

NAME

`jbgtopbm` – JBIG to portable bitmap file converter

SYNOPSIS

`jbgtopbm` [*options*] [*input-file* | - [*output-file*]]

DESCRIPTION

Reads in a *JBIG* bi-level image entity (BIE) from a file or standard input, decompresses it, and outputs a portable bitmap (PBM) file.

JBIG is a highly effective lossless compression algorithm for bi-level images (one bit per pixel), which is particularly suitable for scanned document pages.

A *JBIG* encoded image can be stored in several resolutions in one or several BIEs. All resolution layers except the lowest one are stored efficiently as differences to the next lower resolution layer. Options `-x` and `-y` can be used to stop the decompression at a specified maximal output image size. The input file can consist of several concatenated BIEs which contain different increasing resolution layers of the same image.

If more than one bit per pixel is stored in the JBIG file, then a PGM file will be produced.

OPTIONS

- A single hyphen instead of an input file name will cause *jbgtopbm* to read the data from standard input instead from a file.
- `-x number` Decode only up to the largest resolution layer which is still not more than *number* pixels wide. If no such resolution layer exists, then use the smallest one available.
- `-y number` Decode only up to the largest resolution layer which is still not more than *number* pixels high. If no such resolution layer exists, then use the smallest one available. Options `-x` and `-y` can also be used together in which case the largest layer that satisfies both limits will be selected.
- `-b` Use binary values instead of Gray code words in order to decode pixel values from multiple bitplanes. This option has only an effect if the input has more than one bitplane and a PGM output file is produced. Note that the decoder has to be used in the same mode as the encoder and cannot determine from the BIE, whether Gray or binary code words were used by the encoder.
- `-d` Diagnose a BIE. With this option, *jbgtopbm* will only print a summary of the header information found in the input file and then exit.
- `-p number` If the input contains multiple bitplanes, then extract only the specified single plane as a PBM file. The first plane has number 0.

BUGS

Using standard input and standard output for binary data works only on systems where there is no difference between binary and text streams (e.g., Unix). On other systems (e.g., MS-DOS), using standard input or standard output may cause control characters like CR or LF to be inserted or deleted and this will damage the binary data.

STANDARDS

This program implements the *JBIG* image coding algorithm as specified in ISO/IEC 11544:1993 and ITU-T T.82(1993).

AUTHOR

The *jbgtopbm* program is part of the *JBIG-KIT* package, which has been developed by Markus Kuhn. The most recent version of this portable *JBIG* library and tools set is freely available on the Internet from anonymous ftp server `ftp.informatik.uni-erlangen.de` in directory `pub/doc/ISO/JBIG/`. Bug reports should be sent to `<mkuhn@acm.org>`.

SEE ALSO

`pbm(5)`, `pgm(5)`, `pbmtobjbg(1)`

NAME

`pbmtojbg` – portable bitmap to JBIG file converter

SYNOPSIS

`pbmtojbg` [*options*] [*input-file* | - [*output-file*]]

DESCRIPTION

Reads in a portable bitmap (PBM) from a file or standard input, compresses it, and outputs the image as a *JBIG* bi-level image entity (BIE) file.

JBIG is a highly effective lossless compression algorithm for bi-level images (one bit per pixel), which is particularly suitable for scanned document pages.

A *JBIG* encoded image can be stored in several resolutions (progressive mode). These resolution layers can be stored all in one single BIE or they can be stored in several separate BIE files. All resolution layers except the lowest one are stored merely as differences to the next lower resolution layer, because this requires less space than encoding the full image completely every time. Each resolution layer has twice the number of horizontal and vertical pixels than the next lower layer. *JBIG* files can also store several bits per pixel as separate bitmap planes, and *pbmtojbg* can read a PGM file and transform it into a multi-bitplane BIE.

OPTIONS

- A single hyphen instead of an input file name will cause *pbmtojbg* to read the data from standard input instead from a file.
- q** Encode the image in one single resolution layer (sequential mode). This is usually the most efficient compression method. By default, the number of resolution layers is chosen automatically such that the lowest layer image is not larger than 640×480 pixels.
- x number** Specify the maximal horizontal size of the lowest resolution layer. The default is 640 pixels.
- y number** Specify the maximal vertical size of the lowest resolution layer. The default is 480 pixels.
- l number** Select the lowest resolution layer that will be written to the BIE. It is possible to store the various resolution layers of a *JBIG* image in progressive mode into different BIEs. Options **-l** and **-h** allow to select the resolution-layer interval that will appear in the created BIE. The lowest resolution layer has number 0 and this is also the default value. By default all layers will be written.
- h number** Select the highest resolution layer that will be written to the BIE. By default all layers will be written. See also option **-l**.
- b** Use binary values instead of Gray code words in order to encode pixel values in multiple bitplanes. This option has only an effect if the input is a PGM file and if more than one bitplane is produced. Note that the decoder has to make the same selection but cannot determine from the BIE, whether Gray or binary code words were used by the encoder.
- d number** Specify the total number of differential resolution layers into which the input image will be split in addition to the lowest layer. Each additional layer reduces the size of layer 0 by 50 %. This option overrides options **-x** and **-y** which are usually a more comfortable way of selecting the number of resolution layers.
- s number** The *JBIG* algorithm splits each image into a number of horizontal stripes. This option specifies that each stripe shall have *number* lines in layer 0. The default value is selected so that approximately 35 stripes will be used for the whole image.
- m number** Select the maximum horizontal offset of the adaptive template pixel. The *JBIG* encoder uses a number of neighbour pixels in order to get statistical a priori knowledge of the probability, whether the next pixel will be black or white. One single pixel out of this template of context neighbor pixels can be moved around. Especially for dithered images it

can be a significant advantage to have one neighbor pixel which has a distance large enough to cover the period of a dither function. By default, the adaptive template pixel can be moved up to 8 pixels away. This encoder supports up to 23 pixels, however as decoders are only required to support at least a distance of 16 pixels by the standard, no higher value than 16 for *number* is recommended in order to maintain interoperability with other *JBIG* implementations. The maximal vertical offset of the adaptive template pixel is always zero.

-t *number* Encode only the specified number of most significant bit planes. This option allows to reduce the depth of an input PGM file if not all bits per pixel are needed in the output.

-o *number* *JBIG* separates an image into several horizontal stripes, resolution layers and planes, where each plane contains one bit per pixel. One single stripe in one plane and layer is encoded as a data unit called stripe data entity (SDE) inside the BIE. There are 12 different possible orders in which the SDEs can be stored inside the BIE and *number* selects which one shall be used. The order of the SDEs is only relevant for applications that want to decode a *JBIG* file which has not yet completely arrived from e.g. a slow network connection. For instance some applications prefer that the outermost of the three loops (stripes, layers, planes) is over all layers so that all data of the lowest resolution layer is transmitted first. The following values for *number* select these loop arrangements for writing the SDEs (outermost loop first):

0	planes, layers, stripes
2	layers, planes, stripes
3	layers, stripes, planes
4	stripes, planes, layers
5	planes, stripes, layers
6	stripes, layers, planes

All loops count starting with zero, however by adding 8 to the above order code, the layer loop can be reversed so that it counts down to zero and then higher resolution layers will be stored before lower layers. Default order is 3 which writes at first all planes of the first stripe and then completes layer 0 before continuing with the next layer and so on.

-p *number* This option allows to activate or deactivate various optional algorithms defined in the *JBIG* standard. Just add the numbers of the following options which you want to activate in order to get the *number* value:

4	deterministic prediction (DPON)
8	layer 0 typical prediction (TPBON)
16	diff. layer typ. pred. (TPDON)
64	layer 0 two-line template (LRLTWO)

Except for special applications (like communication with *JBIG* subset implementations) and for debugging purposes you will normally not want to change anything here. The default is 28, which provides the best compression result.

-c The adaptive template pixel movement is determined as suggested in annex C of the standard. By default the template change takes place directly in the next line which is most effective. However a few conformance test examples in the standard require the adaptive template change to be delayed until the first line of the next stripe. This option selects this special behavior, which is normally not required except in order to pass some conformance test suite.

-v After the BIE has been created, a few technical details of the created file will be listed (verbose mode).

BUGS

Using standard input and standard output for binary data works only on systems where there is no difference between binary and text streams (e.g., Unix). On other systems (e.g., MS-DOS), using standard input or standard output may cause control characters like CR or LF to be inserted or deleted and this will damage the binary data.

STANDARDS

This program implements the *JBIG* image coding algorithm as specified in ISO/IEC 11544:1993 and ITU-T T.82(1993).

AUTHOR

The *pbmtojbg* program is part of the *JBIG-KIT* package, which has been developed by Markus Kuhn. The most recent version of this portable *JBIG* library and tools set is freely available on the Internet from anonymous ftp server ftp.informatik.uni-erlangen.de in directory pub/doc/ISO/JBIG/. Bug reports should be sent to <mkuhn@acm.org>.

SEE ALSO

pbm(5), pgm(5), jbgtopbm(1)

NAME

pbm - portable bitmap file format

DESCRIPTION

The portable bitmap format is a lowest common denominator monochrome file format. It was originally designed to make it reasonable to mail bitmaps between different types of machines using the typical stupid network mailers we have today. Now it serves as the common language of a large family of bitmap conversion filters. The definition is as follows:

- A "magic number" for identifying the file type. A pbm file's magic number is the two characters "P1".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- Width * height bits, each either '1' or '0', starting at the top-left corner of the bitmap, proceeding in normal English reading order.
- The character '1' means black, '0' means white.
- Whitespace in the bits section is ignored.
- Characters from a "#" to the next end-of-line are ignored (comments).
- No line should be longer than 70 characters.

Here is an example of a small bitmap in this format:

```
P1
# feep.pbm
24 7
00000000000000000000000000000000
011110011110011110011110011110
010000010000010000010000010010
011100011100011100011100011110
010000010000010000010000010000
010000011110011110010000010000
00000000000000000000000000000000
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a bitmap.

There is also a variant on the format, available by setting the RAWBITS option at compile time. This variant is different in the following ways:

- The "magic number" is "P4" instead of "P1".
- The bits are stored eight per byte, high bit first low bit last.
- No whitespace is allowed in the bits section, and only a single character of whitespace (typically a new-line) is allowed after the height.
- The files are eight times smaller and many times faster to read and write.

SEE ALSO

atktopbm(1), brushtopbm(1), cmuwmtopbm(1), g3topbm(1), gemtopbm(1), icontopbm(1), macptopbm(1), mgrtopbm(1), pi3topbm(1), xbmtopbm(1), ybmtopbm(1), pbmto10x(1), pnmtascii(1), pbmtoatk(1), pbmtoatkg(1), pbmtoatkg(1), pbmtocmuwm(1), pbmtoepson(1), pbmtog3(1), pbmtogem(1), pbmtogo(1), pbmtoicon(1), pbmtolj(1), pbmtomacp(1), pbmtomgr(1), pbmtopi3(1), pbmtoplot(1), pbmtoptx(1), pbmtox10bm(1), pbmtoxbm(1), pbmtoybm(1), pbmtozinc(1), pbmlife(1), pbmmake(1), pbmmask(1), pbmreduce(1), pbmtext(1), pbmupc(1), pnm(5), pgm(5), ppm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pgm - portable graymap file format

DESCRIPTION

The portable graymap format is a lowest common denominator grayscale file format. The definition is as follows:

- A "magic number" for identifying the file type. A pgm file's magic number is the two characters "P2".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- The maximum gray value, again in ASCII decimal.
- Whitespace.
- Width * height gray values, each in ASCII decimal, between 0 and the specified maximum value, separated by whitespace, starting at the top-left corner of the graymap, proceeding in normal English reading order. A value of 0 means black, and the maximum value means white.
- Characters from a "#" to the next end-of-line are ignored (comments).
- No line should be longer than 70 characters.

Here is an example of a small graymap in this format:

```
P2
# feep.pgm
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a graymap.

There is also a variant on the format, available by setting the RAWBITS option at compile time. This variant is different in the following ways:

- The "magic number" is "P5" instead of "P2".
- The gray values are stored as plain bytes, instead of ASCII decimal.
- No whitespace is allowed in the grays section, and only a single character of whitespace (typically a newline) is allowed after the maxval.
- The files are smaller and many times faster to read and write.

Note that this raw format can only be used for maxvals less than or equal to 255. If you use the *pgm* library and try to write a file with a larger maxval, it will automatically fall back on the slower but more general plain format.

SEE ALSO

fitstopgm(1), fstopgm(1), hipstopgm(1), lispmtopgm(1), psidtopgm(1), rawtopgm(1), pgmbentley(1), pgm-crater(1), pgmedge(1), pgmenhance(1), pgmhist(1), pgmnorm(1), pgmoil(1), pgmramp(1), pgmtexture(1), pgmtofits(1), pgmtof(1), pgmtolisp(1), pgmtopbm(1), pnm(5), pbm(5), ppm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.