Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'gpic.1' command

$ man gpic.1

PIC(1) General Commands Manual PIC(1)

NAME

pic - compile pictures for troff or TeX

SYNOPSIS

pic [-nvCSU] [file ...]

pic -t [-cvzCSU] [file ...]

DESCRIPTION

This manual page describes the GNU version of pic, which is part of the groff document formatting system. pic compiles descriptions of pic? tures embedded within troff or TeX input files into commands that are understood by TeX or troff. Each picture starts with a line beginning with .PS and ends with a line beginning with .PE. Anything outside of .PS and .PE is passed through without change.

It is the user's responsibility to provide appropriate definitions of the PS and PE macros. When the macro package being used does not sup? ply such definitions (for example, old versions of -ms), appropriate definitions can be obtained with -mpic: These will center each picture.

OPTIONS

Options that do not take arguments may be grouped behind a single -. The special option -- can be used to mark the end of the options. A filename of - refers to the standard input.

-C Recognize .PS and .PE even when followed by a character other than space or newline.

-S Safer mode; do not execute sh commands. This can be useful when operating on untrustworthy input (enabled by default).

-U Unsafe mode; revert the default option -S.

-n Don't use the groff extensions to the troff drawing commands. You should use this if you are using a postprocessor that doesn't support these extensions. The extensions are described in groff_out(5). The -n option also causes pic not to use zero-length lines to draw dots in troff mode.

-t TeX mode.

-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 be? ginning with .ps is given special treatment: it takes an op? tional 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-nega? tive line thickness has not been specified by use of the thick? ness attribute or by setting the linethick variable.

-v Print the version number.

-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. pic always does this.

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

USAGE

This section describes only the differences between GNU pic and the original version of pic. Many of these differences also apply to newer versions of Unix pic. A complete documentation is available in the file

/usr/share/doc/groff/pic.ms

TeX mode is enabled by the -t option. In TeX mode, pic will define a vbox called \graph for each picture. Use the figname command to change the name of the vbox. You must yourself print that vbox using, for example, the command

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

Actually, 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 make the vbox having a positive height and a depth of zero (as used e.g. by LaTeX's graphics.sty), define the following macro in your document:

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

Now you can simply say \gpicbox{graph} instead of \box\graph.

You must use a TeX driver that supports the tpic specials, version 2.

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 to change the value of \baselineskip. Anything else may well produce undesirable results; use at your own risk. Lines beginning with a period are not given any special treatment.

Commands

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 the arguments and print as a line on stderr. Each arg must be an expression, a position, or text. This is useful for debugging.

command arg...

Concatenate the arguments and pass them through as a line to troff or TeX. Each arg must be an expression, a position, or text. This has a similar effect to a line beginning with . or \, but allows the values of variables to be passed through. For example,

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

prints

x is 14.

sh X command X

Pass command to a shell. X can be any character not occurring in command.

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
```

is equivalent to

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

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.

reset

reset variable1[,] variable2 ...

Reset pre-defined variables variable1, variable2 ... to their default values. If no arguments are given, reset all pre-de? fined variables to their default values. Note that assigning a value to scale also causes all pre-defined variables that con? trol 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.

variable := expr

This is similar to = except variable must already be defined, and expr will be assigned to variable without creating a vari? able local to the current block. (By contrast, = defines the variable in the current block if it is not already defined there, and then changes the value in the current block only.)

For example, the following:

```
.PS
x = 3
y = 3
[
  x := 5
  y = 5
]
print x " " y
.PE
```

prints

```
5 3
```

Arguments of the form

X anything X

are also allowed to be of the form

{ anything }

In this case anything can contain balanced occurrences of { and }.

Strings may contain X or imbalanced occurrences of { and }.

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 equiva? lent 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 vari? ables 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 ig? nored in computing the scaling factor for the picture. Note that GNU pic will always scale a picture by the same amount vertically as well as horizontally. This is different from the 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 mo? tion 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 ob? ject times textht. Initially textwid and textht have a value of 0.

In (almost all) places where a quoted text string can be used, an ex? pression 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) appropri? ate for the number of arguments supplied.

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 sup? ported 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 ob?

jects. 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.

Three additional modifiers are available to specify colored objects: outline[d] sets the color of the outline, shaded the fill color, and colo[u]r[ed] sets both. All three keywords expect a suffix specifying the color, for example

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

Currently, color support isn't available in TeX mode. Predefined color names for groff are in the device macro files, for example ps.tmac; ad? ditional colors can be defined with the .defcolor request (see the man? ual page of troff(1) for more details).

To change the name of the vbox in TeX mode, set the pseudo-variable figname (which is actually a specially parsed command) within a pic? ture. 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. Note that solid ar? row heads are always filled with the current outline color.

The troff output of pic is device-independent. The -T option is there? fore 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, or gropdf. 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. Note that this attribute will have no ef?
fect for objects whose start and end points are coincident.
In places where nth is allowed ?expr?th is also allowed. Note that ?th
is a single token: no space is allowed between the ? 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 trans?
form 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 ps2other or pstoother or the like. Moreover, the PostScript in?
terpreter ghostscript (gs) 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 file can be then converted to virtually any graphics format using the tools of the netpbm package.

FILES

/usr/share/groff/1.22.4/tmac/pic.tmac

Example definitions of the PS and PE macros.

SEE ALSO

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

Eric S. Raymond, Making Pictures With GNU PIC.

/usr/share/doc/groff/pic.ps (this file, together with its source file, pic.ms, is part of the groff documentation)

Tpic: Pic for TeX

Brian W. Kernighan, PIC ? A Graphics Language for Typesetting (User Manual) ?http://cm.bell-labs.com/cm/cs/cstr/116.ps.gz?. AT&T Bell Lab? oratories, Computing Science Technical Report No. 116 (revised May, 1991).

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?

BUGS

Input characters that are invalid for groff (i.e., those with ASCII code 0, or 013 octal, or between 015 and 037 octal, or between 0200 and 0237 octal) are rejected even in TeX mode.

The interpretation of fillval is incompatible with the pic in 10th edi? tion Unix, which interprets 0 as black and 1 as white.

PostScript? is a registered trademark of Adobe Systems Incorporation.