NAME

decode.h − API for Brotli decompression.

SYNOPSIS

Macros

#define **BROTLI_DECODER_ERROR_CODES_LIST**(BROTLI_ERROR_CODE, SEPARATOR)
Template that evaluates items of **BrotliDecoderErrorCode**.

#define **BROTLI_LAST_ERROR_CODE** BROTLI_DECODER_ERROR_UNREACHABLE
The value of the last error code, negative integer.

Typedefs

typedef void(* **brotli_decoder_metadata_chunk_func**) (void *opaque, const uint8_t *data, size_t size)
Callback to fire on metadata block chunk becomes available.

typedef void(* **brotli_decoder_metadata_start_func**) (void *opaque, size_t size)
Callback to fire on metadata block start.

typedef enum **BrotliDecoderParameter BrotliDecoderParameter**
Options to be used with **BrotliDecoderSetParameter**.

typedef struct BrotliDecoderStateStruct **BrotliDecoderState**
Opaque structure that holds decoder state.

Enumerations

Functions

BROTLI_BOOL BrotliDecoderAttachDictionary (**BrotliDecoderState** *state, BrotliSharedDictionaryType type, size_t data_size, const uint8_t data[data_size])
Adds LZ77 prefix dictionary, adds or replaces built-in static dictionary and transforms.

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

BrotliDecoderResult BrotliDecoderDecompress (size_t encoded_size, const uint8_t encoded_buffer[encoded_size], size_t *decoded_size, uint8_t decoded_buffer[*decoded_size])
Performs one-shot memory-to-memory decompression.

BrotliDecoderResult BrotliDecoderDecompressStream (**BrotliDecoderState** *state, size_t *available_in, const uint8_t **next_in, size_t *available_out, uint8_t **next_out, size_t *total_out)
Decompresses the input stream to the output stream.

void **BrotliDecoderDestroyInstance** (**BrotliDecoderState** *state)
Deinitializes and frees **BrotliDecoderState** *instance.*

const char * **BrotliDecoderErrorString** (**BrotliDecoderErrorCode** c)
Converts error code to a c-string.

BrotliDecoderErrorCode BrotliDecoderGetErrorCode (const **BrotliDecoderState** *state)
Acquires a detailed error code.

BROTLI_BOOL BrotliDecoderHasMoreOutput (const **BrotliDecoderState** *state)
Checks if decoder has more output.

BROTLI_BOOL BrotliDecoderIsFinished (const **BrotliDecoderState** *state)
Checks if decoder instance reached the final state.

BROTLI_BOOL BrotliDecoderIsUsed (const **BrotliDecoderState** *state)
Checks if instance has already consumed input.

void **BrotliDecoderSetMetadataCallbacks** (**BrotliDecoderState** *state, **brotli_decoder_metadata_start_func** start_func, **brotli_decoder_metadata_chunk_func** chunk_func, void *opaque)
Sets callback for receiving metadata blocks.

BROTLI_BOOL BrotliDecoderSetParameter (**BrotliDecoderState** *state, **BrotliDecoderParameter** param, uint32_t value)
Sets the specified parameter to the given decoder instance.

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

uint32_t **BrotliDecoderVersion** (void)
Gets a decoder library version.

Detailed Description

API for Brotli decompression.

Macro Definition Documentation

#define BROTLI_DECODER_ERROR_CODES_LIST(BROTLI_ERROR_CODE, SEPARATOR)

Template that evaluates items of **BrotliDecoderErrorCode**. Example:

```
// Log Brotli error code.
switch (brotliDecoderErrorCode) {
#define CASE_(PREFIX, NAME, CODE)   case BROTLI_DECODER ## PREFIX ## NAME:     LOG(INFO) << "error code
#define NEWLINE_
BROTLI_DECODER_ERROR_CODES_LIST(CASE_, NEWLINE_)
#undef CASE_
#undef NEWLINE_
  default: LOG(FATAL) << "unknown brotli error code";
}
```

#define BROTLI_LAST_ERROR_CODE BROTLI_DECODER_ERROR_UNREACHABLE

The value of the last error code, negative integer. All other error code values are in the range from **BROTLI_LAST_ERROR_CODE** to `-1`. There are also 4 other possible non-error codes `0` .. `3` in **BrotliDecoderErrorCode** enumeration.

Typedef Documentation

typedef void(* brotli_decoder_metadata_chunk_func) (void *opaque, const uint8_t *data, size_t size)

Callback to fire on metadata block chunk becomes available. This function can be invoked multiple times per metadata block; block should be considered finished when sum of `size` matches the announced metadata block size. Chunks contents pointed by `data` are transient and shouln not be accessed after leaving the callback.

Parameters:

opaque callback handle
data pointer to metadata contents
size size of metadata block chunk, at least `1`

typedef void(* brotli_decoder_metadata_start_func) (void *opaque, size_t size)

Callback to fire on metadata block start. After this callback is fired, if `size` is not `0`, it is followed by **brotli_decoder_metadata_chunk_func** as more metadata block contents become accessible.

Parameters:

opaque callback handle
size size of metadata block

typedef enum BrotliDecoderParameter BrotliDecoderParameter

Options to be used with **BrotliDecoderSetParameter**.

typedef struct BrotliDecoderStateStruct BrotliDecoderState

Opaque structure that holds decoder state. Allocated and initialized with **BrotliDecoderCreateInstance**. Cleaned up and deallocated with **BrotliDecoderDestroyInstance**.

Enumeration Type Documentation

enum BrotliDecoderErrorCode

Error code for detailed logging / production debugging. See **BrotliDecoderGetErrorCode** and **BROTLI_LAST_ERROR_CODE**.

enum BrotliDecoderParameter

Options to be used with **BrotliDecoderSetParameter**.

Enumerator

BROTLI_DECODER_PARAM_DISABLE_RING_BUFFER_REALLOCATION
: Disable 'canny' ring buffer allocation strategy. Ring buffer is allocated according to window size, despite the real size of the content.

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

enum BrotliDecoderResult

Result type for **BrotliDecoderDecompress** and **BrotliDecoderDecompressStream** functions.

Enumerator

BROTLI_DECODER_RESULT_ERROR
: Decoding error, e.g. corrupted input or memory allocation problem.

BROTLI_DECODER_RESULT_SUCCESS
: Decoding successfully completed.

BROTLI_DECODER_RESULT_NEEDS_MORE_INPUT
: Partially done; should be called again with more input.

BROTLI_DECODER_RESULT_NEEDS_MORE_OUTPUT
: Partially done; should be called again with more output.

Function Documentation

BROTLI_BOOL BrotliDecoderAttachDictionary (BrotliDecoderState * state, BrotliSharedDictionaryType type, size_t data_size, const uint8_t data[data_size])

Adds LZ77 prefix dictionary, adds or replaces built-in static dictionary and transforms. Attached dictionary ownership is not transferred. Data provided to this method should be kept accessible until decoding is finished and decoder instance is destroyed.

Note:

Dictionaries can NOT be attached after actual decoding is started.

Parameters:

state decoder instance
type dictionary data format
data_size length of memory region pointed by `data`
data dictionary data in format corresponding to `type`

Returns:

BROTLI_FALSE if dictionary is corrupted, or dictionary count limit is reached

BROTLI_TRUE if dictionary is accepted / attached

BrotliDecoderState* BrotliDecoderCreateInstance (brotli_alloc_func alloc_func, brotli_free_func free_func, void * opaque)

Creates an instance of **BrotliDecoderState** and initializes it. The instance can be used once for decoding and should then be destroyed with **BrotliDecoderDestroyInstance**, it cannot be reused for a new decoding session.

`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 **BrotliDecoderState** otherwise

BrotliDecoderResult BrotliDecoderDecompress (size_t encoded_size, const uint8_t encoded_buffer[encoded_size], size_t * decoded_size, uint8_t decoded_buffer[*decoded_size])

Performs one-shot memory-to-memory decompression. Decompresses the data in `encoded_buffer` into `decoded_buffer`, and sets `*decoded_size` to the decompressed length.

Parameters:

encoded_size size of `encoded_buffer`
encoded_buffer compressed data buffer with at least `encoded_size` addressable bytes
decoded_size **in:** size of `decoded_buffer`;
out: length of decompressed data written to `decoded_buffer`
decoded_buffer decompressed data destination buffer

Returns:

BROTLI_DECODER_RESULT_ERROR if input is corrupted, memory allocation failed, or `decoded_buffer` is not large enough;

BROTLI_DECODER_RESULT_SUCCESS otherwise

BrotliDecoderResult BrotliDecoderDecompressStream (BrotliDecoderState * state, size_t * available_in, const uint8_t ** next_in, size_t * available_out, uint8_t ** next_out, size_t * total_out)

Decompresses the input stream to the 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 decompressed since the last `state` initialization.

Note:

Input is never overconsumed, so `next_in` and `available_in` could be passed to the next consumer after decoding is complete.

Parameters:

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

Returns:

BROTLI_DECODER_RESULT_ERROR if input is corrupted, memory allocation failed, arguments were invalid, etc.; use **BrotliDecoderGetErrorCode** to get detailed error code

BROTLI_DECODER_RESULT_NEEDS_MORE_INPUT decoding is blocked until more input data is provided

BROTLI_DECODER_RESULT_NEEDS_MORE_OUTPUT decoding is blocked until more output space is provided

BROTLI_DECODER_RESULT_SUCCESS decoding is finished, no more input might be consumed and no more output will be produced

void BrotliDecoderDestroyInstance (BrotliDecoderState * state)
Deinitializes and frees **BrotliDecoderState** instance.

Parameters:
state decoder instance to be cleaned up and deallocated

BrotliDecoderErrorCode BrotliDecoderGetErrorCode (const BrotliDecoderState * state)
Acquires a detailed error code. Should be used only after **BrotliDecoderDecompressStream** returns **BROTLI_DECODER_RESULT_ERROR**.

See also **BrotliDecoderErrorString**

Parameters:
state decoder instance

Returns:
last saved error code

BROTLI_BOOL BrotliDecoderHasMoreOutput (const BrotliDecoderState * state)
Checks if decoder has more output.

Parameters:
state decoder instance

Returns:
BROTLI_TRUE, if decoder has some unconsumed output

BROTLI_FALSE otherwise

BROTLI_BOOL BrotliDecoderIsFinished (const BrotliDecoderState * state)
Checks if decoder instance reached the final state.

Parameters:
state decoder instance

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

BROTLI_FALSE otherwise

BROTLI_BOOL BrotliDecoderIsUsed (const BrotliDecoderState * state)
Checks if instance has already consumed input. Instance that returns **BROTLI_FALSE** is considered 'fresh' and could be reused.

Parameters:
state decoder instance

Returns:
BROTLI_TRUE if decoder has already used some input bytes

BROTLI_FALSE otherwise

void BrotliDecoderSetMetadataCallbacks (BrotliDecoderState * state, brotli_decoder_metadata_start_func start_func, brotli_decoder_metadata_chunk_func chunk_func, void * opaque)
Sets callback for receiving metadata blocks.

Parameters:
state decoder instance
start_func callback on metadata block start
chunk_func callback on metadata block chunk
opaque callback handle

BROTLI_BOOL BrotliDecoderSetParameter (BrotliDecoderState * state, BrotliDecoderParameter param, uint32_t value)

Sets the specified parameter to the given decoder instance.

Parameters:

state decoder instance
param parameter to set
value new parameter value

Returns:

BROTLI_FALSE if parameter is unrecognized, or value is invalid

BROTLI_TRUE if value is accepted

const uint8_t* BrotliDecoderTakeOutput (BrotliDecoderState * 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 **BrotliDecoderDecompressStream**, until **BROTLI_DECODER_RESULT_NEEDS_MORE_OUTPUT** is reported
2. use **BrotliDecoderTakeOutput** 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 **BrotliDecoderTakeOutput** `*size` bytes of output are considered consumed for all consecutive calls to the instance methods; returned pointer becomes invalidated as well.

Note:

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

Parameters:

state decoder 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 BrotliDecoderVersion (void)

Gets a decoder library version. Look at BROTLI_MAKE_HEX_VERSION for more information.

Author

Generated automatically by Doxygen for Brotli from the source code.