NAME

epsutil − a utility for manipulating Encapsulated PostScript files

SYNOPSIS

epsutil [BG=*color-value* **] [BOUNDINGBOX**=*llx,lly,urx,ury* **] [CHECK**=*check* **] [CLIP**=*clip* **]**
[COLORMODEL=*[CMYK | GRAY | HSB | RGB | X]* **]**
[FG=*color-value* **] [FRAME**=*frame-thickness* **] [FRAMEFG**=*color-value* **]**
[GRID=*grid* **]**
[GRIDX=*xmin,dx,xmax,units,gridstep,tickstep,ticklength,thickmultiple,fontname,fontsize* **]**
[GRIDY=*ymin,dy,ymax,units,gridstep,tickstep,ticklength,thickmultiple,fontname,fontsize* **]**
[HEIGHT=*height* **]**
[ID=*idx,idy[,fontsize[,fontname]]* **] [IDBG**=*color-value* **] [IDFG**=*color-value* **]**
[LINEWIDTHSCALE=*line-width-scale* **]**
[MAXLINEWIDTH=*maximum-line-width* **] [MINLINEWIDTH**=*minimum-line-width* **]**
[MXY=*page-magnification* **] [MX**=*page-x-magnification* **] [MY**=*page-y-magnification* **]**
[OXY=*origin* **] [OX**=*llx-origin* **] [OY**=*lly-origin* **] [PAPER**=*paper-name* **]**
[PSCODE=*'*POSTSCRIPT *code'* **] [PSFILE**=*psfile,psfile,. . .* **] [PAGEBG**=*color-value* **]**
[PMXY=*page-magnification-x-and-y-origin* **]**
[PMX=*page-magnification-x-origin* **] [PMY**=*page-magnification-y-origin* **]**
[PTXY=*page-translation-x-and-y* **] [PTX**=*page-translation-x* **] [PTY**=*page-translation-y* **]**
[RGBFILE=*rgbfile* **] [ROTATE**=*degrees* **]**
[SXY=*scale-factor* **] [SX**=*x-scale-factor* **] [SY**=*y-scale-factor* **] [WIDTH**=*width* **]**
[<] *old-PS-file* > *new-PS-file*

DESCRIPTION

epsutil filters an Encapsulated POSTSCRIPT file, or files, named on the command line (or *stdin* if no filenames are given), to *stdout*, inserting wrapper POSTSCRIPT to provide optionally

- background, foreground, frame, label, and page colors,
- BoundingBox adjustment and measurement,
- checking for non-EPS operators,
- clipping,
- framing,
- image rotation, scaling, and translation,
- linewidth scaling,
- page magnification and translation, and
- POSTSCRIPT alteration.

The output file fully conforms to the *POSTSCRIPT Document Structuring Conventions* defined in Appendix G of the *POSTSCRIPT Language Reference Manual*, Second Edition, Addison-Wesley (1985), ISBN 0-201-10174-2 (the Adobe red book).

The output of this program is also suitable input to it; after four filtering steps with 90-degree rotation selected, the original figure will be recovered, although the output POSTSCRIPT file will be about 6200 bytes larger for each rotation because of the wrapper overhead. That overhead could be reduced to about 4100 bytes by eliminating extra spaces and comments beginning with a single percent; readability is, however, considered more important.

The original command line that was used to create the output file is preserved in a *%%Title* comment in the second line of the output file, as a record of the requested transformations.

A verbatim copy of the original file, minus any *%%EOF* comments, is embedded in the output between the *%%BeginDocument* and *%%EndDocument* lines, where it can be recovered simply by stripping the wrapping POSTSCRIPT.

OPTIONS

epsutil options take the form of **NAME**=*value* assignments, rather than the more usual hyphen-prefixed options common in other UNIX utilities. If an option is given more than once, its *last* assigned value is used.

If no dimension unit is specified in the variable values described later, physical dimensions are assumed to be measured in standard POSTSCRIPT units of big points:

72bp = 1in = 25.4mm

Otherwise, any of the TEX dimension units of *bp* (big point), *cc* (cicero), *cm* (centimeter), *dd* (didot point), *in* (inch), *mm* (millimeter), *pc* (pica), *pt* (point), or *sp* (scaled point), may suffix a number, and the value will be converted to big points internally. For convenience, here are conversion tables for these units:

unit	1bp	1cc	1cm	1dd	1in
bp	1	12.7921	28.3465	1.06601	72
cc	0.078173	1	2.21593	0.0833333	5.62846
cm	0.0352778	0.451278	1	0.0376065	2.54
dd	0.938077	12	26.5911	1	67.5415
in	0.0138889	0.177669	0.393701	0.0148057	1
mm	0.352778	4.51278	10	0.376065	25.4
pc	0.0836458	1.07001	2.37106	0.0891674	6.0225
pt	1.00375	12.8401	28.4528	1.07001	72.27
sp	65781.8	841489	1864680	70124.1	4736287

unit	1mm	1pc	1pt	1000000sp
bp	2.83465	11.9552	0.996264	15.2018
cc	0.221593	0.934572	0.077881	1.18837
cm	0.1	0.421752	0.035146	0.536285
dd	2.65911	11.2149	0.934572	14.2604
in	0.0393701	0.166044	0.013837	0.211136
mm	1	4.21752	0.35146	5.36285
pc	0.237106	1	0.0833333	1.27157
pt	2.84528	12	1	15.2588
sp	186468	786432	65536	1000000

Letter case is *significant* in option names and dimension units.

Several options may be assigned lists of items. List items must be separated by any single character other than a letter, digits, plus, minus, or period. However, multiple spaces are reduced to single spaces before extracting list items. Comma or slash is a readable choice for a separator character, and avoids the need to quote the values to protect them from possible shell interpretation.

Color

The options **BG**=*color-value*, **FG**=*color-value*, **FRAMEFG**=*color-value*, **IDBG**=*color-value*, **IDFG**=*color-value*, and **PAGEBG**=*color-value* provide for setting the color of the BoundingBox background (**BG**) and foreground (**FG**), the frame foreground (**FRAMEFG**), the identifying label background (**IDBG**) and foreground (**IDFG**), and the page background (**PAGEBG**).

Color values can be provided in several different color models, set by the **COLORMODEL**=*[CMYK|GRAY|HSB|RGB|X]* option. In most cases, **epsutil** can guess the color model from the syntax of the color value, so that the **COLORMODEL** option rarely needs to be set.

A color can be specified as:

- a color name (e.g. *red*, *green*, *blue*, *skyblue*, *bisque*, . . .) [letter case is *not* significant];
- a hexadecimal color (e.g. *#3af*, *#33aaff*, *#333aaafff*, *#3333aaaaffff*) in the X Window System *RGB* model;
- a triple of numbers, each in the range 0 . . 1, for the *RGB* and *HSB* models;
- a quadruple of numbers, each in the range 0 . . 1, for the *CMYK* model;
- a single number in the range 0 . . 1 for the *GRAY* model.

Color names are looked up in the file specified by the **RGBFILE**=*rgbfile* option, or, if that is not given, in the X Window System *rgb.txt* color mapping file. These files associate red, green, and blue intensities with color names; their lines contain three numbers in the range 0 . . 255, followed by a color name, separated by whitespace. The *rgb.txt* file is found by searching a built-in directory path that covers all major UNIX systems; that path can be overridden by defining a value for the environment variable **RGBPATH**. For example, on an IBM RS/6000 AIX system, you could set that variable to */usr/lpp/X11/lib/X11/It_IT* in order to get Italian color names.

Hexadecimal colors in the X Window System color model specify red, green, and blue intensities on a scale of 0 . . 15 (3 hexadecimal digits), 0 . . 255 (6 hexadecimal digits), 0 . . 4095 (9 hexadecimal digits), or 0 . . 65536 (12 hexadecimal digits). Thus, *#3cf* corresponds to 3/15 red, 12/15 green, and 15/15 blue, that is, 0.2 red, 0.8 green, and 1.0 blue.

In the *RGB* model, three numbers in the range 0 . . 1 define red, green, and blue intensities. In this model, major colors are:

Black	0	0	0	Blue	0	0	1
Red	1	0	0	Magenta	1	0	1
Green	0	1	0	Cyan	0	1	1
Yellow	1	1	0	White	1	1	1

Like the *RGB* model, the *HSB* (hue, saturation, brightness) model also has three numbers in the range 0 . . 1. Hue runs through the rainbow (approximately, red = 0, orange = 0.1, limegreen = 0.2, green = 0.3, cyan = 0.5, blue = 0.6, magenta = 0.8, deeppink = 0.9, red = 1). Saturation goes from 0 (no color = black) to 1 (full color). Brightness goes from 0 (dark) to 1 (bright). The *HSB* model is the only one for which **COLORMODEL** *must* be defined, because **epsutil** otherwise assumes a default *RGB* model, and other models can be determined from context.

The color printing-industry standard *CMYK* (cyan, magenta, yellow, black) model requires four numbers in the range 0 . . 1. The first three are values for cyan, magenta, and yellow (0 means no color, 1 means full color). The fourth value is the amount of black (0 means none, 1 means completely black) to be added to darken the color.

Finally, in the *GRAY* model, a single number in the range 0 . . 1 defines the amount of gray: 0 is black and 1 is white. A value of *g* in the *GRAY* model is equivalent to *g,g,g* in the *RGB* model.

Sample options:

FG=*HotPink4*
BG=*0.9 (GRAY assumed)*
PAGEBG=*1,1,0 (RGB assumed)*
RGBFILE=*mycolors.rgb*
IDBG=*#7ed (X assumed)*
COLORMODEL=*HSB* **BG**=*0.6,0.8,0.9*
FG=*0.5,1,0.8,0.7 (CMYK assumed)*

BoundingBox

Encapsulated POSTSCRIPT files are required to contain a *%%BoundingBox* comment that defines the lower-left and upper-right corner coordinates in big points. Sadly, most POSTSCRIPT-producing programs produce (sometimes wildly) incorrect coordinate values in this comment, so manual adjustment is frequently necessary. Also, sometimes only a portion of the original figure is required, necessitating definition of a new BoundingBox.

To avoid the need for modifying the original POSTSCRIPT file, the two corner coordinate pairs can be specified as four optionally-dimensioned numbers for the value of the **BOUNDINGBOX** option. Values specified this way override those in the *%%BoundingBox* comment.

When a **BOUNDINGBOX** value is given, especially if it covers only a portion of the original figure, clipping should be selected. See the **CLIP** option description following.

Although fractional dimensions are accepted, after converting to big points, the values actually used will be rounded down to the nearest integer at the lower-left corner, and up to the nearest integer at the upper-right corner. This is a requirement of the *POSTSCRIPT Document Structuring Conventions*.

Sample options:

BOUNDINGBOX=*10,20,500,750*
BOUNDINGBOX=*'10 20 500 750'*
BOUNDINGBOX=*0.1in,0.3in,18cm,25cm*
BOUNDINGBOX=*10mm/20mm/30pc/40pc*

BoundingBox frame

A rectangular frame of a specified line thickness will be drawn around the *outside* of the original BoundingBox if the **FRAME**=*frame-thickness* option is given.

If linewidth scaling is requested, the frame will also be affected by the scaling.

The output BoundingBox will automatically be enlarged by the twice the frame thickness, and any **MAXLINEWIDTH** value will be adjusted upward if necessary to ensure that the frame can be drawn properly.

Sample options:

FRAME=*3*
FRAME=*3mm*

Checking for non-EPSF operators

If a non-zero value of the **CHECK**=*check* option is given, check for the presence of POSTSCRIPT operators that are not permitted in EPS files. This check is not guaranteed to succeed; see the **BUGS** section later for why.

Because this option requires additional pattern matching on every input line, and is only occasionally needed, the default is **CHECK**=*0* to suppress the check.

Sample options:

CHECK=*0*
CHECK=*1*

Clipping

If a zero value of the **CLIP**=*clip* option is given, suppress clipping the figure to the BoundingBox.

Although EPS-including programs are supposed to provide this clipping automatically, not all do so. Thus, the default in this program is to turn on clipping.

The clipping path is set after geometric transformations have been completed, but before any drawing operations are performed. Thus, the BoundingBox frame, and the background color, are subject to clipping.

Sample options:

CLIP=*0*
CLIP=*1*

Grid

A non-zero value for the **GRID**=*grid* option requests that a grid be drawn over the picture to allow convenient determination of BoundingBox coordinates for a portion of the view.

The grid is always drawn after the image, so that it cannot be overwritten by it.

The default grid is in standard POSTSCRIPT units of *big points*, and is matched to the selected paper size, leaving a small margin near the page edges where most POSTSCRIPT printers are incapable of printing.

For greater control, you can specify suitable values for the **GRIDX** and **GRIDY** options. These each take

the form of a list of up to *ten* items: *minimum number*, *numbering stepsize*, *maximum number*, *dimension unit name*, *grid stepsize*, *tick stepsize*, *tick length*, *thickening multiple*, *font size*, and *font name*. List items can be omitted, in which case internal defaults corresponding to the selected paper type will be supplied; see the **Paper type** section later. As with ordinary coordinate values, an omitted dimension on a number defaults to big points.

The default setting for the horizontal axis on American standard *A*-format letter paper is **GRIDX**=*50bp,25bp,575bp,bp,25bp,5bp,5bp,100bp,9bp,Helvetica*. This means that the axis will run from 50bp in steps of 25bp to 575bp, using units of big points, with grid lines every 25bp, tick marks 5bp long every 5bp, grid lines thickened at multiples of 100bp, with text in a 9bp-Helvetica font. Thus, the axis will be numbered 25, 75, 100, . . . , 550, 575, with grid lines drawn at the numbers, and 5 tick intervals between each pair of grid lines. If the eighth value were changed from 100bp to 0bp, then grid line thickening would be suppressed.

Sample options:

GRID=*0*
GRID=*1*

GRIDX=*,,,,,,,100bp*
GRIDX=*,,,,,,,,18bp,Times-Bold*
GRIDX=*50bp,25bp,575bp,bp*
GRIDX=*0bp,50bp,600bp,bp,5bp,5bp,5bp,100bp,9bp,Palatino-BoldItalic*
GRIDY=*0cm@1cm@25cm@cm@1cm@1mm@@5cm*
GRIDY=*0in,1in,11in,in*
GRIDY=*0in:1in:11in:in*
GRIDY=*0/50/700*

Identifying label

When working with large numbers of figures, or experimenting with various options, it is sometimes difficult to keep track of them. The **ID** option can be specified to provide a page location where an identifying label recording the command line will be printed.

The label location is measured on the output page, relative to the lower-left corner, independent of geometric transformations. The label is drawn after the image and the grid, in a shaded box which hides what lies under it.

An optional third value defines the font size (default: *24bp*), and an optional fourth value defines the font name (default: *Helvetica-Bold*).

The font size is automatically reduced to fit a long label into the available space, so that it will always be completely visible.

For convenience, the special assignment **ID**=*1* will produce a label in a default position, near the top of the page.

Sample options:

ID=*1*
ID=*50,750*
ID=*0.4in/0.4in/12pt*
ID=*'10mm 10mm 8pt Helvetica-Narrow'*

Linewidth control

Sometimes, linewidths need adjustment, usually to thicken them for better reproduction. The **LINEWIDTHSCALE** value causes all linewidths to be multiplied by that value. This scaling also affects any framing rectangle that is requested.

POSTSCRIPT allows zero linewidths, which means the thinnest line that the output device is capable of producing. Such linewidths would not be affected by scaling. Thus, a value for **MINLINEWIDTH** can be given to force all lines to be at least that thick, and therefore, subject to scaling.

To prevent scaling from producing ridiculously thick lines, a **MAXLINEWIDTH** value can be set to force an upper limit to linewidths. If not specified, it defaults to 24.

If linewidth scaling is applied to a POSTSCRIPT file that already has linewidth scaling, the two scalings are *not* cumulative: only the last scaling applied will have any effect. This unfortunate irregularity is a consequence of the inability of the POSTSCRIPT language to permit redefinition of an operator to more than one level; attempts to do otherwise simply produce infinite loops in the POSTSCRIPT interpreter.

Sample options:

LINEWIDTHSCALE=*3*
MAXLINEWIDTH=*6*
MAXLINEWIDTH=*6bp*
MINLINEWIDTH=*0.1pt*

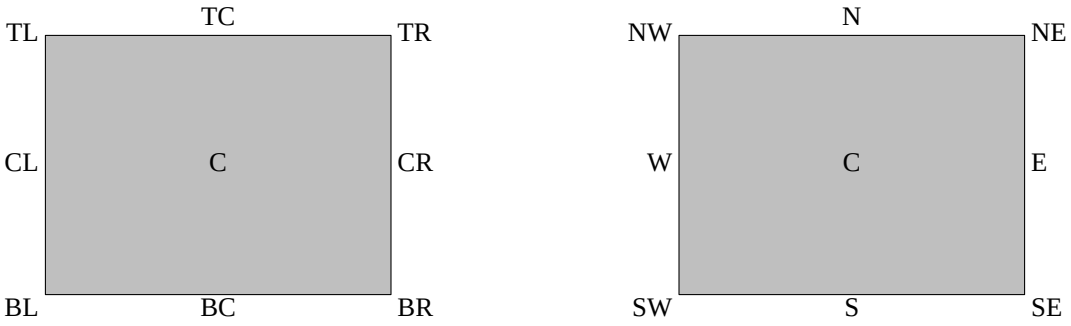
Magnification of page

To facilitate precise measurement of the BoundingBox, it can be helpful to magnify the page.

The **MX**, **MY**, and **MXY** options define magnification factors which are applied *after* page translation, and *before* all other transformations. **MX** and **MY** are used if provided; otherwise, **MXY** is used for both *x* and *y* scaling. If none of these is specified, then no page magnification is done; this is equivalent to setting **MXY**=1.

The page location about which magnification is done is (**PMX**,**PMY**)**,** or (**PMXY**,**PMXY**) if **PMX** and **PMY** are not provided. The default page location is the lower-left corner, (0bp,0bp).

As a convenience, symbolic coordinates of nine points of the BoundingBox (four corners, four edge centers, or box center) may be used for the page location values: combine a corner/edge/center designator *L* (left), *R* (right), *B* (bottom), *T* (top), *C* (center), or alternatively, compass-point names, with an additional coordinate designator *X* or *Y* for **PMX** and **PMY**,



In the symbolic value, neither the letter order, nor the letter case, is significant.

Page magnification does *not* affect the BoundingBox coordinates, so it must *not* be used for production of final EPS files.

Sample options:

MX=*2*
MY=*0.5*
MXY=*3*

PMX=*100bp*
PMY=*1cm*
PMXY=*5cc*
PMX=*SWX*
PMX=*BLX*
PMX=*xtr*
PMY=*cy*
PMY=*ney*
PMXY=*C*

Paper type

You can alter the default paper type by specifying a value for the **PAPER** option. The paper types recognized are:

Name	Width	Height	Name	Width	Height
A	8.5in	11in	A7L	105mm	74mm
B	11in	17in	A8L	74mm	52mm
C	17in	22in	A9L	52mm	37mm
D	22in	34in	A10L	37mm	26mm
E	34in	44in	B0	1000mm	1414mm
Computer-1411	14in	11in	B1	707mm	1000mm
Legal	8.5in	13in	B2	500mm	707mm
Letter	8.5in	11in	B3	353mm	500mm
US-Legal	8.5in	14in	B4	250mm	353mm
A-L	11in	8.5in	B5	176mm	250mm
Computer-1411-L	11in	14in	B6	125mm	176mm
Legal-L	13in	8.5in	B0L	1414mm	1000mm
Letter-L	11in	8.5in	B1L	1000mm	707mm
US-Legal-L	14in	8.5in	B2L	707mm	500mm
COM10	4.1in	9.5in	B3L	500mm	353mm
DL	110mm	220mm	B4L	353mm	250mm
Executive	7.25in	10.5in	B5L	250mm	176mm
Monarch	3.9in	7.5in	B6L	176mm	125mm
A4Small	210mm	297mm	C0	1294mm	916mm
Ledger	11in	17in	C1	916mm	647mm
LetterSmall	8.5in	11in	C2	647mm	458mm
Note	8.5in	11in	C3	458mm	323mm
A0	841mm	1189mm	C4	323mm	229mm
A1	594mm	841mm	C5	229mm	161mm
A2	420mm	594mm	C6	161mm	114mm
A3	297mm	420mm	Octavo	5in	8in
A4	210mm	297mm	Sixmo	6.5in	8in
A5	148mm	210mm	Quarto	8in	10in
A6	105mm	148mm	Foolscap	8.5in	13in
A7	74mm	105mm	Government-legal	8.5in	13in
A8	52mm	74mm	Folio	8.3in	13in
A9	37mm	52mm	ArchA	9in	12in
A10	26mm	37mm	ArchB	12in	18in
A0L	1189mm	841mm	ArchC	18in	24in
A1L	841mm	594mm	ArchD	24in	36in
A2L	594mm	420mm	ArchE	36in	48in
A3L	420mm	297mm	Flsa	8.5in	13in
A4L	297mm	210mm	Flsb	8.5in	13in
A5L	210mm	148mm	HalfLetter	5.5in	8.5in
A6L	148mm	105mm			

In addition to the standard sizes, any particular size can be requested by giving it as a list of *width* and *height* values, for example, *8.5in,11in* or *210mm/297mm*.

An invalid **PAPER** type produces a list of recognized types on *stderr*, so you can use a nonsense value like **PAPER**=*helpme* to find out what is available.

The default paper type is installation dependent: usually *A* (in the USA) or *A4* (elsewhere).

The page dimensions are always defined in the output POSTSCRIPT prolog as *PageWidth* and *PageHeight*, so

that other user-defined POSTSCRIPT commands can make use of them. This is true even if the **PAPER** option is not used.

Besides the page dimensions, the BoundingBox coordinates are always available for use in POSTSCRIPT as *LLX*, *LLY*, *URX*, and *URY*.

Sample options:

PAPER=*148mm/210mm*
PAPER=*A5*
PAPER=*B*
PAPER=*Government-legal*

POSTSCRIPT insertions

Two options, **PSCODE** and **PSFILE**, provide for insertion of additional POSTSCRIPT code into the end of the *%%BeginProlog . . . %%EndProlog* section where POSTSCRIPT commands are defined.

The value of **PSCODE** is a string containing arbitrary literal POSTSCRIPT code.

The value of **PSFILE** is the name of one or more comma- or space-separated files whose contents are to be inserted.

When both options are specified, **PSFILE** is handled *before* **PSCODE**, so that any POSTSCRIPT read from files can be overridden by literal POSTSCRIPT from the command line.

For additional power and flexibility, this code can also optionally redefine one or more *hooks : BeginPageSetupHook*, *EndPageSetupHook*, *BeginDocumentHook*, *EndDocumentHook*, and *EndJobHook*. These commands are executed at the stages indicated by their names, allowing POSTSCRIPT code to be effectively inserted at five different places in the execution stream, without the need for tedious hand editing.

The inserted POSTSCRIPT, including the hooks, is executed with a private dictionary, *EPSutilDict*, at the top of the dictionary stack. This dictionary is large (1000 elements), so that it is highly unlikely that dictionary overflow can occur. In any event, this could be a problem only with older POSTSCRIPT devices, since Level 2 POSTSCRIPT, which automatically enlarges dictionaries as needed, dates from 1990, and most POSTSCRIPT devices manufactured since then fully support it.

If you define hooks in private packages of macros, you will soon encounter the need for extending an existing hook. If each package just does something like

/EndJobHook { . . . code . . . } def

then only the last-loaded hook definition will survive. Fortunately, there is a solution to this problem which works for POSTSCRIPT Version 38 and later (Level 1 POSTSCRIPT implementations from the late 1980s, and all Level 2 POSTSCRIPT implementations):

/EndJobHook { //EndJobHook exec . . . more code . . . } def

You could also write this as

/EndJobHook { . . . more code . . . //EndJobHook exec } def

if you wanted the new hook code to be added *before* the old hook code. The double slash in *//EndJobHook* asks for immediate evaluation of the name. This immediately replaces the name by its definition (a procedure) within the new definition of the hook. When the new definition is executed, the *exec* operator will execute that procedure, effectively inserting the old definition at the beginning of the new one. Without the *exec*, the procedure would just be left as an unused object on the execution stack. I know of no way to accomplish this without the Version 38 and later // immediate-execution feature.

The facility provided by these options is very powerful, since arbitrary extra POSTSCRIPT code can be inserted, and POSTSCRIPT operators can even be redefined.

Sample options:

Disable region fill permanently:
PSCODE=*'/fill { } def'*

Disable region fill only for its first invocation, thereby removing an initial region fill that some POSTSCRIPT-producing programs always provide:
PSCODE=*'/oldfill /fill load def*
/fill { /fill /oldfill load def } def'

Supply a copyright claim in the bottom right margin of the BoundingBox:
PSCODE=*'/EndJobHook { /Helvetica findfont 6 scalefont setfont*
(Copyright, International Widget Corporation (2001))
dup stringwidth pop URX exch sub LLY 4 add moveto show } def'

Change all solid lines and dashed lines to dashed lines with a fixed dash pattern:
PSCODE=*'/BeginDocumentHook { [3] 0 setdash /setdash { pop pop } def } def'*

Disable rectangle fill:
PSCODE=*'/rectfill { pop pop pop pop } def'*

Color the page light blue:
PSCODE=*'gsave 0.7 0.7 1 setrgbcolor 0 0 PageWidth PageHeight rectfill grestore'*
These commands are equivalent to **COLORMODEL**=*RGB* **PAGEBG**=*0.7,0.7,1.*

Set the default foreground color to blue:
PSCODE=*'0 0 1 setrgbcolor'*

Change all colors to shades of gray, assuming that the POSTSCRIPT file uses only the *RGB* color model:
PSCODE=*'/setrgbcolor {add add 3 div setgray} bind def'*

Fill the background with a blue color ramp, darker at the bottom, lighter at the top (a popular choice for overhead transparencies and slides):

```
PSCODE='gsave
  /delta 0.005 def
  0.0 delta 1
  {
    /f exch def
    /g 0.6 dup 1 exch sub f mul add def
    0.9 g mul 0.9 g mul 1 g mul setrgbcolor
    0 PageHeight f mul PageWidth PageHeight delta mul rectfill
  } for
grestore'
```

Suppress fills 1, 2, 4, and 7:

```
PSCODE='/BeginDocumentHook {
 /IgnoreFill [ 1 2 4 7 ] def
 /oldfill /fill load def
 /FillCount 1 def
 /fill {
  /DoFill true def
  0 1 IgnoreFill length 1 sub
  {
   IgnoreFill exch get FillCount eq { /DoFill false def } if
  } for
  /FillCount FillCount 1 add def
  DoFill { oldfill } if
 } def
} def'
```

This example is complex enough that some explanation is in order. The *IgnoreFill* vector contains a list of fill counts at which filling is to be suppressed. Each time *fill* is invoked, the *for* loop iterates over this list, and sets the *DoFill* flag to *false* if the current fill count is in the list. At the end of the loop, *FillCount* is incremented by one, and the original fill command is executed if *DoFill* is still *true*. Even without understanding the details, you can easily modify the code by changing the *IgnoreFill* list entries to select any particular fill operations that you want to suppress. This can be useful for removing unwanted background fills produced by some programs. Given the common use by many programs of inscrutable collections of private POSTSCRIPT macros, direct modification of the original POSTSCRIPT code to accomplish the same goal would likely be humanly

infeasible.

For more extensive modifications:
PSFILE=*mymacros.eps*
PSFILE=*header1.eps,../tools/monochrome.eps*

Rotation

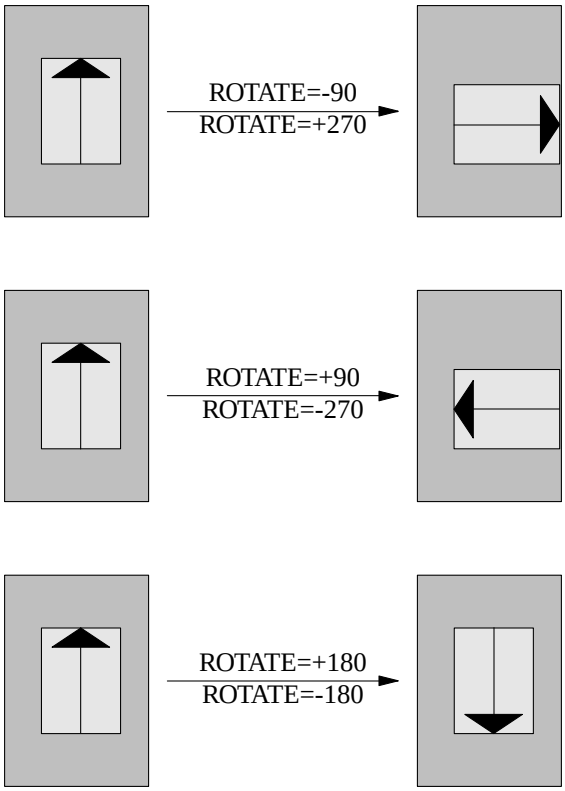
The **ROTATE** option value is a rotation angle in degrees, positive counterclockwise, and negative clockwise. Only multiples of +/-90 degrees, or values less than 90 degrees in absolute value, are supported. Non-conforming values will elicit a warning and be ignored, preventing any rotation.

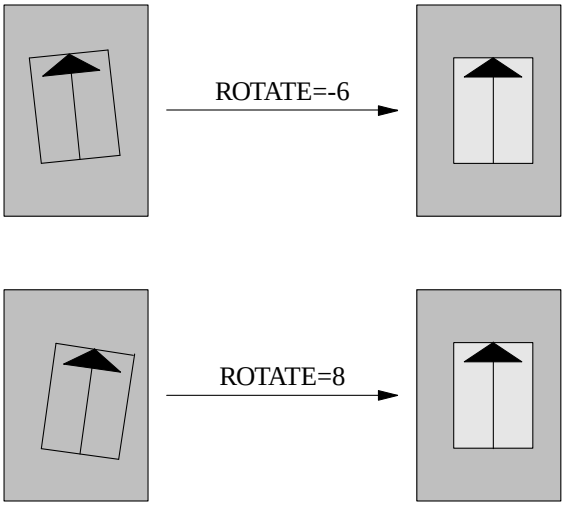
For example, to turn landscape orientation with the right side at the top of the page into portrait orientation, specify **ROTATE**=*-90*. It will usually be necessary to supply additional scaling (**HEIGHT** or **WIDTH**, or **SX**, **SY**, or **SXY** options) to make the picture fit on the output page.

Unless otherwise specified by the translation variables described later, the position of the lower-left corner of the BoundingBox on the output display device is kept invariant, so that after a -90-degree rotation, the lower-right corner of the original BoundingBox will be positioned where the lower-left corner was.

Small rotation angles can be useful for correcting the orientation of slightly-rotated scanned images, since it is often difficult to align precisely an original document on the scanner glass. The output file then contains a comment of the form *%%RotatedBoundingBox: 71 363 229 522* giving coordinates of a new BoundingBox that may be suitable for use in a subsequent invocation of **epsutil**, *provided* that no further transformations are made. That comment is also printed on *stderr*.

Here are some sample rotations; notice that the page position of the lower-left corner of the original image is always occupied by an image corner after the rotation.





Sample options:

ROTATE=*0*
ROTATE=*-90*
ROTATE=*90*
ROTATE=*180*
ROTATE=*270*
ROTATE=*360*
ROTATE=*450*
ROTATE=*-2*
ROTATE=*3.5*

Translation of BoundingBox

The origin option variables move the lower-left corner of the BoundingBox to (**OX**,**OY**), relative to the lower-left page corner, or if **OX** and **OY** are not specified, then to (**OXY**,**OXY**). Otherwise, the lower-left corner of the new BoundingBox will be at the same physical page location as one of the corners of the original picture; which one depends on the rotation angle chosen.

Sample options:

OX=*0*
OY=*0*
OX=*10mm*
OY=*36bp*
OXY=*1in*
OXY=*3pc*

Translation of page

The lower-left corner of the page is normally at (0bp,0bp). The page translation variables move that corner to (**PTX**,**PTY**), or if that is not specified, then to (**PTXY**,**PTXY**).

Use this option to move a figure, and any overlaid grid, away from the page edges where most POSTSCRIPT printers are incapable of printing.

This option does *not* affect the BoundingBox coordinates, so it must *not* be used for production of final EPS files.

Sample options:

PTX=*0*
PTY=*0*
PTX=*1cm*

PTY=*36dd*
PTXY=*1000000sp*
PTXY=*3cc*

Scaling (absolute)

The **HEIGHT**=*height* and **WIDTH**=*width* options provide a way to request absolute scaling of the BoundingBox to a specified height or width. If only one of them is given, then the other is computed to preserve the aspect ratio for distortion-free scaling. If both are given, then the horizontal and vertical scale factors may be different; the aspect ratio will then not be preserved.

The two sizes refer to the final figure, *after* any rotation.

If both absolute (**HEIGHT** and **WIDTH**) and relative (**SX**, **SY**, and **SXY**) scaling are specified, then absolute scaling is done, and the relative scale factors are ignored.

Sample options:

HEIGHT=*400bp*
HEIGHT=*15cm*
WIDTH=*340*
WIDTH=*150mm*

Scaling (relative)

The **SX**, **SY**, and **SXY** options provide for relative scaling. Any scale factors provided are applied *after* the rotation, so that they apply to the final figure. **SX** and **SY** are used if provided; otherwise, **SXY** is used for both *x* and *y* scaling if provided.

Sample options:

SX=*0.5*
SY=*2.0*
SXY=*0.7727*

HAND MODIFICATION

The output file can be easily modified by hand later to adjust any of the options that can be set on the command line, even if you are unfamiliar with the POSTSCRIPT language. Look for the lines following the first *%%BeginProlog* comment that begin */NAME ... def*, and then change the alphanumeric values as needed. Comment lines preceding each set of related variable definitions provide suitable documentation.

BUGS

Until a valid *%%BoundingBox* comment has been located, or a **BOUNDINGBOX** command-line option has been supplied, the input POSTSCRIPT is buffered in memory. On small machines, or with very large POSTSCRIPT files that contain the *%%BoundingBox* comment near the end instead of near the beginning, it is possible that your computer memory (and swap space, if any), might be exceeded. The easiest solution for such a problem is to supply a command-line **BOUNDINGBOX** value. This problem is expected to be exceedingly rare.

Encapsulated POSTSCRIPT is a subset of standard POSTSCRIPT that excludes approximately twenty operators whose use can prevent successful inclusion of a POSTSCRIPT file inside another POSTSCRIPT file, usually because those operators would obviate the use of a local independently-transformed coordinate system, or would modify the global state of the POSTSCRIPT environment. When the output file fails to conform to the specified transformations, it is often because of the use of these excluded operators. If requested by the **CHECK**=*1* option, **epsutil** looks for those operators, and will raise warnings when they are seen. Unfortunately, their apparent absence does not mean that they are not used, because POSTSCRIPT files can also contain binary, hexadecimal, and other encodings that can hide such operators from view. Thus, the only truly reliable check is visual examination of the output of **epsutil** with a POSTSCRIPT previewer or printer, and its successful incorporation in other documents, such as with TEX *\epsffile* or *\special* commands, or in word processors.

The **ROTATE** support needs further generalization to handle arbitrary rotation angles. For now, in some cases, two passes through **epsutil** may be needed.

ENVIRONMENT VARIABLES

RGBPATH Search path (a colon-separated list of directories) for the X Window System color definition file, *rgb.txt*.

FILES

rgb.txt Color mapping file distributed with the X Window System.

AVAILABILITY

The master source distribution for **epsutil** is maintained at the Internet archive location *ftp://ftp.math.utah.edu/pub/misc/index.html#epsutil-2.01* for both anonymous ftp and World-Wide Web access. The source code is in the public domain, and copies of the program may be freely distributed and used for any purpose.

SEE ALSO

dvialw(1), **dvips**(1), **enscript**(1), **genscript**(1), **ghostscript**(1), **gs**(1), **lptops**(1), **pageview**(1), **plot79**(1), **postscript**(7), **psposter**(1), **tek2ps**(1), **tekalw**(1), **transcript**(1), **x79ps**(1), **xpsview**(1).

AUTHOR

Nelson H. F. Beebe
Center for Scientific Computing
Department of Mathematics
University of Utah
Salt Lake City, UT 84112
USA
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Email: beebe@math.utah.edu
URL: http://www.math.utah.edu/~beebe