_out*(5). This option also causes *pic* not to use zero-length lines to draw dots in *troff* mode.

−S Operate in *safer mode;* **sh** commands are ignored. This mode, enabled by default, can be useful when operating on untrustworthy input.

−t Produce TEX output.

−U Operate in *unsafe mode;* **sh** commands are interpreted.

−z In TEX mode, draw dots using zero-length lines.

The following options supported by other versions of *pic* are ignored.

−D Draw all lines using the \D escape sequence. GNU *pic* always does this.

−T *dev* Generate output for the *troff* device *dev*. This is unnecessary because the *troff* output generated by GNU *pic* is device-independent.

Usage

This section primarily discusses the differences between GNU *pic* and the Eighth Edition Research Unix version of AT&T *pic* (1985). Many of these differences also apply to later versions of AT&T *pic*.

TEX mode

TEX-compatible output is produced when the **−t** option is specified. You must use a TEX driver that supports *tpic* version 2 specials. (*tpic* was a fork of AT&T *pic* by Tim Morgan of the University of California at Irvine that diverged from its source around 1984. It is best known today for lending its name to a group of **\special** commands it produced for TEX.)

Lines beginning with **** are passed through transparently; a **%** is added to the end of the line to avoid unwanted spaces. You can safely use this feature to change fonts or the value of **\baselineskip**. Anything else may well produce undesirable results; use at your own risk. By default, lines beginning with a dot are not treated specially—but see the **−c** option.

In TEX mode, *pic* will define a vbox called **\graph** for each picture. Use GNU *pic*'s **figname** command to change the name of the vbox. You must print that vbox yourself using the command

```
\centerline{\box\graph}
```

for instance. Since the vbox has a height of zero (it is defined with **\vtop**) this will produce slightly more vertical space above the picture than below it;

```
\centerline{\raise 1em\box\graph}
```

would avoid this. To give the vbox a positive height and a depth of zero (as used by LATEX's *graphics.sty*, for example) define the following macro in your document.

```
\def\gpicbox#1{%
  \vbox{\unvbox\csname #1\endcsname\kern 0pt}}
```

You can then simply say **\gpicbox{graph}** instead of **\box\graph**.

Commands

Several commands new to GNU *pic* accept delimiters, shown in their synopses as braces **{ }**. Nesting of braces is supported. Any other characters (except a space, tab, or newline) may be used as alternative delimiters, in which case the members of a given pair must be identical. Strings are recognized within delimiters of either kind; they may contain the delimiter character or unbalanced braces.

for *variable* **=** *expr1* **to** *expr2* [**by** [*****]*expr3*] **do** *X body X*

Set *variable* to *expr1*. While the value of *variable* is less than or equal to *expr2*, do *body* and increment *variable* by *expr3*; if **by** is not given, increment *variable* by 1. If *expr3* is prefixed by ***** then *variable* will instead be multiplied by *expr3*. The value of *expr3* can be negative for the additive case; *variable* is then tested whether it is greater than or equal to *expr2*. For the multiplicative case, *expr3* must be greater than zero. If the constraints aren't met, the loop isn't executed. *X* can be any character not occurring in *body*.

if *expr* **then** *X if-true X* [**else** *Y if-false Y*]

Evaluate *expr*; if it is non-zero then do *if-true*, otherwise do *if-false*. *X* can be any character not occurring in *if-true*. *Y* can be any character not occurring in *if-false*.

print *arg* . . .

Concatenate and write arguments to the standard error stream followed by a newline. Each *arg* must be an expression, a position, or text. This is useful for debugging.

command *arg* . . .

Concatenate arguments and pass them as a line to *troff* or TEX. Each *arg* must be an expression, a position, or text. **command** allows the values of *pic* variables to be passed to the formatter. For example,

```
.PS
x = 14
command ".ds string x is " x "."
.PE
\*[string]
```

produces

```
x is 14.
```

when formatted with *troff*.

sh *X command X*
: Pass *command* to a shell.

copy "*filename***"**
: Include *filename* at this point in the file.

copy [**"***filename***"**] **thru** *X body X* [**until** "*word***"**]
copy [**"***filename***"**] **thru** *macro* [**until** "*word***"**]
: This construct does *body* once for each line of *filename*; the line is split into blank-delimited words, and occurrences of **$***i* in *body*, for *i* between 1 and 9, are replaced by the *i*-th word of the line. If *filename* is not given, lines are taken from the current input up to **.PE**. If an **until** clause is specified, lines will be read only until a line the first word of which is *word*; that line will then be discarded. *X* can be any character not occurring in *body*. For example,

```
.PS
copy thru % circle at ($1,$2) % until "END"
1 2
3 4
5 6
END
box
.PE
```

and

```
.PS
circle at (1,2)
circle at (3,4)
circle at (5,6)
box
.PE
```

are equivalent. The commands to be performed for each line can also be taken from a macro defined earlier by giving the name of the macro as the argument to **thru**. The argument after **thru** is looked up as a macro name first; if not defined, its first character is interpreted as a delimiter.

reset
reset *pvar1*[**,**] *pvar2 . . .*
: Reset predefined variables *pvar1*, *pvar2 . . .* to their default values; if no arguments are given, reset all predefined variables to their default values. Variable names may be separated by commas, spaces, or both. Assigning a value to **scale** also causes all predefined variables that control dimensions to be reset to their default values times the new value of **scale**.

plot *expr* [**"***text***"**]
: This is a text object which is constructed by using *text* as a format string for sprintf with an argument of *expr*. If *text* is omitted a format string of **"%g"** is used. Attributes can be specified in the same way as for a normal text object. Be very careful that you specify an appropriate format string; *pic* does only very limited checking of the string. This is deprecated in favour of **sprintf**.

var **:=** *expr*
: This syntax resembles variable assignment with **=** except that *var* must already be defined, and *expr* will be assigned to *var* without creating a variable local to the current block. (By contrast, **=** defines *var* in the current block if it is not already defined there, and then changes the value in the current block only.) For example,

```
.PS
x = 3
y = 3
[
x := 5
```

```
        y = 5
        ]
        print x   y
        .PE
```

writes

```
        5 3
```

to the standard error stream.

Expressions

The syntax for expressions has been significantly extended.

x **^** *y* (exponentiation)
sin(*x***)**
cos(*x***)**
atan2(*y***,** *x***)**
log(*x***)** (base 10)
exp(*x***)** (base 10, i.e. 10^x)
sqrt(*x***)**
int(*x***)**
rand() (return a random number between 0 and 1)
rand(*x***)** (return a random number between 1 and *x*; deprecated)
srand(*x***)** (set the random number seed)
max(*e1***,** *e2***)**
min(*e1***,** *e2***)**
!*e*
e1 **&&** *e2*
e1 **||** *e2*
e1 **==** *e2*
e1 **!=** *e2*
e1 **>=** *e2*
e1 **>** *e2*
e1 **<=** *e2*
e1 **<** *e2*
"*str1***" == "***str2***"**
"*str1***" != "***str2***"**

String comparison expressions must be parenthesised in some contexts to avoid ambiguity.

Other changes

A bare expression, *expr*, is acceptable as an attribute; it is equivalent to *dir expr*, where *dir* is the current direction. For example

line 2i

means draw a line 2 inches long in the current direction. The 'i' (or 'I') character is ignored; to use another measurement unit, set the *scale* variable to an appropriate value.

The maximum width and height of the picture are taken from the variables **maxpswid** and **maxpsht**. Initially, these have values 8.5 and 11.

Scientific notation is allowed for numbers. For example

x = 5e−2

Text attributes can be compounded. For example,

"foo" above ljust

is valid.

There is no limit to the depth to which blocks can be examined. For example,

```
[A: [B: [C: box ]]] with .A.B.C.sw at 1,2
```

```
circle at last [].A.B.C
```

is acceptable.

Arcs now have compass points determined by the circle of which the arc is a part.

Circles, ellipses, and arcs can be dotted or dashed. In TEX mode splines can be dotted or dashed also.

Boxes can have rounded corners. The **rad** attribute specifies the radius of the quarter-circles at each corner. If no **rad** or **diam** attribute is given, a radius of **boxrad** is used. Initially, **boxrad** has a value of 0. A box with rounded corners can be dotted or dashed.

Boxes can have slanted sides. This effectively changes the shape of a box from a rectangle to an arbitrary parallelogram. The **xslanted** and **yslanted** attributes specify the x and y offset of the box's upper right corner from its default position.

The **.PS** line can have a second argument specifying a maximum height for the picture. If the width of zero is specified the width will be ignored in computing the scaling factor for the picture. GNU *pic* will always scale a picture by the same amount vertically as well as horizontally. This is different from DWB 2.0 *pic* which may scale a picture by a different amount vertically than horizontally if a height is specified.

Each text object has an invisible box associated with it. The compass points of a text object are determined by this box. The implicit motion associated with the object is also determined by this box. The dimensions of this box are taken from the width and height attributes; if the width attribute is not supplied then the width will be taken to be **textwid**; if the height attribute is not supplied then the height will be taken to be the number of text strings associated with the object times **textht**. Initially, **textwid** and **textht** have a value of 0.

In (almost all) places where a quoted text string can be used, an expression of the form

sprintf("*format***",** *arg***,** . . .**)**

can also be used; this will produce the arguments formatted according to *format*, which should be a string as described in *printf*(3) appropriate for the number of arguments supplied. Only the modifiers "**#**", "**−**", "**+**", and " " [space]), a minimum field width, an optional precision, and the conversion specifiers **%e**, **%E**, **%f**, **%g**, **%G**, and **%%** are supported.

The thickness of the lines used to draw objects is controlled by the **linethick** variable. This gives the thickness of lines in points. A negative value means use the default thickness: in TEX output mode, this means use a thickness of 8 milliinches; in TEX output mode with the **−c** option, this means use the line thickness specified by **.ps** lines; in *troff* output mode, this means use a thickness proportional to the pointsize. A zero value means draw the thinnest possible line supported by the output device. Initially, it has a value of −1. There is also a **thick**[**ness**] attribute. For example,

circle thickness 1.5

would draw a circle using a line with a thickness of 1.5 points. The thickness of lines is not affected by the value of the **scale** variable, nor by the width or height given in the **.PS** line.

Boxes (including boxes with rounded corners or slanted sides), circles and ellipses can be filled by giving them an attribute of **fill**[**ed**]. This takes an optional argument of an expression with a value between 0 and 1; 0 will fill it with white, 1 with black, values in between with a proportionally gray shade. A value greater than 1 can also be used: this means fill with the shade of gray that is currently being used for text and lines. Normally this will be black, but output devices may provide a mechanism for changing this. Without an argument, then the value of the variable **fillval** will be used. Initially, this has a value of 0.5. The invisible attribute does not affect the filling of objects. Any text associated with a filled object will be added after the object has been filled, so that the text will not be obscured by the filling.

Additional modifiers are available to draw colored objects: **outline**[**d**] sets the color of the outline, **shaded** the fill color, and **colo**[**u**]**r**[**ed**] sets both. All expect a subsequent string argument specifying the color.

```
circle shaded "green" outline "black"
```

Color is not yet supported in TEX mode. Device macro files like *ps.tmac* declare color names; you can define additional ones with the **defcolor** request (see *groff*(7)).

To change the name of the vbox in TEX mode, set the pseudo-variable **figname** (which is actually a specially parsed command) within a picture. Example:

```
.PS
figname = foobar;
...
.PE
```

The picture is then available in the box **\foobar**.

pic assumes that at the beginning of a picture both glyph and fill color are set to the default value.

Arrow heads will be drawn as solid triangles if the variable **arrowhead** is non-zero and either TEX mode is enabled or the **−n** option has not been given. Initially, **arrowhead** has a value of 1. Solid arrow heads are always filled with the current outline color.

The *troff* output of *pic* is device-independent. The **−T** option is therefore redundant. All numbers are taken to be in inches; numbers are never interpreted to be in *troff* machine units.

Objects can have an **aligned** attribute. This will only work if the postprocessor is *grops*(1) or *gropdf*(1). Any text associated with an object having the **aligned** attribute will be rotated about the center of the object so that it is aligned in the direction from the start point to the end point of the object. This attribute will have no effect on objects whose start and end points are coincident.

In places where *n***th** is allowed, **'***expr***'th** is also allowed. "**'th**" is a single token: no space is allowed between the apostrophe and the "**th**". For example,

```
for i = 1 to 4 do {
   line from 'i'th box.nw to 'i+1'th box.se
}
```

Conversion

To obtain a stand-alone picture from a *pic* file, enclose your *pic* code with **.PS** and **.PE** requests; *roff* configuration commands may be added at the beginning of the file, but no *roff* text.

It is necessary to feed this file into *groff* without adding any page information, so you must check which **.PS** and **.PE** requests are actually called. For example, the *mm* macro package adds a page number, which is very annoying. At the moment, calling standard *groff* without any macro package works. Alternatively, you can define your own requests, e.g., to do nothing:

```
.de PS
..
.de PE
..
```

groff itself does not provide direct conversion into other graphics file formats. But there are lots of possibilities if you first transform your picture into PostScript® format using the *groff* option **−Tps**. Since this *ps*-file lacks BoundingBox information it is not very useful by itself, but it may be fed into other conversion programs, usually named **ps2***other* or **psto***other* or the like. Moreover, the PostScript interpreter Ghostscript (*gs*(1)) has built-in graphics conversion devices that are called with the option

gs −sDEVICE=<*devname*>

Call

gs −−help

for a list of the available devices.

An alternative may be to use the **−Tpdf** option to convert your picture directly into **PDF** format. The MediaBox of the file produced can be controlled by passing a **−P−p** papersize to *groff*.

As the Encapsulated PostScript File Format **EPS** is getting more and more important, and the conversion wasn't regarded trivial in the past you might be interested to know that there is a conversion tool named *ps2eps* which does the right job. It is much better than the tool *ps2epsi* packaged with *gs*.

For bitmapped graphic formats, you should use *pstopnm*; the resulting (intermediate) *pnm*(5) file can be then converted to virtually any graphics format using the tools of the **netpbm** package.

Files

/usr/local/share/groff/1.23.0/tmac/pic.tmac
offers simple definitions of the **PS**, **PE**, **PF**, and **PY** macros.

Bugs

Characters that are invalid as input to GNU *troff* (see the *groff* Texinfo manual or *groff_char*(7) for a list) are rejected even in TEX mode.

The interpretation of **fillval** is incompatible with the *pic* in Tenth Edition Research Unix, which interprets 0 as black and 1 as white.

See also

/usr/local/share/doc/groff−1.23.0/pic.ps
"Making Pictures with GNU pic", by Eric S. Raymond. This file, together with its source, *pic.ms*, is part of the *groff* distribution.

"PIC—A Graphics Language for Typesetting: User Manual", by Brian W. Kernighan, 1984 (revised 1991), AT&T Bell Laboratories Computing Science Technical Report No. 116

ps2eps is available from CTAN mirrors, e.g., ⟨ftp://ftp.dante.de/tex−archive/support/ps2eps/⟩

W. Richard Stevens, *Turning PIC into HTML* ⟨http://www.kohala.com/start/troff/pic2html.html⟩

W. Richard Stevens, *Examples of* pic *Macros* ⟨http://www.kohala.com/start/troff/pic.examples.ps⟩

troff(1), *groff_out*(5), *tex*(1), *gs*(1), *ps2eps*(1), *pstopnm*(1), *ps2epsi*(1), *pnm*(5)