NAME

encode.h − API for Brotli compression.

SYNOPSIS

Macros

#define **BROTLI_DEFAULT_MODE BROTLI_MODE_GENERIC**
Default value for **BROTLI_PARAM_MODE** *parameter.*

#define **BROTLI_DEFAULT_QUALITY** 11
Default value for **BROTLI_PARAM_QUALITY** *parameter.*

#define **BROTLI_DEFAULT_WINDOW** 22
Default value for **BROTLI_PARAM_LGWIN** *parameter.*

#define **BROTLI_LARGE_MAX_WINDOW_BITS** 30
Maximal value for **BROTLI_PARAM_LGWIN** *parameter in 'Large Window Brotli' (32-bit).*

#define **BROTLI_MAX_INPUT_BLOCK_BITS** 24
Maximal value for **BROTLI_PARAM_LGBLOCK** *parameter.*

#define **BROTLI_MAX_QUALITY** 11
Maximal value for **BROTLI_PARAM_QUALITY** *parameter.*

#define **BROTLI_MAX_WINDOW_BITS** 24
Maximal value for **BROTLI_PARAM_LGWIN** *parameter.*

#define **BROTLI_MIN_INPUT_BLOCK_BITS** 16
Minimal value for **BROTLI_PARAM_LGBLOCK** *parameter.*

#define **BROTLI_MIN_QUALITY** 0
Minimal value for **BROTLI_PARAM_QUALITY** *parameter.*

#define **BROTLI_MIN_WINDOW_BITS** 10
Minimal value for **BROTLI_PARAM_LGWIN** *parameter.*

Typedefs

typedef enum **BrotliEncoderMode BrotliEncoderMode**
Options for **BROTLI_PARAM_MODE** *parameter.*

typedef enum **BrotliEncoderOperation BrotliEncoderOperation**
Operations that can be performed by streaming encoder.

typedef enum **BrotliEncoderParameter BrotliEncoderParameter**
Options to be used with **BrotliEncoderSetParameter**.

typedef struct BrotliEncoderStateStruct **BrotliEncoderState**
Opaque structure that holds encoder state.

Enumerations

Functions

BROTLI_BOOL BrotliEncoderAttachPreparedDictionary (**BrotliEncoderState** *state, const BrotliEncoderPreparedDictionary *dictionary)
Attaches a prepared dictionary of any type to the encoder.

BROTLI_BOOL BrotliEncoderCompress (int quality, int lgwin, **BrotliEncoderMode** mode, size_t input_size, const uint8_t input_buffer[input_size], size_t *encoded_size, uint8_t encoded_buffer[*encoded_size])
Performs one-shot memory-to-memory compression.

BROTLI_BOOL BrotliEncoderCompressStream (**BrotliEncoderState** *state, **BrotliEncoderOperation** op, size_t *available_in, const uint8_t **next_in, size_t *available_out, uint8_t **next_out, size_t *total_out)
Compresses input stream to output stream.

BrotliEncoderState * **BrotliEncoderCreateInstance** (**brotli_alloc_func** alloc_func, **brotli_free_func** free_func, void *opaque)
Creates an instance of **BrotliEncoderState** *and initializes it.*

void **BrotliEncoderDestroyInstance** (**BrotliEncoderState** *state)
Deinitializes and frees **BrotliEncoderState** *instance.*

BROTLI_BOOL BrotliEncoderHasMoreOutput (**BrotliEncoderState** *state)

Checks if encoder has more output.

BROTLI_BOOL BrotliEncoderIsFinished (**BrotliEncoderState** *state)
Checks if encoder instance reached the final state.

size_t **BrotliEncoderMaxCompressedSize** (size_t input_size)
Calculates the output size bound for the given `input_size`.

BrotliEncoderPreparedDictionary * **BrotliEncoderPrepareDictionary** (BrotliSharedDictionaryType type, size_t data_size, const uint8_t data[data_size], int quality, **brotli_alloc_func** alloc_func, **brotli_free_func** free_func, void *opaque)
Prepares a shared dictionary from the given file format for the encoder.

BROTLI_BOOL BrotliEncoderSetParameter (**BrotliEncoderState** *state, **BrotliEncoderParameter** param, uint32_t value)
Sets the specified parameter to the given encoder instance.

const uint8_t * **BrotliEncoderTakeOutput** (**BrotliEncoderState** *state, size_t *size)
Acquires pointer to internal output buffer.

uint32_t **BrotliEncoderVersion** (void)
Gets an encoder library version.

Detailed Description

API for Brotli compression.

Macro Definition Documentation

#define BROTLI_DEFAULT_MODE BROTLI_MODE_GENERIC
Default value for **BROTLI_PARAM_MODE** parameter.

#define BROTLI_DEFAULT_QUALITY 11
Default value for **BROTLI_PARAM_QUALITY** parameter.

#define BROTLI_DEFAULT_WINDOW 22
Default value for **BROTLI_PARAM_LGWIN** parameter.

#define BROTLI_MAX_INPUT_BLOCK_BITS 24
Maximal value for **BROTLI_PARAM_LGBLOCK** parameter.

#define BROTLI_MAX_QUALITY 11
Maximal value for **BROTLI_PARAM_QUALITY** parameter.

#define BROTLI_MAX_WINDOW_BITS 24
Maximal value for **BROTLI_PARAM_LGWIN** parameter.

Note:
equal to `BROTLI_MAX_DISTANCE_BITS` constant.

#define BROTLI_MIN_INPUT_BLOCK_BITS 16
Minimal value for **BROTLI_PARAM_LGBLOCK** parameter.

#define BROTLI_MIN_QUALITY 0
Minimal value for **BROTLI_PARAM_QUALITY** parameter.

#define BROTLI_MIN_WINDOW_BITS 10
Minimal value for **BROTLI_PARAM_LGWIN** parameter.

Typedef Documentation

typedef enum BrotliEncoderMode BrotliEncoderMode
Options for **BROTLI_PARAM_MODE** parameter.

typedef enum BrotliEncoderOperation BrotliEncoderOperation
Operations that can be performed by streaming encoder.

typedef enum BrotliEncoderParameter BrotliEncoderParameter
Options to be used with **BrotliEncoderSetParameter**.

typedef struct BrotliEncoderStateStruct BrotliEncoderState

Opaque structure that holds encoder state. Allocated and initialized with **BrotliEncoderCreateInstance**. Cleaned up and deallocated with **BrotliEncoderDestroyInstance**.

Enumeration Type Documentation

enum BrotliEncoderMode

Options for **BROTLI_PARAM_MODE** parameter.

Enumerator

BROTLI_MODE_GENERIC

Default compression mode. In this mode compressor does not know anything in advance about the properties of the input.

BROTLI_MODE_TEXT

Compression mode for UTF-8 formatted text input.

BROTLI_MODE_FONT

Compression mode used in WOFF 2.0.

enum BrotliEncoderOperation

Operations that can be performed by streaming encoder.

Enumerator

BROTLI_OPERATION_PROCESS

Process input. Encoder may postpone producing output, until it has processed enough input.

BROTLI_OPERATION_FLUSH

Produce output for all processed input. Actual flush is performed when input stream is depleted and there is enough space in output stream. This means that client should repeat **BROTLI_OPERATION_FLUSH** operation until `available_in` becomes `0`, and **BrotliEncoderHasMoreOutput** returns **BROTLI_FALSE**. If output is acquired via **BrotliEncoderTakeOutput**, then operation should be repeated after output buffer is drained.

Warning:

Until flush is complete, client **SHOULD NOT** swap, reduce or extend input stream.

When flush is complete, output data will be sufficient for decoder to reproduce all the given input.

BROTLI_OPERATION_FINISH

Finalize the stream. Actual finalization is performed when input stream is depleted and there is enough space in output stream. This means that client should repeat **BROTLI_OPERATION_FINISH** operation until `available_in` becomes `0`, and **BrotliEncoderHasMoreOutput** returns **BROTLI_FALSE**. If output is acquired via **BrotliEncoderTakeOutput**, then operation should be repeated after output buffer is drained.

Warning:

Until finalization is complete, client **SHOULD NOT** swap, reduce or extend input stream.

Helper function **BrotliEncoderIsFinished** checks if stream is finalized and output fully dumped.

Adding more input data to finalized stream is impossible.

BROTLI_OPERATION_EMIT_METADATA

Emit metadata block to stream. Metadata is opaque to Brotli: neither encoder, nor decoder processes this data or relies on it. It may be used to pass some extra information from encoder client to decoder client without interfering with main data stream.

Note:

Encoder may emit empty metadata blocks internally, to pad encoded stream to byte boundary.

Warning:

Until emitting metadata is complete client **SHOULD NOT** swap, reduce or extend input stream.

The whole content of input buffer is considered to be the content of metadata block. Do **NOT** *append*

metadata to input stream, before it is depleted with other operations.

Stream is soft-flushed before metadata block is emitted. Metadata block **MUST** be no longer than than 16MiB.

enum BrotliEncoderParameter

Options to be used with **BrotliEncoderSetParameter**.

Enumerator

BROTLI_PARAM_MODE
: Tune encoder for specific input. **BrotliEncoderMode** enumerates all available values.

BROTLI_PARAM_QUALITY
: The main compression speed-density lever. The higher the quality, the slower the compression. Range is from **BROTLI_MIN_QUALITY** to **BROTLI_MAX_QUALITY**.

BROTLI_PARAM_LGWIN
: Recommended sliding LZ77 window size. Encoder may reduce this value, e.g. if input is much smaller than window size.

Window size is `(1 << value) - 16`.

Range is from **BROTLI_MIN_WINDOW_BITS** to **BROTLI_MAX_WINDOW_BITS**.

BROTLI_PARAM_LGBLOCK
: Recommended input block size. Encoder may reduce this value, e.g. if input is much smaller than input block size.

Range is from **BROTLI_MIN_INPUT_BLOCK_BITS** to **BROTLI_MAX_INPUT_BLOCK_BITS**.

Note:
Bigger input block size allows better compression, but consumes more memory.
The rough formula of memory used for temporary input storage is `3 << lgBlock`.

BROTLI_PARAM_DISABLE_LITERAL_CONTEXT_MODELING
: Flag that affects usage of 'literal context modeling' format feature. This flag is a 'decoding-speed vs compression ratio' trade-off.

BROTLI_PARAM_SIZE_HINT
: Estimated total input size for all **BrotliEncoderCompressStream** calls. The default value is 0, which means that the total input size is unknown.

BROTLI_PARAM_LARGE_WINDOW
: Flag that determines if 'Large Window Brotli' is used.

BROTLI_PARAM_NPOSTFIX
: Recommended number of postfix bits (NPOSTFIX). Encoder may change this value.

Range is from 0 to **BROTLI_MAX_NPOSTFIX**.

BROTLI_PARAM_NDIRECT
: Recommended number of direct distance codes (NDIRECT). Encoder may change this value.

Range is from 0 to (15 << NPOSTFIX) in steps of (1 << NPOSTFIX).

BROTLI_PARAM_STREAM_OFFSET
: Number of bytes of input stream already processed by a different instance.

Note:
It is important to configure all the encoder instances with same parameters (except this one) in order to allow all the encoded parts obey the same restrictions implied by header.

If offset is not 0, then stream header is omitted. In any case output start is byte aligned, so for proper streams stitching 'predecessor' stream must be flushed.

Range is not artificially limited, but all the values greater or equal to maximal window size have the same effect. Values greater than 2**30 are not allowed.

Function Documentation

BROTLI_BOOL BrotliEncoderAttachPreparedDictionary (BrotliEncoderState * state, const BrotliEncoderPreparedDictionary * dictionary)

Attaches a prepared dictionary of any type to the encoder. Can be used multiple times to attach multiple dictionaries. The dictionary type was determined by BrotliEncoderPrepareDictionary. Multiple raw prefix dictionaries and/or max 1 serialized dictionary with custom words can be attached.

Returns:

BROTLI_FALSE in case of error

BROTLI_TRUE otherwise

BROTLI_BOOL BrotliEncoderCompress (int quality, int lgwin, BrotliEncoderMode mode, size_t input_size, const uint8_t input_buffer[input_size], size_t * encoded_size, uint8_t encoded_buffer[*encoded_size])

Performs one-shot memory-to-memory compression. Compresses the data in `input_buffer` into `encoded_buffer`, and sets `*encoded_size` to the compressed length.

Note:

If **BrotliEncoderMaxCompressedSize**(`input_size`) returns non-zero value, then output is guaranteed to be no longer than that.

If `lgwin` is greater than **BROTLI_MAX_WINDOW_BITS** then resulting stream might be incompatible with RFC 7932; to decode such streams, decoder should be configured with **BROTLI_DECODER_PARAM_LARGE_WINDOW** = `1`

Parameters:

quality quality parameter value, e.g. **BROTLI_DEFAULT_QUALITY**
lgwin lgwin parameter value, e.g. **BROTLI_DEFAULT_WINDOW**
mode mode parameter value, e.g. **BROTLI_DEFAULT_MODE**
input_size size of `input_buffer`
input_buffer input data buffer with at least `input_size` addressable bytes
encoded_size **in:** size of `encoded_buffer`;
out: length of compressed data written to `encoded_buffer`, or `0` if compression fails
encoded_buffer compressed data destination buffer

Returns:

BROTLI_FALSE in case of compression error

BROTLI_FALSE if output buffer is too small

BROTLI_TRUE otherwise

BROTLI_BOOL BrotliEncoderCompressStream (BrotliEncoderState * state, BrotliEncoderOperation op, size_t * available_in, const uint8_t ** next_in, size_t * available_out, uint8_t ** next_out, size_t * total_out)

Compresses input stream to output stream. The values `*available_in` and `*available_out` must specify the number of bytes addressable at `*next_in` and `*next_out` respectively. When `*available_out` is `0`, `next_out` is allowed to be `NULL`.

After each call, `*available_in` will be decremented by the amount of input bytes consumed, and the `*next_in` pointer will be incremented by that amount. Similarly, `*available_out` will be decremented by the amount of output bytes written, and the `*next_out` pointer will be incremented by that amount.

`total_out`, if it is not a null-pointer, will be set to the number of bytes compressed since the last `state` initialization.

Internally workflow consists of 3 tasks:

1. (optionally) copy input data to internal buffer

2. actually compress data and (optionally) store it to internal buffer
3. (optionally) copy compressed bytes from internal buffer to output stream

Whenever all 3 tasks can't move forward anymore, or error occurs, this method returns the control flow to caller.

`op` is used to perform flush, finish the stream, or inject metadata block. See **BrotliEncoderOperation** for more information.

Flushing the stream means forcing encoding of all input passed to encoder and completing the current output block, so it could be fully decoded by stream decoder. To perform flush set `op` to **BROTLI_OPERATION_FLUSH**. Under some circumstances (e.g. lack of output stream capacity) this operation would require several calls to **BrotliEncoderCompressStream**. The method must be called again until both input stream is depleted and encoder has no more output (see **BrotliEncoderHasMoreOutput**) after the method is called.

Finishing the stream means encoding of all input passed to encoder and adding specific 'final' marks, so stream decoder could determine that stream is complete. To perform finish set `op` to **BROTLI_OPERATION_FINISH**. Under some circumstances (e.g. lack of output stream capacity) this operation would require several calls to **BrotliEncoderCompressStream**. The method must be called again until both input stream is depleted and encoder has no more output (see **BrotliEncoderHasMoreOutput**) after the method is called.

Warning:
When flushing and finishing, `op` should not change until operation is complete; input stream should not be swapped, reduced or extended as well.

Parameters:
state encoder instance
op requested operation
available_in **in:** amount of available input;
out: amount of unused input
next_in pointer to the next input byte
available_out **in:** length of output buffer;
out: remaining size of output buffer
next_out compressed output buffer cursor; can be `NULL` if `available_out` is `0`
total_out number of bytes produced so far; can be `NULL`

Returns:
BROTLI_FALSE if there was an error

BROTLI_TRUE otherwise

BrotliEncoderState* BrotliEncoderCreateInstance (brotli_alloc_func alloc_func, brotli_free_func free_func, void * opaque)
Creates an instance of **BrotliEncoderState** and initializes it. `alloc_func` and `free_func` **MUST** be both zero or both non-zero. In the case they are both zero, default memory allocators are used. `opaque` is passed to `alloc_func` and `free_func` when they are called. `free_func` has to return without doing anything when asked to free a NULL pointer.

Parameters:
alloc_func custom memory allocation function
free_func custom memory free function
opaque custom memory manager handle

Returns:
`0` if instance can not be allocated or initialized

pointer to initialized **BrotliEncoderState** otherwise

void BrotliEncoderDestroyInstance (BrotliEncoderState * state)
Deinitializes and frees **BrotliEncoderState** instance.

Parameters:
state decoder instance to be cleaned up and deallocated

BROTLI_BOOL BrotliEncoderHasMoreOutput (BrotliEncoderState * state)
Checks if encoder has more output.

Parameters:
state encoder instance

Returns:
BROTLI_TRUE, if encoder has some unconsumed output

BROTLI_FALSE otherwise

BROTLI_BOOL BrotliEncoderIsFinished (BrotliEncoderState * state)
Checks if encoder instance reached the final state.

Parameters:
state encoder instance

Returns:
BROTLI_TRUE if encoder is in a state where it reached the end of the input and produced all of the output

BROTLI_FALSE otherwise

size_t BrotliEncoderMaxCompressedSize (size_t input_size)
Calculates the output size bound for the given `input_size`.

Warning:
Result is only valid if quality is at least `2` and, in case **BrotliEncoderCompressStream** was used, no flushes (**BROTLI_OPERATION_FLUSH**) were performed.

Parameters:
input_size size of projected input

Returns:
`0` if result does not fit `size_t`

BrotliEncoderPreparedDictionary* BrotliEncoderPrepareDictionary (BrotliSharedDictionaryType type, size_t data_size, const uint8_t data[data_size], int quality, brotli_alloc_func alloc_func, brotli_free_func free_func, void * opaque)
Prepares a shared dictionary from the given file format for the encoder. `alloc_func` and `free_func` **MUST** be both zero or both non-zero. In the case they are both zero, default memory allocators are used. `opaque` is passed to `alloc_func` and `free_func` when they are called. `free_func` has to return without doing anything when asked to free a NULL pointer.

Parameters:
type type of dictionary stored in data
data_size size of `data` buffer
data pointer to the dictionary data
quality the maximum Brotli quality to prepare the dictionary for, use BROTLI_MAX_QUALITY by default
alloc_func custom memory allocation function
free_func custom memory free function
opaque custom memory manager handle

BROTLI_BOOL BrotliEncoderSetParameter (BrotliEncoderState * state, BrotliEncoderParameter param, uint32_t value)
Sets the specified parameter to the given encoder instance.

Parameters:

state encoder instance
param parameter to set
value new parameter value

Returns:

BROTLI_FALSE if parameter is unrecognized, or value is invalid

BROTLI_FALSE if value of parameter can not be changed at current encoder state (e.g. when encoding is started, window size might be already encoded and therefore it is impossible to change it)

BROTLI_TRUE if value is accepted

Warning:

invalid values might be accepted in case they would not break encoding process.

const uint8_t* BrotliEncoderTakeOutput (BrotliEncoderState * state, size_t * size)

Acquires pointer to internal output buffer. This method is used to make language bindings easier and more efficient:

1. push data to **BrotliEncoderCompressStream**, until **BrotliEncoderHasMoreOutput** returns BROTLI_TRUE
2. use **BrotliEncoderTakeOutput** to peek bytes and copy to language-specific entity

Also this could be useful if there is an output stream that is able to consume all the provided data (e.g. when data is saved to file system).

Attention:

After every call to **BrotliEncoderTakeOutput** `*size` bytes of output are considered consumed for all consecutive calls to the instance methods; returned pointer becomes invalidated as well.

Note:

Encoder output is not guaranteed to be contiguous. This means that after the size-unrestricted call to **BrotliEncoderTakeOutput**, immediate next call to **BrotliEncoderTakeOutput** may return more data.

Parameters:

state encoder instance
size **in:** number of bytes caller is ready to take, `0` if any amount could be handled;
out: amount of data pointed by returned pointer and considered consumed;
out value is never greater than in value, unless it is `0`

Returns:

pointer to output data

uint32_t BrotliEncoderVersion (void)

Gets an encoder library version. Look at BROTLI_MAKE_HEX_VERSION for more information.

Author

Generated automatically by Doxygen for Brotli from the source code.