NAME

cwebp - compress an image file to a WebP file

SYNOPSIS

cwebp [*options*] *input_file -o output_file.webp*

DESCRIPTION

This manual page documents the **cwebp** command.

cwebp compresses an image using the WebP format. Input format can be either PNG, JPEG, TIFF, WebP or raw Y’CbCr samples. Note: Animated PNG and WebP files are not supported.

OPTIONS

The basic options are:

-o *string*

Specify the name of the output WebP file. If omitted, **cwebp** will perform compression but only report statistics. Using "-" as output name will direct output to ’stdout’.

-- *string*

Explicitly specify the input file. This option is useful if the input file starts with a ’-’ for instance. This option must appear **last**. Any other options afterward will be ignored.

-h, -help

A short usage summary.

-H, -longhelp

A summary of all the possible options.

-version

Print the version number (as major.minor.revision) and exit.

-lossless

Encode the image without any loss. For images with fully transparent area, the invisible pixel values (R/G/B or Y/U/V) will be preserved only if the -exact option is used.

-near_lossless *int*

Specify the level of near-lossless image preprocessing. This option adjusts pixel values to help compressibility, but has minimal impact on the visual quality. It triggers lossless compression mode automatically. The range is 0 (maximum preprocessing) to 100 (no preprocessing, the

default). The typical value is around 60. Note that lossy with **-q 100** can at times yield better results.

-q *float*

Specify the compression factor for RGB channels between 0 and 100. The default is 75.
In case of lossy compression (default), a small factor produces a smaller file with lower quality. Best quality is achieved by using a value of 100.
In case of lossless compression (specified by the **-lossless** option), a small factor enables faster compression speed, but produces a larger file. Maximum compression is achieved by using a value of 100.

-z *int*

Switch on **lossless** compression mode with the specified level between 0 and 9, with level 0 being the fastest, 9 being the slowest. Fast mode produces larger file size than slower ones. A good default is **-z 6**. This option is actually a shortcut for some predefined settings for quality and method. If options **-q** or **-m** are subsequently used, they will invalidate the effect of this option.

-alpha_q *int*

Specify the compression factor for alpha compression between 0 and 100. Lossless compression of alpha is achieved using a value of 100, while the lower values result in a lossy compression. The default is 100.

-preset *string*

Specify a set of pre-defined parameters to suit a particular type of source material. Possible values are: **default**, **photo**, **picture**, **drawing**, **icon**, **text**. Since **-preset** overwrites the other parameters' values (except the **-q** one), this option should preferably appear first in the order of the arguments.

-m *int*

Specify the compression method to use. This parameter controls the trade off between encoding speed and the compressed file size and quality. Possible values range from 0 to 6. Default value is 4. When higher values are used, the encoder will spend more time inspecting additional encoding possibilities and decide on the quality gain. Lower value can result in faster processing time at the expense of larger file size and lower compression quality.

-crop *x_position y_position width height*

Crop the source to a rectangle with top-left corner at coordinates (**x_position**, **y_position**) and size **width** x **height**. This cropping area must be fully contained within the source rectangle. Note: the cropping is applied *before* any scaling.

-resize *width height*

Resize the source to a rectangle with size **width** x **height**. If either (but not both) of the **width** or **height** parameters is 0, the value will be calculated preserving the aspect-ratio. Note: scaling is applied *after* cropping.

-mt Use multi-threading for encoding, if possible.

-low_memory

Reduce memory usage of lossy encoding by saving four times the compressed size (typically). This will make the encoding slower and the output slightly different in size and distortion. This flag is only effective for methods 3 and up, and is off by default. Note that leaving this flag off will have some side effects on the bitstream: it forces certain bitstream features like number of partitions (forced to 1). Note that a more detailed report of bitstream size is printed by **cwebp** when using this option.

LOSSY OPTIONS

These options are only effective when doing lossy encoding (the default, with or without alpha).

-size *int*

Specify a target size (in bytes) to try and reach for the compressed output. The compressor will make several passes of partial encoding in order to get as close as possible to this target. If both **-size** and **-psnr** are used, **-size** value will prevail.

-psnr *float*

Specify a target PSNR (in dB) to try and reach for the compressed output. The compressor will make several passes of partial encoding in order to get as close as possible to this target. If both **-size** and **-psnr** are used, **-size** value will prevail.

-pass *int*

Set a maximum number of passes to use during the dichotomy used by options **-size** or **-psnr**. Maximum value is 10, default is 1. If options **-size** or **-psnr** were used, but **-pass** wasn't specified, a default value of '6' passes will be used.

-qrange *int int*

Specifies the permissible interval for the quality factor. This is particularly useful when using multi-pass (**-size** or **-psnr** options). Default is 0 100. If the quality factor is outside this range, it will be clamped. If the minimum value must be less or equal to the maximum one.

-af Turns auto-filter on. This algorithm will spend additional time optimizing the filtering strength to

reach a well-balanced quality.

-jpeg_like
Change the internal parameter mapping to better match the expected size of JPEG compression. This flag will generally produce an output file of similar size to its JPEG equivalent (for the same **-q** setting), but with less visual distortion.

Advanced options:

-f *int*
Specify the strength of the deblocking filter, between 0 (no filtering) and 100 (maximum filtering). A value of 0 will turn off any filtering. Higher value will increase the strength of the filtering process applied after decoding the picture. The higher the value the smoother the picture will appear. Typical values are usually in the range of 20 to 50.

-sharpness *int*
Specify the sharpness of the filtering (if used). Range is 0 (sharpest) to 7 (least sharp). Default is 0.

-strong
Use strong filtering (if filtering is being used thanks to the **-f** option). Strong filtering is on by default.

-nostrong
Disable strong filtering (if filtering is being used thanks to the **-f** option) and use simple filtering instead.

-sharp_yuv
Use more accurate and sharper RGB->YUV conversion if needed. Note that this process is slower than the default 'fast' RGB->YUV conversion.

-sns *int*
Specify the amplitude of the spatial noise shaping. Spatial noise shaping (or **sns** for short) refers to a general collection of built-in algorithms used to decide which area of the picture should use relatively less bits, and where else to better transfer these bits. The possible range goes from 0 (algorithm is off) to 100 (the maximal effect). The default value is 50.

-segments *int*

Change the number of partitions to use during the segmentation of the sns algorithm. Segments should be in range 1 to 4. Default value is 4. This option has no effect for methods 3 and up, unless **-low_memory** is used.

-partition_limit *int*

Degrade quality by limiting the number of bits used by some macroblocks. Range is 0 (no degradation, the default) to 100 (full degradation). Useful values are usually around 30-70 for moderately large images. In the VP8 format, the so-called control partition has a limit of 512k and is used to store the following information: whether the macroblock is skipped, which segment it belongs to, whether it is coded as intra 4x4 or intra 16x16 mode, and finally the prediction modes to use for each of the sub-blocks. For a very large image, 512k only leaves room to few bits per 16x16 macroblock. The absolute minimum is 4 bits per macroblock. Skip, segment, and mode information can use up almost all these 4 bits (although the case is unlikely), which is problematic for very large images. The partition_limit factor controls how frequently the most bit-costly mode (intra 4x4) will be used. This is useful in case the 512k limit is reached and the following message is displayed: *Error code: 6 (PARTITION0_OVERFLOW: Partition #0 is too big to fit 512k)*. If using **-partition_limit** is not enough to meet the 512k constraint, one should use less segments in order to save more header bits per macroblock. See the **-segments** option.

LOGGING OPTIONS

These options control the level of output:

-v Print extra information (encoding time in particular).

-print_psnr

Compute and report average PSNR (Peak-Signal-To-Noise ratio).

-print_ssim

Compute and report average SSIM (structural similarity metric, see https://en.wikipedia.org/wiki/SSIM for additional details).

-print_lsim

Compute and report local similarity metric (sum of lowest error amongst the collocated pixel neighbors).

-progress

Report encoding progress in percent.

-quiet

Do not print anything.

-short
Only print brief information (output file size and PSNR) for testing purposes.

-map *int*
Output additional ASCII-map of encoding information. Possible map values range from 1 to 6. This is only meant to help debugging.

ADDITIONAL OPTIONS

More advanced options are:

-s *width height*
Specify that the input file actually consists of raw Y'CbCr samples following the ITU-R BT.601 recommendation, in 4:2:0 linear format. The luma plane has size **width** x **height**.

-pre *int*
Specify some preprocessing steps. Using a value of '2' will trigger quality-dependent pseudo-random dithering during RGBA->YUVA conversion (lossy compression only).

-alpha_filter *string*
Specify the predictive filtering method for the alpha plane. One of 'none', 'fast' or 'best', in increasing complexity and slowness order. Default is 'fast'. Internally, alpha filtering is performed using four possible predictions (none, horizontal, vertical, gradient). The 'best' mode will try each mode in turn and pick the one which gives the smaller size. The 'fast' mode will just try to form an a priori guess without testing all modes.

-alpha_method *int*
Specify the algorithm used for alpha compression: 0 or 1. Algorithm 0 denotes no compression, 1 uses WebP lossless format for compression. The default is 1.

-exact
Preserve RGB values in transparent area. The default is off, to help compressibility.

-blend_alpha *int*
This option blends the alpha channel (if present) with the source using the background color specified in hexadecimal as 0xrrggbb. The alpha channel is afterward reset to the opaque value 255.

-noalpha
Using this option will discard the alpha channel.

-hint *string*
Specify the hint about input image type. Possible values are: **photo**, **picture** or **graph**.

-metadata *string*
A comma separated list of metadata to copy from the input to the output if present. Valid values: **all**, **none**, **exif**, **icc**, **xmp**. The default is **none**.

Note: each input format may not support all combinations.

-noasm
Disable all assembly optimizations.

BUGS

Please report all bugs to the issue tracker: https://bugs.chromium.org/p/webp
Patches welcome! See this page to get started:
https://www.webmproject.org/code/contribute/submitting-patches/

EXAMPLES

cwebp -q 50 -lossless picture.png -o picture_lossless.webp
cwebp -q 70 picture_with_alpha.png -o picture_with_alpha.webp
cwebp -sns 70 -f 50 -size 60000 picture.png -o picture.webp
cwebp -o picture.webp -- ---picture.png

AUTHORS

cwebp is a part of libwebp and was written by the WebP team.
The latest source tree is available at https://chromium.googlesource.com/webm/libwebp

This manual page was written by Pascal Massimino <pascal.massimino@gmail.com>, for the Debian project (and may be used by others).

SEE ALSO

dwebp(1), **gif2webp**(1)
Please refer to https://developers.google.com/speed/webp/ for additional information.