Documentation Language: Swift

Compression / compression_algorithm

Structure

compression_algorithm

A structure for values that represent compression algorithms.

iOS | iPadOS | Mac Catalyst | macOS | tvOS | visionOS | watchOS

```
struct compression_algorithm
```

Overview

Choose an algorithm according to the following guidelines:

- If speed and compression ratio are important, use COMPRESSION_LZFSE.
- If you require interoperability with non-Apple devices, use COMPRESSION_ZLIB.
- If speed is critical, and you're willing to sacrifice compression ratio to achieve it, use COMPRESSION_LZ4.
- If compression ratio is critical, and you're willing to sacrifice speed to achieve it, use COMPRESSION_LZMA. Note that COMPRESSION_LZMA is an order of magnitude slower for both compression and decompression than other choices.

COMPRESSION_LZFSE is faster than COMPRESSION_ZLIB and generally achieves a better compression ratio. However, it's slower than COMPRESSION_LZ4 and doesn't compress as well as COMPRESSION_LZMA.

COMPRESSION_LZBITMAP provides a compression-ratio and performance that's between COMPRESSION_LZ4 and COMPRESSION_LZFSE. When compression ratio and performance are equally important, use COMPRESSION_LZFSE to favor compression ratio and COMPRESSION_LZBITMAP to favor performance.

Topics

Algorithm Constants

`var COMPRESSION_LZFSE: compression_algorithm`

The LZFSE compression algorithm, which is recommended for use on Apple platforms.

`var COMPRESSION_LZ4: compression_algorithm`

The LZ4 compression algorithm for fast compression.

`var COMPRESSION_LZ4_RAW: compression_algorithm`

The LZ4 compression algorithm, without frame headers.

`var COMPRESSION_LZMA: compression_algorithm`

The LZMA compression algorithm, which is recommended for high-compression ratio.

`var COMPRESSION_ZLIB: compression_algorithm`

The zlib compression algorithm, which is recommended for cross-platform compression.

`var COMPRESSION_BROTLI: compression_algorithm`

The Brotli compression algorithm, which is recommended for text compression.

`var COMPRESSION_LZBITMAP: compression_algorithm`

The LZBITMAP compression algorithm, which is designed to exploit the vector instruction set of current CPUs.

Initializers

`init(UInt32)`

Creates a new constant from the given raw value.

`init(rawValue: UInt32)`

Creates a new constant from the given raw value.

Instance Properties

`var rawValue: UInt32`

The raw value of the constant.

Relationships

Conforms To

```
BitwiseCopyable
Equatable
Hashable
RawRepresentable
Sendable
```

See Also

Multiple-step compression

```
struct compression_stream
```

A structure representing a compression stream.

```
func compression_stream_init(UnsafeMutablePointer<compression_stream>, compression_stream_operation, compression_algorithm) -> compression_status
```

Initializes a compression stream for either compression or decompression.

```
func compression_stream_process(UnsafeMutablePointer<compression_stream>, Int32) -> compression_status
```

Performs compression or decompression using an initialized compression stream structure.

```
func compression_stream_destroy(UnsafeMutablePointer<compression_stream>) -> compression_status
```

Frees any memory allocated by stream initialization function.

```
struct compression_status
```

A set of values used to represent the status of stream compression.

```
struct compression_stream_flags
```

A set of values used to represent stream compression flags.

`struct compression_stream_operation`

A set of values used to represent a stream compression operation.