Advanced Usage

Jump to bottom

Oz edited this page on Jan 21, 2023 · 10 revisions

Callbacks

Bit7z supports many callbacks for compression/extraction operations.

To set a callback, you should call the appropriate setter function (inherited by the base class BitAbstractArchiveHandler) of the compressor or extractor class being used, e.g.:

```
compressor.setRatioCallback( []( uint64_t in_size, uint64_t out_size ) {
    auto ratio = static_cast<int>( ( 100.0 * out_size ) / in_size );
    std::cout << "Current compression ratio: " << ratio << "%" << std::endl;
} );
```

Callbacks are stored as `std::function` s; hence they can be lambdas, free functions, member functions, etc.

Total Callback

Signature: `void( uint64_t total_size )`.

7-zip calls this function to inform the user about the total data size the requested compression or extraction operation will process.

Progress Callback

Signature: `bool( uint64_t processed_size )`.

This function, whose argument is the current processed size (in bytes), returns a boolean value that can be used to decide whether to continue or stop the current operation.

Together with the total size value obtained from the `TotalCallback`, the current processed size can be used to calculate the current operation progress percentage (e.g., `static_cast<int>( ( 100.0 * completed ) / total )`).

Ratio Callback

Signature: `void( uint64_t in_size, uint64_t out_size )`.

A function whose two arguments are, respectively, the current processed input and output sizes (in bytes) of the running operation.

These values can be used, for example, to calculate the compression ratio as `static_cast<int>( ( 100.0 * out_size ) / in_size )`.

File Callback

Signature: `void( std::string file_path )`.

7-zip calls this function before starting to process a file, either for compressing it to an archive or extracting it from one.

The callback argument is the path to the file currently being processed.

Password Callback

Signature: `std::string()`.

7-zip calls this function when it needs a password to open or extract an archive; the callback must return a `std::string` containing the password.

This callback can be handy for telling the user that a password is required to continue the operation so that the user can enter it (e.g., in a GUI dialog) to continue the process.

Stopping the current operation

TODO

Multi-volume archives

Creation

To create multi-volume archives, you need to set the desired size of each volume using the `setVolumeSize` method of the compressor/writer class you are using.

Once you start the compression, the archive volumes will be created automatically, e.g.:

```
BitFileCompressor compressor{ lib, BitFormat::SevenZip };
compressor.setVolumeSize( 100 * 1024 * 1024 ); // 100 MiB in bytes

/* The following compression operation will create the volumes
 * as "output.7z.001", "output.7z.002", and so on.
 * The number of volume files generated will vary according
 * to the size of the input and the volume size setting. */
compressor.compressDirectory( "input/path/", "output.7z" );
```

Extraction

To extract a multi-volume archive, extract the first volume: bit7z will take care of searching all the other volumes and will directly extract the content of the complete archive without having to reassemble it:

```
BitFileExtractor extractor{ lib, BitFormat::SevenZip };

// Extracting the content of the original input.7z to the output directory
extractor.extract( "input.7z.001", "output/" );
```

If you need to recreate the whole original archive from its volumes, use the `BitFormat::Split` format:

```
BitFileExtractor extractor{ lib, BitFormat::Split };

// Recreating the original input.7z (without extracting it)
extractor.extract( "input.7z.001", "output/" );
```

Compression Methods

Each format supports only a limited number of compression methods (BitCompressionMethod), as summarized in the following table:

	BitFormat						
	SevenZip	Zip	Tar	Wim	Xz	BZip2	GZip
Copy	✓	✓	✓	✓	✗	✗	✗
Ppmd	✓	✓	✗	✗	✗	✗	✗
Lzma	✓	✓	✗	✗	✗	✗	✗
Lzma2	✓	✗	✗	✗	✓	✗	✗
BZip2	✓	✓	✗	✗	✗	✓	✗
Deflate	✗	✓	✗	✗	✗	✗	✓
Deflate64	✗	✓	✗	✗	✗	✗	✗