NAME

poster − Scale and tile a postscript image to print on multiple pages

SYNOPSIS

poster <options> infile

DESCRIPTION

Poster can be used to create a large poster by building it from multiple pages and/or printing it on large media. It expects as input a generic (encapsulated) postscript file, normally printing on a single page. The output is again a postscript file, maybe containing multiple pages together building the poster. The output pages bear cutmarks and have slightly overlapping images for easier assembling. The input picture will be scaled to obtain the desired size.

The program uses a brute-force method: it copies the entire input file for each output page, hence the output file can be very large. Since the program does not really bother about the input file contents, it clearly works for both black-and-white and color postscript.

To control its operation, you need to specify either the size of the desired poster or a scale factor for the image:

- Given the poster size, it calculates the required number of sheets to print on, and from that a scale factor to fill these sheets optimally with the input image.
- Given a scale factor, it derives the required number of pages from the input image size, and positions the scaled image centered on this area.

Its input file should best be a real ‘Encapsulated Postscript’ file (often denoted with the extension .eps or .epsf). Such files can be generated from about all current drawing applications, and text processors like Word, Interleaf and Framemaker.
However *poster* tries to behave properly also on more relaxed, general postscript files containing a single page definition. Proper operation is obtained for instance on pages generated by (La)TeX and (g)troff.

The media to print on can be selected independently from the input image size and/or the poster size. *Poster* will determine by itself whether it is beneficial to rotate the output image on the media.

To preview the output results of *poster* and/or to (re-)print individual output pages, you should use a postscript previewer like ghostview(1).

OPTIONS

-v Be verbose. Tell about scaling, rotation and number of pages.
Default is silent operation.

-f Ask manual media feed on the plotting/printing device, instead of using its standard paper tray.
Default is adhering to the device settings.

-i <box>
Specify the size of the input image.
Default is reading the image size from the ‘%%BoundingBox’ specification in the input file header.

-m <box>
Specify the desired media size to print on. See below for <box>.
The default is set at compile time, being A4 in the standard package.

-p <box>
Specify the poster size. See below for <box>. Since *poster* will autonomously choose for rotation, always specify a ‘portrait’ poster size (i.e. higher then wide).
If you don’t give the -s option, the default poster size is identical to the media size.

-s <number>
Specify a linear scaling factor to produce the poster. Together with the input image size and optional margins, this induces an output poster size. So don’t specify both -s and -p.
Default is deriving the scale factor to fit a given poster size.

-c <box> *or* -c <number>%
Specify the cut margin. This is the distance between the cutmarks and the paper edge. If the output is really tiled on multiple sheets, the cut marks indicate where to cut the paper for assembly. This margin must be big enough to cover the non-printable margin which almost all printers have. For <box> see below.
Default is 5%. Only when you specify identical poster and media sizes, the default cut margin becomes 0, effectively removing the cutmarks.

-w <box> *or* -w <number>%
Specify a white margin around the output image.
In the ‘ideal’ situation (when an input ‘eps’ file specifies an exact BoundingBox in its header), the output image will be scaled exactly to the edges of the resulting poster (minus cut margin). If you desire a certain margin to remain around the picture after poster assembly, you can specify this with ‘-w’.
(This option is actually redundant, since you can obtain the same result using -s or -i. However some might find this more convenient.)
Default is 0.

-o <outputfile>
Specify the name of the file to write the output into.
(Only added for those poor people who cannot specify output redirection from their command line due to a silly OS.)
Default is writing to standard output.

The <box> mentioned above is a specification of horizontal and vertical size. Only in combination with the ‘-i’ option, the program also understands the offset specification in the <box>.
In general:
<box> = [<multiplier>][<offset>]<unit>
with multipier and offset being specified optionally.
<multiplier> = <number>*<number>
<offset> = +<number>,<number>
<unit> = <medianame> or <distancename>

Many international media names are recognised by the program, in upper and lower case, and can be shortened to their first few characters, as long as unique. For instance ‘A0’, ‘Let’.
Distance names are like ‘cm’, ‘i’, ‘ft’.

EXAMPLES

The following command prints an A4 input file on 8 A3 pages, forming an A0 poster:

```
poster -v -iA4 -mA3 -pA0 infile >outfile
```

The next command prints an eps input image on a poster of 3x3 Letter pages:

```
poster -v -mLet -p3x3Let  image.eps > outfile
```

The next command enlarges an eps input image to print on a large-media A0 capable device, maintaining 2 inch margins:

```
poster -v -mA0 -w2x2i image.eps > outfile
```

Enlarge a postscript image exactly 4 times, print on the default A4 media, and let *poster* determine the number of pages required:

```
poster -v -s4 image.eps > outfile
```

Scale a postscript image to a poster of about 1 square meter, printing on 'Legal' media, maintaining a 10% of 'Legal' size as white margin around the poster.

poster -v -mLegal -p1x1m -w10% infile.ps >outfile

PROBLEMS & QUESTIONS

I get a blurry image and/or interference patterns

If your input file contains -or consists of- pixel images (as opposed to just vector data which is essentially resolution independent), you might have this problem. Such pixel images are normally made to fit well to standard 300 (or 600) dpi devices. Scaling such a picture with an uncarefully chosen factor, can easily lead to hazy edges and interference patterns on the output. The solution is to provide *poster* with an exact scaling factor (with the -s option), chosen as an integer. If integer scaling is unpractical for your purpose, choose a fractional number made from a small integer denominator (2, 3, 4).

Can I select only a small part of a given input picture?

Yes, for this purpose you can define both the size (width and height) and offset (from left and bottom) of a window on the input image. Specify these numbers as argument to a '-i' command line option.
One way to obtain such numbers is previewing the original image with ghostview, and observing the coordinate numbers which it continually displays. These numbers are in postscript units (points), named by *poster* as just 'p'.

Poster doesn't seem to work properly, output pages are empty

The major cause for poster not to work correctly, is giving it postscript files which don't conform to proper 'eps' behaviour. Try whether your application (or printer driver) cannot generate real 'encapsulated postscript'.

If I ask for a 50x50cm poster, it always generates something bigger

Yes, probably. When specifying a desired output size with the '-p' option, *poster* first determines an array of sheets to cover such an area. Then it determines a scale factor for the picture to fill these sheets upto their edge. As result your requested size is used as rough guess only. If you want an exact output size, specify the scaling factor yourself with the '-s' option (and omit the '-p').

I want to keep the white space around the poster as in my original

Poster will as default use the input image bounding box, and scale/translate that to the edges of your poster. If the program which generated your input file specifies an exact and tight %%BoundingBox, you will indeed loose your white margin. To keep the original margin, specify a '-i' option with as argument the papersize on which the original document was formatted (such as '-iA4'). Alternatively specify a smaller scale factor (with -s) or an explicit new margin (with -w).

POSTER ASSEMBLY

Our preferred method for the assembly of a poster from multiple sheets is as follows:

- Arrange the sheets in the proper order on a large table or on the floor.
- Remove from all sheets, except from those in the leftmost column or bottom row, their left and bottom cutmargin.
- In left-to-right and bottom-to-top order, glue the right (and top) cutmargin and stick the right (and upper) neighbouring page on top of that.
- Turn the glued-together poster face bottom, and put adhesive tape on the sheet edges (on the backside of the poster) for more strength.
- Remove the remaining cutmargin around the poster.

DEVICE SETTINGS

For postscript level-2 capable printers/plotters, which is about all modern postscript devices today, *poster* will send device settings in its output file. This consists of a 'setpagedevice' call, setting:

- the media size.
  This is required for all printers I know to get correct behaviour on different media/picture sizes.
- duplexing off.
  Some printers will otherwise perform double-side printing by default. Clearly that is not what you want to print a poster.
- manual media feed.
  This is given only when *poster* was executed with the '-f' command line option. This is a convenient feature if you want to print your job on different media than normally installed in the paper tray, and you are submitting your job through a multi-user networking and spooling environment.

These settings cause proper device behaviour, without the need to manually interact with the printer settings, and has been here locally tested to work on devices like the HP300XL and HP650C.

The settings thus passed in the postscript file, will affect the device for this job only.

DSC CONFORMANCE

Poster will generate its own DSC header and other DSC lines in the output file, according the 'Document Structuring Conventions - version 3.0', as written down in the 'Postscript Language Reference Manual, 2nd ed.' from Adobe Systems Inc, Addison Wesley Publ comp., 1990.

It will copy any '%%Document...' line from the input file DSC header to its own header output. This is used here in particular for required nonresident fonts.

However the copy(s) of the input file included in the output, are stripped from all lines starting with a '%', since they tend to disturb our 'ghostview' previewer and take useless space anyhow.

SEE ALSO

ghostview(1)

CONTRIBUTED BY

Jos van Eijndhoven (email: J.T.J.v.Eijndhoven@ele.tue.nl)
Design Automation Section (http://www.ics.ele.tue.nl)
Dept. of Elec. Eng.
Eindhoven Univ of Technology
The Netherlands
24 August, 1995