NAME

pnmtojpeg − convert PNM image to a JFIF ("JPEG") image

SYNOPSIS

pnmtojpeg [*options*] [*filename*]

DESCRIPTION

pnmtojpeg converts the named PBM, PGM, or PPM image file, or the standard input if no file is named, to a JFIF file on the standard output.

pnmtojpeg uses the Independent JPEG Group's JPEG library to create the output file. See **http://www.ijg.org** for information on the library.

"JFIF" is the correct name for the image format commonly known as "JPEG." Strictly speaking, JPEG is a method of compression. The image format using JPEG compression that is by far the most common is JFIF. There is also a subformat of TIFF that uses JPEG compression.

EXIF is an image format that is a subformat of JFIF (to wit, a JFIF file that contains an EXIF header as an APP1 marker). **pnmtojpeg** creates an EXIF image when you specify the **-exif** option.

OPTIONS

The basic options are:

−−exif=*filespec*

This option specifies that the output image is to be EXIF (a subformat of JFIF), i.e. it will have an EXIF header as a JFIF APP1 marker. The contents of that marker are the contents of the specified file. The special value **−** means to read the EXIF header contents from standard input. It is invalid to specify standard input for both the EXIF header and the input image.

The EXIF file starts with a two byte field which is the length of the file, including the length field, in pure binary, most significant byte first. The special value of zero for the length field means there is to be no EXIF header, i.e. the same as no **-exif** option. This is useful for when you convert a file from JFIF to PNM using **jpegtopnm**, then transform it, then convert it back to JFIF with **pnmtojpeg**, and you don't know whether or not it includes an EXIF header. **jpegtopnm** creates an EXIF file containing nothing but two bytes of zero when the input JFIF file has no EXIF header. Thus, you can transfer any EXIF header from the input JFIF to the output JFIF without worrying about whether an EXIF header actually exists.

The contents of the EXIF file after the length field are the exact byte for byte contents of the APP1 marker, not counting the length field, that constitutes the EXIF header.

−−quality=*n*

Scale quantization tables to adjust image quality. *n* is 0 (worst) to 100 (best); default is 75. (See below for more info.)

−−grayscale

−−greyscale

Create gray scale JFIF file. With this option, **pnmtojpeg** converts color input to gray scale. If you don't specify this option, The output file is in color format if the input is PPM, and grayscale format if the input is PBM or PGM.

In the PPM input case, even if all the colors in the image are gray, the output is in color format. Of course, the colors in it are still gray. The difference is that color format takes up a lot more space and takes longer to create and process.

−−optimize

Perform optimization of entropy encoding parameters. Without this, **pnmtojpeg** uses default encoding parameters. **−−optimize** usually makes the JFIF file a little smaller, but **pnmtojpeg** runs somewhat slower and needs much more memory. Image quality and speed of decompression are unaffected by **−−optimize**.

−−progressive

Create a progressive JPEG file (see below).

−−comment=*text*

Include a comment marker in the JFIF output, with comment text *text*. Without this option, there are no comment markers in the output.

The **−−quality** option lets you trade off compressed file size against quality of the reconstructed image: the higher the quality setting, the larger the JFIF file, and the closer the output image will be to the original input. Normally you want to use the lowest quality setting (smallest file) that decompresses into something visually indistinguishable from the original image. For this purpose the quality setting should be between 50 and 95; the default of 75 is often about right. If you see defects at **−−quality=75**, then go up 5 or 10 counts at a time until you are happy with the output image. (The optimal setting will vary from one image to another.)

−−quality=100 generates a quantization table of all 1's, minimizing loss in the quantization step (but there is still information loss in subsampling, as well as roundoff error). This setting is mainly of interest for experimental purposes. Quality values above about 95 are *not* recommended for normal use; the compressed file size goes up dramatically for hardly any gain in output image quality.

In the other direction, quality values below 50 will produce very small files of low image quality. Settings around 5 to 10 might be useful in preparing an index of a large image library, for example. Try **−−quality=2** (or so) for some amusing Cubist effects. (Note: quality values below about 25 generate 2-byte quantization tables, which are considered optional in the JFIF standard. **pnmtojpeg** emits a warning message when you give such a quality value, because some other JFIF programs may be unable to decode the resulting file. Use **−−baseline** if you need to ensure compatibility at low quality values.)

The **−−progressive** option creates a "progressive JPEG" file. In this type of JFIF file, the data is stored in multiple scans of increasing quality. If the file is being transmitted over a slow communications link, the decoder can use the first scan to display a low-quality image very quickly, and can then improve the display with each subsequent scan. The final image is exactly equivalent to a standard JFIF file of the same quality setting, and the total file size is about the same -- often a little smaller. **Caution:** progressive JPEG is not yet widely implemented, so many decoders will be unable to view a progressive JPEG file at all.

Options for advanced users:

−−dct=int

Use integer DCT method (default).

−−dct=fast

Use fast integer DCT (less accurate).

−−dct=float

Use floating-point DCT method. The float method is very slightly more accurate than the int method, but is much slower unless your machine has very fast floating-point hardware. Also note that results of the floating-point method may vary slightly across machines, while the integer methods should give the same results everywhere. The fast integer method is much less accurate than the other two.

−−restart=*n*

Emit a JPEG restart marker every *n* MCU rows, or every *n* MCU blocks if you append **B** to the number. **−−restart 0** (the default) means no restart markers.

−−smooth=*n*

Smooth the input image to eliminate dithering noise. *n*, ranging from 1 to 100, indicates the strength of smoothing. 0 (the default) means no smoothing.

−−maxmemory=*n*

Set a limit for amount of memory to use in processing large images. Value is in thousands of bytes, or millions of bytes if you append **M** to the number. For example, **−−max=4m** selects 4,000,000 bytes. If **pnmtojpeg** needs more space, it will use temporary files.

−−verbose

Print to the Standard Error file messages about the conversion process. This can be helpful in debugging problems.

The **−−restart** option tells **pnmtojpeg** to insert extra markers that allow a JPEG decoder to resynchronize after a transmission error. Without restart markers, any damage to a compressed file will usually ruin the image from the point of the error to the end of the image; with restart markers, the damage is usually confined to the portion of the image up to the next restart marker. Of course, the restart markers occupy extra space. We recommend **−−restart=1** for images that will be transmitted across unreliable networks such as Usenet.

The **−−smooth** option filters the input to eliminate fine-scale noise. This is often useful when converting dithered images to JFIF: a moderate smoothing factor of 10 to 50 gets rid of dithering patterns in the input file, resulting in a smaller JFIF file and a better-looking image. Too large a smoothing factor will visibly blur the image, however.

Options for wizards:

−−baseline

Force baseline-compatible quantization tables to be generated. This clamps quantization values to 8 bits even at low quality settings. (This switch is poorly named, since it does not ensure that the output is actually baseline JPEG. For example, you can use **−−baseline** and **−−progressive** together.)

−−qtables=*filespec*

Use the quantization tables given in the specified text file.

−−qslots=n[,...]

Select which quantization table to use for each color component.

−−sample=*HxV[,...]*

Set JPEG sampling factors for each color component.

−−scans=*filespec*

Use the scan script given in the specified text file. See below for information on scan scripts.

The "wizard" options are intended for experimentation with JPEG. If you don't know what you are doing, **don't use them**. These switches are documented further in the file wizard.doc that comes with the Independent JPEG Group's JPEG library.

EXAMPLES

This example compresses the PPM file foo.ppm with a quality factor of 60 and saves the output as foo.jpg:

pnmtojpeg −−quality=60 foo.ppm > foo.jpg

cat foo.bmp | bmptoppm | pnmtojpeg > foo.jpg

HINTS

JFIF is not ideal for cartoons, line drawings, and other images that have only a few distinct colors. For those, try instead **pnmtopng** or **ppmtobmp**. If you need to convert such an image to JFIF, though, you should experiment with **pnmtojpeg**'s **−−quality** and **−−smooth** options to get a satisfactory conversion. **−−smooth 10** or so is often helpful.

JPEG compression is notable for being a "lossy." This means that, unlike with most graphics conversions,

you lose information, which means image quality, when you convert to JFIF. If you convert from PPM to JFIF and back repeatedly, image quality loss will accumulate. After ten or so cycles the image may be noticeably worse than it was after one cycle.

Because of this, you should do all the manipulation you have to do on the image in some other format and convert to JFIF as the last step. And if you can keep a copy in the original format, so much the better. PNG is a good choice for a format that is lossless, yet fairly compact. GIF is another way to go, but chances are you can't create a GIF image without owing a lot of money to Unisys and IBM, holders of patents on the LZW compression used in the GIF format.

The **−−optimize** option to **pnmtojpeg** is worth using when you are making a "final" version for posting or archiving. It's also a win when you are using low quality settings to make very small JFIF files; the percentage improvement is often a lot more than it is on larger files. (At present, **−−optimize** mode is automatically in effect when you generate a progressive JPEG file).

Another program, **cjpeg**, is similar. **cjpeg** is maintained by the Independent JPEG Group and packaged with the JPEG library which **pnmtojpeg** uses for all its JPEG work. Because of that, you may expect it to exploit more current JPEG features. Also, since you have to have the library to run **pnmtojpeg**, but not vice versa, **cjpeg** may be more commonly available.

On the other hand, **cjpeg** does not use the NetPBM libraries to process its input, as all the NetPBM tools such as **pnmtojpeg** do. This means it is less likely to be consistent with all the other programs that deal with the NetPBM formats. Also, the command syntax of **pnmtojpeg** is consistent with that of the other Netpbm tools, unlike **cjpeg**.

SCAN SCRIPTS

Use the **-scan** option to specify a scan script. Or use the **-progressive** option to specify a particular built-in scan script.

Just what a scan script is, and the basic format of the scan script file, is covered in the **wizard.doc** file that comes with the Independent JPEG Group's JPEG library. Scan scripts are same for **pnmtojpeg** as the are for **cjpeg**.

This section contains additional information that isn't, but probably should be, in that document.

First, there are many restrictions on what is a valid scan script. The JPEG library, and thus **pnmtojpeg**, checks thoroughly for any lack of compliance with these restrictions, but does little to tell you how the script fails to comply. The messages are very general and sometimes untrue.

To start with, the entries for the DC coefficient must come before any entries for the AC coefficients. The DC coefficient is Coefficient 0; all the other coefficients are AC coefficients. So in an entry for the DC coefficient, the two numbers after the colon must be 0 and 0. In an entry for AC coefficients, the first number after the colon must not be 0.

In a DC entry, the color components must be in increasing order. E.g. "0,2,1" before the colon is wrong. So is "0,0,0".

In an entry for an AC coeffient, you must specify only one color component. I.e. there can be only one number before the colon.

In the first entry for a particular coefficient for a particular color component, the "Ah" value must be zero, but the Al value can be any valid bit number. In subsequent entries, Ah must be the Al value from the previous entry (for that coefficient for that color component), and the Al value must be one less than the Ah value.

The script must ultimately specify at least some of the DC coefficent for every color component.

Otherwise, you get the error message "Script does not transmit all the data." You need not specify all of the bits of the DC coefficient, or any of the AC coefficients.

There is a standard option in building the JPEG library to omit scan script capability. If for some reason your library was built with this option, you get the message "Requested feature was omitted at compile time."

ENVIRONMENT

JPEGMEM

If this environment variable is set, its value is the default memory limit. The value is specified as described for the **--maxmemory** option. An explicit **--maxmemory** option overrides any **JPEGMEM**.

SEE ALSO

cjpeg(1), **djpeg**(1), **jpegtran**(1), **rdjpgcom**(1), **wrjpgcom**(1)
ppm(5), **pgm**(5), **jpegtopnm**(1)
Wallace, Gregory K. "The JPEG Still Picture Compression Standard", Communications of the ACM, April 1991 (vol. 34, no. 4), pp. 30-44.

LIMITATIONS

Arithmetic coding is not supported for legal reasons.

The program could be much faster.

AUTHOR

pnmtojpeg and this man page were derived in large part from **cjpeg**, by the Independent JPEG Group. The program is otherwise by Bryan Henderson on March 07, 2000.