NAME

ps2eps − convert PostScript to EPS (Encapsulated PostScript) files

SYNOPSIS

ps2eps [**-f**] [**-q**] [**-N**] [**-O**] [**-n**] [**-P**] [**-c**] [**-C**] [**-m**] [**-B**] [**-E**] [**-s** *pagedim*] [**-t** *offset*] [**-r** *resolution*] [**-R** +|-|^] [**-l**] [**-g**] [**-H**] [**-d**] [**-h|--help**] [**-a**] [**-W**] [**-L**] [**-V|--version**] [**--**] [*psfile1*] [*psfile2*] [*...*]

DESCRIPTION

This manual page documents **ps2eps** version 1.68.

ps2eps is a tool (written in Perl) to produce Encapsulated PostScript Files (EPS/EPSF) from usual one-paged Postscript documents. It calculates correct Bounding Boxes for those EPS files and filters some special postscript command sequences that can produce erroneous results on printers. EPS files are often needed for including (scalable) graphics of high quality into TeX/LaTeX (or even Word) documents.

Without any argument, ps2eps reads from standard input and writes to standard output. If filenames are given as arguments they are processed one by one and output files are written to filenames with extension *.eps*. If input filenames have the extension *.ps* or *.prn*, this extension is replaced with *.eps*. In all other cases *.eps* is appended to the input filename. Please note that PostScript files for input should contain only one single page (you can possibly use the **psselect** from the psutils package to extract a single page from a document that contains multiple pages).

If BoundingBox in output seems to be wrong, please try options **--size** or **--ignoreBB**. See also section TROUBLESHOOTING.

OPTIONS

ps2eps follows the usual GNU command line syntax, with long options starting with two dashes (‘-’). A summary of options is included below.

-h, --help
Show summary of options.

-V, --version
Show version of program.

-f, --force
Force overwriting existing files. **ps2eps** will not overwrite files by default to avoid deleting original EPS files accidently.

-q, --quiet
quiet operation (no output while processing files, except errors).

-N, --noinsert
do not insert any postscript code. Normally a few postscript instructions are added around the original postscript code by **ps2eps** which can be turned off by this option.

-O, --preserveorientation
do not filter %%Orientation: header comment.

-n, --nofix
do not try to fix postscript code by filtering some instructions.

-P, --removepreview
remove preview image (smaller file, but no preview anymore).

-F, --fixps
fix postscript code unconditionally. Otherwise, filtering is usually triggered by detection of certain drivers only.

-c, --comments
preserve document structure comments.

-C, --clip
insert postscript code for clipping. Unless **--nohires** is specified, the HiResBoundingBox (enlarged by 0.1 points) is used for clipping.

-m, --mono
use black/white bitmap as base for calculation (default: off).

-s, --size=pagedim
where pagedim is a pre-defined standard page size (e.g., a4,a0,b0,letter,...) or explicitly specified in a format pagedim:=*X*x*Y*[cm|in], where *X* and *Y* are numbers (floating points are accepted) followed by units centimeter (cm) or inch (in), (default: cm). Use **--size=list** to list pre-defined pagesizes. See also environment variable PS2EPS_SIZE.

-t, --translate=x,y
specify an x,y offset (may be negative) in postscript points (1/72 dpi) for drawing. This option may be required if your drawing has negative coordinates which usually lets ghostscript cut the negative part of your picture, because it starts to render at positive coordinates. The resulting output will also be shifted.

-r, --resolution=dpi
specify a resolution in dpi (dots per inch) for drawing under ghostscript. Default resolution is 144 dpi which is the double of the typical 72 dpi. This option may help if there is a hardware dependent resolution encoded in the postscript, e.g., 600dpi. Example: **ps2eps -l -r 600 test.ps**

-R, --rotate=direction
This option rotates the resulting EPS output. The parameter direction determines the direction of rotation: + means +90 degrees (clockwise),- means -90 degrees (counter-clockwise), and ^ means 180 degrees (up-side down).

-l, --loose
expand the original tight bounding box by one point in each direction.

-B, --ignoreBB
do not use existing bounding box as page size for rendering.

-E, --ignoreEOF
do not use %%EOF as hint for end of file. Otherwise, **ps2eps** assumes that postscript code ends after the last %%EOF comment, because some drivers add trailing binary ‘‘garbage’’ code which gets deleted by **ps2eps** by default.

-g, --gsbbox
use internal bbox device of ghostscript instead of the external C program **bbox**. The internal bbox device of ghostscript generates different values (sometimes even incorrect), so using the provided **bbox** should be more robust. See also environment variable

PS2EPS_GSBBOX.

-H, --nohires
do not generate a %%HiResBoundingBox comment for output.

-a, --accuracy
increase the accuracy by turning subsample antialiasing on (may be slower)

-L, --license
show licensing information.

-d, --debuggs
show ghostscript call. This may be helpful for solving problems that occur during a ghostscript call.

-W, --warnings
show warnings about sanity of generated EPS file. Certain postscript commands should not be contained in an EPS file. With this option set **ps2eps** will issue a warning if it detects at least one of them.

TROUBLESHOOTING

Based on the given postscript source code (in most cases generated by some postscript printer driver) there are many potential obstacles or problems that may occur when trying to create proper EPS files. Please read this section carefully to be aware of common pitfalls.

INCOMPLETE/CLIPPED IMAGES

or how to determine the right size for ghostscript.

If you have documents that are larger than your ghostscript default (usually A4 or US letter), you have to specify the page dimensions explicitly using the **-s** option. Otherwise your EPS might be cut off during rasterizing by ghostscript resulting in a wrongly calculated bounding box. You can pass all pre-defined page sizes to **-s** that ghostscript understands. These are currently: 11x17, ledger, legal, letter, lettersmall, archA, archB, archC, archD, archE a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, isob0, isob1, isob2, isob3, isob4, isob5, isob6, b0, b1, b2, b3, b4, b5, c0, c1, c2, c3, c4, c5, c6, jisb0, jisb1, jisb2, jisb3, jisb4, jisb5, jisb6, flsa, flse, halfletter. Unfortunately, all sizes are currently only available in portrait orientation (not landscape).

By default, **ps2eps** uses an already given %%BoundingBox from the source file, which often corresponds to the size of the physical page format for which the document was printed. However, you should be aware that this already specified bounding box may be not correct, thus resulting in a wrongly cropped (or even no usable) *.eps*-file. **ps2eps** can only do as good as ghostscript does in rendering the original postscript file (though **ps2eps** even works with negative and fractional values are contained in the original bounding box by using automatic translation). Therefore, if the given bounding box is to small or incorrect anyway, you can ignore the existing bounding box with the **-B** option, which will cause ghostscript to use its internal default size (or use **-s**). However, if the BoundingBox has negative coordinates, which is not allowed by the specification, **ps2eps** will shift the output to positive values.

Hint: to avoid rotating the picture if you have the original drawing in landscape format, you may use the ‘‘Encapsulated Postscript’’ option in the printer driver which should generate an EPS file (but with a bounding box of the sheet size!). But some Windows printer drivers are drawing the image with an offset from the bottom of the portrait page, so that a part of it is drawn outside the landscape oriented page. In this case, you’ll have to specify a square size of the page using the maximum length, e.g., 29.7cm x 29.7cm for an A4 page.

CLIPPING

or why gets some of my text deleted above the included *.eps* file?

Some postscript drivers draw a white rectangle from the top left corner of the page to the right lower corner of the object. This may erase some or even all text above your imported/included EPS file, which is very annoying. In order to prevent this, most programs have a clipping option for imported *.eps* files (within LaTeX you can use \includegraphics*{}) for this purpose. If this is unfortunately not the case, you can use the **-C** option of **ps2eps** which will (hopefully) do it for you. Unfortunately, PScript.dll 5.2 (Windows XP) introduced new very badly behaving Postscript code (initclip) which will even override the outer clipping! Thus, a new filter had to be installed in **ps2eps** which will fix it.

However, because most programs clip directly on the bounding box, you still may loose some pixels of your image, because the bounding box is described in the coarse resolution of postscript points, i.e. 72 dpi. In order to prevent this, you can use the **-l** option or **-C** option (for the latter, clipping by the importing program should be disabled then) to allow for a 1 point larger bounding box. **-C** clips around a 1 point enlarged bounding box and **-l** enlarges the bounding box values by 1 point (you can also combine both options).

INCLUDED FILTERS

Some postscript sequences, e.g., for using specific printer features (featurebegin ...), are not working well within an *.eps* file, so **ps2eps** tries to filter them out. But please note that filters for postscript code may not work properly for your printer driver (**ps2eps** was mainly tested with HP and Adobe printer drivers, although it may work for all printers using the PScript.dll). In this case you can try to turn of filtering by using option **-n**, or try to find the bad sequence in the postscript code and adapt the filter rule in the **ps2eps** script (variables $linefilter, $rangefilter_begin, $rangefilter_end; linefilter is an expression for filtering single lines, rangefilter_... are expressions that filter all lines between a pattern matching $rangefilter_begin and $rangefilter_end; drop me an e-mail with your modifications). However, things may change as the printer drivers (e.g., PScript.dll) or postscript language evolve.

Some applications or drivers generate postscript code with leading or trailing binary code, which often confuses older postscript interpreters. **ps2eps** tries to remove such code, but it may sometimes make a wrong guess about start and end of the real postscript code (drop me an e-mail with a zipped postscript source, see section BUGS).

Comment lines or even blank lines are removed (which is the default to make .eps files smaller), which may corrupt your output. Please check the next section how to fix this. **ps2eps** removes blank lines and also <CR> (carriage ceturn ‘‘\r’’) at the end of lines. However, nicely formatted postscript code gives a hint by using ‘‘%%BeginBinary’’ ‘‘%%EndBinary’’ comments. When **ps2eps** detects these comments it will refrain from any filtering action within the marked binary sections.

ps2eps filters also %%Orientation: comments by default (you can use option **-O** to turn off filtering), because ghostscript may ‘‘automagically’’ rotate images when generating PDF images, which is not desired in most cases. Hint: you can turn off that feature in ghostscript unconditionally by specifying -dAutoRotatePages=/None.

CORRUPTED OUTPUT

Some postscript code may get corrupted when comment lines or even blank lines are removed (which is the default to make .eps files smaller), because those files may contain encoded images which also have a % as first character in a line or use a special comment as end of image

delimiter. If this is the case, use the **-c** option to prevent filtering comments.

COLOR AND MEMORY

ps2eps supports colored postscript, consequently letting ghostscript consume more resources for drawing its bitmap (roughly 6MBytes for an A4 page). **bbox** is reading the bitmap line by line so it consumes only minimal memory. If you experience problems with memory consumption of ghostscript, you may use the **-m** option for using a monochrome image. But this will probably result in wrongly determined bounding boxes with colored images, because ghostscript has to do black/white dithering and may thus suppress objects drawn in light colors.

Another option in case of memory problems and too long run times is to use the much more memory efficient internal ghostscript bbox by using the **-g** option.

ENVIRONMENT VARIABLES

Please note that a command line option always takes precedence over the related environment variable.

The environment variable PS2EPS_SIZE can be used to specify a default page size and take any argument that **--size** accepts. Examples: **export PS2EPS_SIZE=a0** (bash-like syntax) or **setenv PS2EPS_SIZE letter** (csh syntax).

If the environment variable PS2EPS_GSBBOX is set the internal bbox device of ghostscript will be used instead of the external command **bbox**. Examples: **export PS2EPS_GSBBOX=true** (bash-like syntax) or **setenv PS2EPS_GSBBOX 1** (csh syntax).

EXAMPLES

The usual call is simply: **ps2eps -l** *file*

A relatively failsafe call would be (if your postscript is smaller than iso b0 [100cm x 141.4cm] and you have a fast computer with enough memory): **ps2eps -l -B -s b0 -c -n** *file*

If output is not correct try: **ps2eps -l -B -s b0 -F** *file*

AUTHOR

ps2eps was written by Roland Bless.

WHY?

Other programs like **ps2epsi** do not calculate the bounding box always correctly (because the values are put on the postscript stack which may get corrupted by bad postscript code) or rounded it off so that clipping the EPS cut off some part of the image. **ps2eps** uses a double precision resolution of 144 dpi and appropriate rounding to get a proper bounding box. The internal bbox device of ghostscript generates different values (sometimes even incorrect), so using the provided **bbox** should be more robust. However, because normal clipping has only a resolution of 1/72dpi (postscript point), the clipping process may still erase parts of your EPS image. In this case please use the **-l** option to add an additional point of white space around the tight bounding box.

ACKNOWLEDGMENTS

Some people contributed code or suggestions to improve **ps2eps**. Here are at least some names (sorry if I forgot your name): Christophe Druet, Hans Ecke, Berend Hasselman, Erik Joergensen, Koji Nakamaru, Hans Fredrik Nordhaug, Michael Sharpe. Special thanks goes to Michael Sharpe from UCSD who suggested a lot of useful features for ps2eps and who fixed bbox to become more precise and robust.

An earlier version of this manual page was originally written by Rafael Laboissiere <rafael at debian.org> for the Debian system. Thank you Rafael!

BUGS

If you experience problems, please check carefully all hints in the section TROUBLESHOOTING first. Otherwise, check for an updated version at <URL:http://www.tm.uka.de/~bless/ps2eps> or send a gzipped file of relevant postscript source code with your error description and **ps2eps** version number to <roland at bless.de> (please allow some time to reply).

SEE ALSO

bbox (1), gs (1), ps2epsi (1)