NAME

flif − convert a image files from or to FLIF

SYNOPSIS

flif [*OPTION*]... [*-d*] *input_file.flif output_file*
flif [*OPTION*]... [*-e*] *input_file... output_file.flif*
flif [*OPTION*]... [*-t*] *input_file.flif output_file.flif*

DESCRIPTION

This program can encode and decode images in the **Free Lossless Image Format** (FLIF). Input and output formats can be either PNG, PAM, PNM (PPM, PGM, PBM) or FLIF.

flif will automatically deduce the action (encode, decode or transcode) from the names and contents of the files. The specifiers *-e* (encode), *-d* (decode) and *-t* (transcode) are optional and can be safely omitted. The extension of the output file (the last argument) determines the image format that is used.

OPTIONS

The general options are:

−v, **−−verbose**
: Increases verbosity. By default, **flif** is silent (unless there is a problem). Adding one or more **-v** will produce more output.

−h, **−−help**
: Display help and quit. Use in combination with **-v** for advanced help.

DECODING

To decode a FLIF image, the output filename must have one of the following extensions:

.png Portable Network Graphics (PNG)

.pnm Portable anymap (PNM). If the FLIF image has transparency (alpha channel), it will be discarded. If the image is a grayscale image, a PGM (P5) will be produced; otherwise a PPM (P6) is produced.

.pam Portable arbitrary map (PAM). If the FLIF image has an alpha channel, a PAM (P7) is produced; otherwise a PPM (P6) or PGM (P5) is produced.

.rggb This is a non-standard format which is used to represent raw camera image with RGGB data (i.e. raw Bayer CFA sensor data). It is actually a PGM image as produced by **dcraw -E -4**.

Alternatively, the special output filename **null:** can be used if the decoded output does not need to be saved. This might be useful to verify the integrity of a FLIF image, or to benchmark the decode time.

The following decode options are available:

−q, **−−quality**=*PERCENT*
: FLIF is a lossless format, but it allows you to do a 'lossy decoding' (i.e. progressive loading), where only the beginning of the file is read (i.e. a partially downloaded FLIF image). The default is **-q***100* (lossless decoding). The percentage corresponds to the minimal proportion of subpixels that has to be decoded.
This option is most effective for interlaced FLIF files. However, it can also be used on non-interlaced FLIF files; in that case the percentage corresponds to the proportion of color channels to decode: e.g. on RGB images (with the YCoCg color transform enabled, which is the default), **-q***33* (or lower) produces a more or less grayscale image (only Y); quality values between 34 and 66 produce a color image with only the orange-blue chroma channel decoded (only Y and Co); values above 67 correspond to a lossless decoding, including the green-magenta chroma channel (Y, Co and Cg).

−s, **−−scale**=*FACTOR*

For large images, in many use cases it suffices to load only a scaled-down version of the image, e.g. because the resolution of the target canvas is (much) lower than the image resolution. This option can be used to do a 'lossy decode', like **-q**. It automatically selects a quality percentage suitable for the desired scale-down factor. For an input FLIF image of dimensions **width** x **height**, the resulting output image will have dimensions **width**/*FACTOR* x **height**/*FACTOR*. Only powers of two (1,2,4,8,16,32,...) are allowed as scale-down factors.

The scaled-down image is lossy in two ways: all subpixels are subsampled, not averaged (which may result in aliasing artifacts), and only the (alpha and) luma channels are decoded at the full scaled-down resolution; the chroma channels are relatively subsampled (which may result in chroma blockiness).

This option is only effective for interlaced FLIF files.

−r, **−−resize**=*WIDTH*[**x***HEIGHT*]

Automatically pick a scale-down factor such that the decoded downscaled image fits in target rectangle of *WIDTH* by *HEIGHT* pixels. Note that the actual decoded output can be significantly smaller than this: the image aspect ratio is respected, and only power-of-two scale-down factors are used. If the aspect ratio of the target rectangle is equal to the image aspect ratio, then the width of the decoded image is between *WIDTH*/2 and *WIDTH*, and the height of the decoded image is between *HEIGHT*/2 and *HEIGHT*.

If *HEIGHT* is omitted, it is assumed to be equal to *WIDTH*.

In order to obtain a high-quality scaled-down image that fits a given target rectangle exactly, it is recommended to pick a **−−resize** dimension which is at least two times higher than this target, and use a downscaling method of choice to scale down from that.

This option is only effective for interlaced FLIF files.

−i, **−−identify**

Do not fully decode the input FLIF file, just decode its header and output some metadata like dimensions and color depth. You can specify multiple input files when this option is used.

ENCODING

To encode an image to FLIF, the input file(s) can be in any of the output formats supported by the decoder: PNG, PAM, or PNM. Additionally it is possible to transcode from FLIF to FLIF, which might be useful when using non-default encoder settings.

All of the encode options are optional; the defaults are supposed to be OK. If progressive decoding is not useful in your particular use case, then **−N** might be a good idea, but other than that, it is recommended to simply use the default settings, since all of these options can easily lead to worse compression.

The following encode options are available:

−E, **−−effort**=*EFFORT*

How much effort to spend on compression, as expressed on a scale from 0 (nearly no effort) to 100 (a lot of effort). The default is **−E***60*. At the moment, most of the values of *EFFORT* result in the same internal configuration; interesting values to try are: **−E***0*, **−E***5*, **−E***10*, **−E***20*, **−E***30*, **−E***60*, **−E***80*, **−E***100*. There is no guarantee that higher values for *EFFORT* actually result in smaller files: it is very much possible that a file encoded with **−E***100* is larger than one encoded with **−E***10*. There is a guarantee, however, that more time will be spent for higher values of *EFFORT*. On average, higher values for *EFFORT* do result in smaller files, though the difference in file size between **−E***10* and **−E***100* is typically small, while the difference in encode time is significant.

This option is no substitute for brute-force optimization: even at **−E***100*, there is only one encoding performed. External tools that try many encodings can achieve smaller sizes than **−E***100*. This option is mosty useful to reduce the encode time without really harming compression much.

−I, **−−interlace**

Force the resulting image to be interlaced (also known as 'progressive'). This is the default setting, except for very small images (currently defined as 'less than 10,000 pixels'), where progressive decoding would be of little use.

−N, **−−no−interlace**
Force the resulting image to be non-interlaced (also known as 'scanlines'). By default, **flif** will produce interlaced files. Non-interlaced files tend to be (slightly) smaller for most image types, but they cannot be decoded progressively.

−Q, **−−lossy**=*QUALITY*
FLIF is a lossless format, but if you want to, you can use this option to modify the image before encoding it, in such a way that it compresses better. The parameter *QUALITY* indicates the desired quality, where 100 is lossless and 0 is very lossy.

−U, **−−adaptive**
By default, **-Q** treats every pixel the same (the same amount of loss is allowed). This option can be used for adaptive lossy encoding. You have to create a saliency map that indicates the regions of interest that should be stored with less loss. The saliency map has to be a grayscale image of the same dimensions as the input image: black means lossy (maximum loss depends on **-Q**), white means lossless, and intermediate values are intermediate levels of lossiness. To create the saliency map, you can use external tools like SaliencyDetector (https://github.com/technopagan/mss-saliency). In practice, you may want to darken the saliency map to avoid fully lossless storage. Here is an example:
flif -Q50 -U input-image.png saliency-map.png output.flif

−K, **−−keep−invisible−rgb**
By default, pixels that are fully transparent have undefined RGB values in a FLIF image, since those values are irrelevant for nearly all purposes. If you insist on storing the RGB values hidden behind A=0, use this option. In rare cases this can lead to better compression.

ADVANCED ENCODE OPTIONS

The options below can be used to manually tune some encoder parameters in order to try to get (slightly) better compression.

−P, **−−max−palette−size**=*NB_COLORS*
Images which use relatively few different colors, e.g. ex-GIF images, can be compressed better using a palette of colors instead of the full RGB(A) color space. By default, **flif** uses a palette if the image has less than 512 distinct colors. With this option, you can adjust this threshold. In particular, **-P***0* disables the use of a color palette.
There are two kinds of palettes: Palette_Alpha contains RGBA colors, while Palette contains RGB colors. On images with transparency, it can be the case that there are more than *NB_COLORS* distinct RGBA colors, but less than *NB_COLORS* distinct RGB colors; in that case the Alpha channel gets encoded separately and the Palette transform is used.
By default, **flif** orders the palette in lexicographical order on the transformed color values -- typically (Y,Co,Cg) or (Alpha,Y,Co,Cg). If *NB_COLORS* is a negative number, then the palette is not ordered and the colors are added in the order in which they appear in the image (in scanline order). In that case, the maximum palette size is the absolute value of *NB_COLORS*.

−A, **−−force−color−buckets**
For images which use relatively few different colors, but more than what would fit in a color palette, FLIF implements the Color_Buckets transform to improve compression. By default, **flif** uses a heuristic to decide whether or not to use Color_Buckets. With this option, Color_Buckets is forced on, unless the image is a grayscale image or uses a palette (so to use color buckets instead of a palette, use **−AP***0*.

−B, **−−no−color−buckets**
Similar to **−A**, this option overrides the heuristic and forces Color_Buckets to be disabled.

−C, **−−no−channel−compact**
This option disables the Channel_Compact transform. This transformation reduces the domain of each channel to eliminate unused values. While this typically results in better compression, it is by no means necessarily the case.

−Y, **−−no−ycocg**

This option disables the YCoCg color transform. This color space transform is aimed at decorrelating the RGB channels, and usually leads to better compression. It also helps to improve the quality of progressive decoding, by encoding the most important Y channel earlier than the chroma channels.

−G, **−−guess**=*METHOD*[*METHOD*]...

Interlaced FLIF can use different pixel prediction (guess) methods. By default, the encoder uses a simple heuristic to automatically pick a good method. This option lets you manually override that choice.

−G*0* uses the average of top and bottom (H) or left and right (V);

−G*1* uses the median of top+left-topleft, bottom+left-bottomleft (H) or top+right-topright (V), and the **−G***0* guess;

−G*2* uses the median of top, left, and bottom (H) or right (V);

−G*?* picks one of the above predictors, depending on some heuristic. This is the default setting.

−G*3* uses different predictors (any of the above) for each plane/zoomlevel, depending on some heuristic. This is usually a bad idea.

For photographs, **−G***0* tends to be better, while for line art, **−G***1* or **−G***2* are usually best. You can specify the pixel predictor separately for each plane (Y,Co,Cg,A). Unspecified predictors are set to the Y plane predictor. So for example **−G***0?2* means: use predictor 0 for the Y plane, an automatically chosen predictor for the Co plane, predictor 2 for the Cg plane, and if there's an alpha channel, use predictor 0 (the same as for the Y plane).

−H, **−−invisible−guess**=*METHOD*

Interlaced FLIF with an alpha channel can use different pixel prediction methods to define the RGB values of invisible (A=0) pixels. This can have a (small) effect on compression. The available methods are **−H***0*, **−H***1*, and **−H***2*, which have the same meaning as in the option **−G** (see above). The default method is **−H***2*.

−R, **−−maniac−repeats**=*NB_ITERATIONS*

The first and computationally most demanding step of FLIF encoding is performing a number of iterations of dummy-encoding in order to learn image-adapted MANIAC trees. More iterations will result in larger and better MANIAC trees, resulting in better compression. However, since the trees themselves are part of the compressed file, too many iterations will result in worse overall compression. Also, larger MANIAC trees do have a (slight) negative impact on decode speed. The default value **−R***2* tends to be near the optimum, but usually **−R***3*, **−R***4* or **−R***5* produces a slightly smaller compressed file (at the cost of a longer encode time). For fast encoding without MANIAC trees, use **−R***0*.

−T, **−−maniac_threshold**=*BITS*

While constructing a MANIAC tree, a leaf node turns into a decision node (i.e. it splits into two new leaf nodes) when a certain threshold is reached. This threshold can be expressed in the hypothetical number of bits that would have been saved so far if the node would have been split from the beginning. The default setting is **−T***40* (i.e. 5 bytes). Lower values will cause the MANIAC trees to be more eagerly grown, thus the trees get larger and potentially more 'noisy'. Higher values will result in smaller trees, and potentially less adaptation to the image (so worse compression).

−D, **−−maniac−divisor**=*DIV*

After constructing a MANIAC trees, a simple post-processing step takes place. Each inner node in the MANIAC tree contains a counter which determines when the node gets split during the actual encoding or decoding. During learning, the nodes are always split 'too late' (that is, after the split threshold has already been reached). Therefore, the counters are divided by some fixed constant, with the goal of make sure that during actual encoding, the splitting takes place 'early enough'. However, decreasing the counters too much (i.e. a value of *DIV* that is too high) means that the AC contexts in the inner nodes have no time to adjust, leading to worse compression. The default setting is **−D***30*.

−M, **−−maniac−min-size**=*SIZE*

Also as part of the post-processing step after constructing the MANIAC trees, some pruning takes place in order to reduce the size of the trees (which is important since they are part of the compressed file). The pruning will remove leaf nodes and subtrees that are not frequently visited, i.e. the sum of the counters in the subtree is small. As a result these contexts will be merged with the one of the parent node. This option controls the threshold at which such pruning is done. The default setting is **−M***50*, which roughly means that subtrees are pruned if they are used for less than 50/NB_ITERATIONS subpixels.

−X, **−−chance-cutoff**=*CUTOFF*

The entropy coding ultimately outputs bits according to some adaptive chance. Chances are represented as 12-bit numbers which represent a rational number of the form *x*/4096. The lowest possible chance is set at *CUTOFF*/4096, the highest possible chance is (4096-*CUTOFF*)/4096. The default value is **−X***2*. If you have an input image that is extremely predictable, you may want to try **−X***1*, which allows chances to converge to more extreme values, resulting in even better compression. If however the input is rather noisy, you could use a higher value like **−X***20* to limit the cost of bad prediction. (If the input is very noisy, it may be better to not try to compress it in the first place.)

−Z, **−−chance-alpha**=*ALPHA*

The chance adaptation in the entropy coding uses this parameter to control how rapidly the chance is allowed to change. If it changes too rapidly, it will fluctuate wildly around the optimal chance instead of converging to it. If it changes too slowly, it will not compress well because it takes too long to adapt. The default value is **−Z***19*

ANIMATION

FLIF supports animation, so if multiple input files are given, an animated FLIF file will be produced where each input image corresponds to one frame of the animation. All input images need to have the exact same dimensions (width, height, number of color channels and color depth). All input frames are interpreted as complete frames (’replace mode’); there is no notion of ’combine mode’ frames. In other words, transparent pixels are always transparent, they do not combine with the pixels from the previous frame.

When decoding an animated FLIF file, multiple output images will be produced. The filenames of the decoded output images are constructed as follows: if the output filename is *filename.ext*, then the actual output files are *filename*-**000**.*ext*, *filename*-**001**.*ext*, *filename*-**002**.*ext*, ..., *filename*-<**nb_frames - 1**>.*ext*.

Options specific to encoding (or transcoding) animations are as follows:

−F, **−−frame−delay**=*DELAY*[,*DELAY*]...

The time between two consecutive frames of the animation, in milliseconds. The default setting is **−f***100* (100ms for all frames), which corresponds to 10 frames per second. If multiple delays are given, each number corresponds to the duration of one frame. In case the number of delays is smaller than the number of frames, the last number is repeated implicitly.

−L, **−−max−frame−lookback**=*NB_FRAMES*

In animations, typically the frames are somewhat similar. To improve compression, FLIF does a generalization of ’combine mode’: it will look back at most *NB_FRAMES* frames to ’reuse’ pixels. This transformation is called Frame_Lookback. Using **−L***0*, the method can be disabled. It does not make sense to use a value larger than the number of frames in the animation minus one. The default setting is **−L***1*. Different values can result in better or worse compression.

−S, **−−no−frame−shape**

By default, the Frame_Shape transform is enabled. The shape of a frame is described row-by-row, so it is more general than a simple bounding box (e.g. it could also be a sphere or triangle). However, if the shape of the changed pixels is not convex, and if Frame_Lookback is also activated (which is the default setting), Frame_Shape does not always produce smaller files. This option can be used to disable the Frame_Shape transform.

BUGS

Please report all bugs or feature requests to our issue tracker: http://github.com/FLIF-hub/FLIF/issues/

EXAMPLES

flif picture.png picture.flif

Encode the PNG file **picture.png** to a FLIF file **picture.flif**

flif frame-*.png -F40 -L10 animation.flif

Encode a sequence of PNG files (**frame-*.png**) to an animated FLIF file **animation.flif**, with a delay of 40ms between each frame (25 frames per second), using a frame lookback of 10 frames.

flif -q50 animation.flif decoded_frame.pam

Decode the FLIF animation **animation.flif** at quality 50%, to a series of Portable AnyMap files **decoded_frame-000.pam**, **decoded_frame-001.pam**, **decoded_frame-002.pam**, ...

flif -s2 animation.flif -NAP0 -F50 -L3 -R2 -T38 -D32 -M70 animation_downscaled_and_tweaked.flif

Transcode the FLIF animation **animation.flif**, scaling down by a factor of two, using a non-interlaced encoding, forced color buckets and no palette, a frame delay of 50ms, a lookback of 3 frames, 2 MANIAC learning iterations, a MANIAC split threshold of 38 bits, a node-count divisor of 32, and a post-pruning minimal size threshold of 70 subpixels.

AUTHORS

flif was written by Jon Sneyers and Pieter Wuille, with contributions from many others.
The latest source code is available at http://github.com/FLIF-hub/FLIF/

This manual page was written by Jon Sneyers.

SEE ALSO

viewflif(1), **convert**(1), **png**(5), **pnm**(5), **pgm**(5), **pam**(5), **dcraw**(1)

Please refer to http://flif.info/ and http://github.com/FLIF-hub/ for additional information.

COPYRIGHT

LICENSE