NAME

IO::Compress::Deflate - Write RFC 1950 files/buffers

SYNOPSIS

```
use IO::Compress::Deflate qw(deflate $DeflateError) ;

my $status = deflate $input => $output [,OPTS]
    or die "deflate failed: $DeflateError\n";

my $z = new IO::Compress::Deflate $output [,OPTS]
    or die "deflate failed: $DeflateError\n";

$z->print($string);
$z->printf($format, $string);
$z->write($string);
$z->syswrite($string [, $length, $offset]);
$z->flush();
$z->tell();
$z->eof();
$z->seek($position, $whence);
$z->binmode();
$z->fileno();
$z->opened();
$z->autoflush();
$z->input_line_number();
$z->newStream( [OPTS] );

$z->deflateParams();

$z->close() ;

$DeflateError ;

# IO::File mode

print $z $string;
printf $z $format, $string;
tell $z
eof $z
seek $z, $position, $whence
binmode $z
fileno $z
close $z ;
```

DESCRIPTION

This module provides a Perl interface that allows writing compressed data to files or buffer as defined in RFC 1950.

For reading RFC 1950 files/buffers, see the companion module *IO::Uncompress::Inflate*.

Functional Interface

A top-level function, `deflate`, is provided to carry out "one-shot" compression between buffers and/or files. For finer control over the compression process, see the *OO Interface* section.

```
    use IO::Compress::Deflate qw(deflate $DeflateError) ;

    deflate $input_filename_or_reference => $output_filename_or_reference
[,OPTS]
        or die "deflate failed: $DeflateError\n";
```

The functional interface needs Perl5.005 or better.

deflate $input_filename_or_reference => $output_filename_or_reference [, OPTS]

`deflate` expects at least two parameters, `$input_filename_or_reference` and `$output_filename_or_reference`.

The $input_filename_or_reference parameter

The parameter, `$input_filename_or_reference`, is used to define the source of the uncompressed data.

It can take one of the following forms:

A filename

If the <$input_filename_or_reference> parameter is a simple scalar, it is assumed to be a filename. This file will be opened for reading and the input data will be read from it.

A filehandle

If the `$input_filename_or_reference` parameter is a filehandle, the input data will be read from it. The string '-' can be used as an alias for standard input.

A scalar reference

If `$input_filename_or_reference` is a scalar reference, the input data will be read from `$$input_filename_or_reference`.

An array reference

If `$input_filename_or_reference` is an array reference, each element in the array must be a filename.

The input data will be read from each file in turn.

The complete array will be walked to ensure that it only contains valid filenames before any data is compressed.

An Input FileGlob string

If `$input_filename_or_reference` is a string that is delimited by the characters "<" and ">" `deflate` will assume that it is an *input fileglob string*. The input is the list of files that match the fileglob.

See *File::GlobMapper* for more details.

If the `$input_filename_or_reference` parameter is any other type, `undef` will be returned.

The $output_filename_or_reference parameter

The parameter `$output_filename_or_reference` is used to control the destination of the compressed data. This parameter can take one of these forms.

A filename

If the `$output_filename_or_reference` parameter is a simple scalar, it is assumed to

be a filename. This file will be opened for writing and the compressed data will be written to it.

A filehandle

If the `$output_filename_or_reference` parameter is a filehandle, the compressed data will be written to it. The string '-' can be used as an alias for standard output.

A scalar reference

If `$output_filename_or_reference` is a scalar reference, the compressed data will be stored in `$$output_filename_or_reference`.

An Array Reference

If `$output_filename_or_reference` is an array reference, the compressed data will be pushed onto the array.

An Output FileGlob

If `$output_filename_or_reference` is a string that is delimited by the characters "<" and ">" `deflate` will assume that it is an *output fileglob string*. The output is the list of files that match the fileglob.

When `$output_filename_or_reference` is an fileglob string, `$input_filename_or_reference` must also be a fileglob string. Anything else is an error.

See *File::GlobMapper* for more details.

If the `$output_filename_or_reference` parameter is any other type, `undef` will be returned.

Notes

When `$input_filename_or_reference` maps to multiple files/buffers and `$output_filename_or_reference` is a single file/buffer the input files/buffers will be stored in `$output_filename_or_reference` as a concatenated series of compressed data streams.

Optional Parameters

Unless specified below, the optional parameters for `deflate`, `OPTS`, are the same as those used with the OO interface defined in the *Constructor Options* section below.

`AutoClose => 0|1`

This option applies to any input or output data streams to `deflate` that are filehandles.

If `AutoClose` is specified, and the value is true, it will result in all input and/or output filehandles being closed once `deflate` has completed.

This parameter defaults to 0.

`BinModeIn => 0|1`

When reading from a file or filehandle, set `binmode` before reading.

Defaults to 0.

`Append => 0|1`

The behaviour of this option is dependent on the type of output data stream.

* A Buffer

  If `Append` is enabled, all compressed data will be append to the end of the output buffer. Otherwise the output buffer will be cleared before any compressed data is written to it.

* A Filename

  If `Append` is enabled, the file will be opened in append mode. Otherwise the

contents of the file, if any, will be truncated before any compressed data is written to it.

* A Filehandle

If Append is enabled, the filehandle will be positioned to the end of the file via a call to seek before any compressed data is written to it. Otherwise the file pointer will not be moved.

When Append is specified, and set to true, it will *append* all compressed data to the output data stream.

So when the output is a filehandle it will carry out a seek to the eof before writing any compressed data. If the output is a filename, it will be opened for appending. If the output is a buffer, all compressed data will be appended to the existing buffer.

Conversely when Append is not specified, or it is present and is set to false, it will operate as follows.

When the output is a filename, it will truncate the contents of the file before writing any compressed data. If the output is a filehandle its position will not be changed. If the output is a buffer, it will be wiped before any compressed data is output.

Defaults to 0.

Examples

To read the contents of the file `file1.txt` and write the compressed data to the file `file1.txt.1950`.

```
use strict ;
use warnings ;
use IO::Compress::Deflate qw(deflate $DeflateError) ;

my $input = "file1.txt";
deflate $input => "$input.1950"
    or die "deflate failed: $DeflateError\n";
```

To read from an existing Perl filehandle, `$input`, and write the compressed data to a buffer, `$buffer`.

```
use strict ;
use warnings ;
use IO::Compress::Deflate qw(deflate $DeflateError) ;
use IO::File ;

my $input = new IO::File "<file1.txt"
    or die "Cannot open 'file1.txt': $!\n" ;
my $buffer ;
deflate $input => \$buffer
    or die "deflate failed: $DeflateError\n";
```

To compress all files in the directory "/my/home" that match "*.txt" and store the compressed data in the same directory

```
use strict ;
use warnings ;
use IO::Compress::Deflate qw(deflate $DeflateError) ;

deflate '</my/home/*.txt>' => '<*.1950>'
```

```
    or die "deflate failed: $DeflateError\n";
```

and if you want to compress each file one at a time, this will do the trick

```
use strict ;
use warnings ;
use IO::Compress::Deflate qw(deflate $DeflateError) ;

for my $input ( glob "/my/home/*.txt" )
{
    my $output = "$input.1950" ;
    deflate $input => $output
        or die "Error compressing '$input': $DeflateError\n";
}
```

OO Interface

Constructor

The format of the constructor for `IO::Compress::Deflate` is shown below

```
my $z = new IO::Compress::Deflate $output [,OPTS]
    or die "IO::Compress::Deflate failed: $DeflateError\n";
```

It returns an `IO::Compress::Deflate` object on success and undef on failure. The variable `$DeflateError` will contain an error message on failure.

If you are running Perl 5.005 or better the object, `$z`, returned from IO::Compress::Deflate can be used exactly like an *IO::File* filehandle. This means that all normal output file operations can be carried out with `$z`. For example, to write to a compressed file/buffer you can use either of these forms

```
$z->print("hello world\n");
print $z "hello world\n";
```

The mandatory parameter `$output` is used to control the destination of the compressed data. This parameter can take one of these forms.

A filename

If the `$output` parameter is a simple scalar, it is assumed to be a filename. This file will be opened for writing and the compressed data will be written to it.

A filehandle

If the `$output` parameter is a filehandle, the compressed data will be written to it. The string '-' can be used as an alias for standard output.

A scalar reference

If `$output` is a scalar reference, the compressed data will be stored in `$$output`.

If the `$output` parameter is any other type, `IO::Compress::Deflate`::new will return undef.

Constructor Options

`OPTS` is any combination of the following options:

`AutoClose => 0|1`

This option is only valid when the `$output` parameter is a filehandle. If specified, and the value is true, it will result in the `$output` being closed once either the `close` method is

called or the IO::Compress::Deflate object is destroyed.

This parameter defaults to 0.

Append => 0|1

Opens $output in append mode.

The behaviour of this option is dependent on the type of $output.

* A Buffer

If $output is a buffer and Append is enabled, all compressed data will be append to the end of $output. Otherwise $output will be cleared before any data is written to it.

* A Filename

If $output is a filename and Append is enabled, the file will be opened in append mode. Otherwise the contents of the file, if any, will be truncated before any compressed data is written to it.

* A Filehandle

If $output is a filehandle, the file pointer will be positioned to the end of the file via a call to seek before any compressed data is written to it. Otherwise the file pointer will not be moved.

This parameter defaults to 0.

Merge => 0|1

This option is used to compress input data and append it to an existing compressed data stream in $output. The end result is a single compressed data stream stored in $output.

It is a fatal error to attempt to use this option when $output is not an RFC 1950 data stream.

There are a number of other limitations with the Merge option:

1 This module needs to have been built with zlib 1.2.1 or better to work. A fatal error will be thrown if Merge is used with an older version of zlib.

2 If $output is a file or a filehandle, it must be seekable.

This parameter defaults to 0.

-Level

Defines the compression level used by zlib. The value should either be a number between 0 and 9 (0 means no compression and 9 is maximum compression), or one of the symbolic constants defined below.

```
Z_NO_COMPRESSION
Z_BEST_SPEED
Z_BEST_COMPRESSION
Z_DEFAULT_COMPRESSION
```

The default is Z_DEFAULT_COMPRESSION.

Note, these constants are not imported by IO::Compress::Deflate by default.

```
use IO::Compress::Deflate qw(:strategy);
use IO::Compress::Deflate qw(:constants);
use IO::Compress::Deflate qw(:all);
```

-Strategy

Defines the strategy used to tune the compression. Use one of the symbolic constants

defined below.

```
Z_FILTERED
Z_HUFFMAN_ONLY
Z_RLE
Z_FIXED
Z_DEFAULT_STRATEGY
```

The default is Z_DEFAULT_STRATEGY.

`Strict => 0|1`

This is a placeholder option.

Examples

TODO

Methods

print

Usage is

```
$z->print($data)
print $z $data
```

Compresses and outputs the contents of the `$data` parameter. This has the same behaviour as the `print` built-in.

Returns true if successful.

printf

Usage is

```
$z->printf($format, $data)
printf $z $format, $data
```

Compresses and outputs the contents of the `$data` parameter.

Returns true if successful.

syswrite

Usage is

```
$z->syswrite $data
$z->syswrite $data, $length
$z->syswrite $data, $length, $offset
```

Compresses and outputs the contents of the `$data` parameter.

Returns the number of uncompressed bytes written, or `undef` if unsuccessful.

write

Usage is

```
$z->write $data
$z->write $data, $length
$z->write $data, $length, $offset
```

Compresses and outputs the contents of the `$data` parameter.

Returns the number of uncompressed bytes written, or `undef` if unsuccessful.

flush

Usage is

```
$z->flush;
$z->flush($flush_type);
```

Flushes any pending compressed data to the output file/buffer.

This method takes an optional parameter, `$flush_type`, that controls how the flushing will be carried out. By default the `$flush_type` used is `Z_FINISH`. Other valid values for `$flush_type` are `Z_NO_FLUSH`, `Z_SYNC_FLUSH`, `Z_FULL_FLUSH` and `Z_BLOCK`. It is strongly recommended that you only set the `flush_type` parameter if you fully understand the implications of what it does - overuse of `flush` can seriously degrade the level of compression achieved. See the `zlib` documentation for details.

Returns true on success.

tell

Usage is

```
$z->tell()
tell $z
```

Returns the uncompressed file offset.

eof

Usage is

```
$z->eof();
eof($z);
```

Returns true if the `close` method has been called.

seek

```
$z->seek($position, $whence);
seek($z, $position, $whence);
```

Provides a sub-set of the `seek` functionality, with the restriction that it is only legal to seek forward in the output file/buffer. It is a fatal error to attempt to seek backward.

Empty parts of the file/buffer will have NULL (0x00) bytes written to them.

The `$whence` parameter takes one the usual values, namely SEEK_SET, SEEK_CUR or SEEK_END.

Returns 1 on success, 0 on failure.

binmode

Usage is

```
$z->binmode
binmode $z ;
```

This is a noop provided for completeness.

opened

```
$z->opened()
```

Returns true if the object currently refers to a opened file/buffer.

autoflush

```
my $prev = $z->autoflush()
my $prev = $z->autoflush(EXPR)
```

If the `$z` object is associated with a file or a filehandle, this method returns the current autoflush setting for the underlying filehandle. If EXPR is present, and is non-zero, it will enable flushing after every write/print operation.

If `$z` is associated with a buffer, this method has no effect and always returns `undef`.

Note that the special variable `$|` **cannot** be used to set or retrieve the autoflush setting.

input_line_number

```
$z->input_line_number()
$z->input_line_number(EXPR)
```

This method always returns `undef` when compressing.

fileno

```
$z->fileno()
fileno($z)
```

If the `$z` object is associated with a file or a filehandle, `fileno` will return the underlying file descriptor. Once the `close` method is called `fileno` will return `undef`.

If the `$z` object is associated with a buffer, this method will return `undef`.

close

```
$z->close() ;
close $z ;
```

Flushes any pending compressed data and then closes the output file/buffer.

For most versions of Perl this method will be automatically invoked if the IO::Compress::Deflate object is destroyed (either explicitly or by the variable with the reference to the object going out of scope). The exceptions are Perl versions 5.005 through 5.00504 and 5.8.0. In these cases, the `close` method will be called automatically, but not until global destruction of all live objects when the program is terminating.

Therefore, if you want your scripts to be able to run on all versions of Perl, you should call `close` explicitly and not rely on automatic closing.

Returns true on success, otherwise 0.

If the `AutoClose` option has been enabled when the IO::Compress::Deflate object was created, and the object is associated with a file, the underlying file will also be closed.

newStream([OPTS])

Usage is

```
$z->newStream( [OPTS] )
```

Closes the current compressed data stream and starts a new one.

OPTS consists of any of the options that are available when creating the `$z` object.

See the *Constructor Options* section for more details.

deflateParams

Usage is

```
$z->deflateParams
```

TODO

Importing

A number of symbolic constants are required by some methods in `IO::Compress::Deflate`. None are imported by default.

:all

Imports `deflate`, `$DeflateError` and all symbolic constants that can be used by `IO::Compress::Deflate`. Same as doing this

```
use IO::Compress::Deflate qw(deflate $DeflateError :constants) ;
```

:constants

Import all symbolic constants. Same as doing this

```
use IO::Compress::Deflate qw(:flush :level :strategy) ;
```

:flush

These symbolic constants are used by the `flush` method.

```
Z_NO_FLUSH
Z_PARTIAL_FLUSH
Z_SYNC_FLUSH
Z_FULL_FLUSH
Z_FINISH
Z_BLOCK
```

:level

These symbolic constants are used by the `Level` option in the constructor.

```
Z_NO_COMPRESSION
Z_BEST_SPEED
Z_BEST_COMPRESSION
Z_DEFAULT_COMPRESSION
```

:strategy

These symbolic constants are used by the `Strategy` option in the constructor.

```
Z_FILTERED
Z_HUFFMAN_ONLY
Z_RLE
Z_FIXED
Z_DEFAULT_STRATEGY
```

EXAMPLES

Apache::GZip Revisited

See *IO::Compress::FAQ*

Working with Net::FTP

See *IO::Compress::FAQ*

SEE ALSO

Compress::Zlib, *IO::Compress::Gzip*, *IO::Uncompress::Gunzip*, *IO::Uncompress::Inflate*, *IO::Compress::RawDeflate*, *IO::Uncompress::RawInflate*, *IO::Compress::Bzip2*, *IO::Uncompress::Bunzip2*, *IO::Compress::Lzma*, *IO::Uncompress::UnLzma*, *IO::Compress::Xz*, *IO::Uncompress::UnXz*, *IO::Compress::Lzop*, *IO::Uncompress::UnLzop*, *IO::Compress::Lzf*, *IO::Uncompress::UnLzf*, *IO::Uncompress::AnyInflate*, *IO::Uncompress::AnyUncompress*

IO::Compress::FAQ

File::GlobMapper, *Archive::Zip*, *Archive::Tar*, *IO::Zlib*

For RFC 1950, 1951 and 1952 see *http://www.faqs.org/rfcs/rfc1950.html*, *http://www.faqs.org/rfcs/rfc1951.html* and *http://www.faqs.org/rfcs/rfc1952.html*

The *zlib* compression library was written by Jean-loup Gailly *gzip@prep.ai.mit.edu* and Mark Adler *madler@alumni.caltech.edu*.

The primary site for the *zlib* compression library is *http://www.zlib.org*.

The primary site for gzip is *http://www.gzip.org*.

AUTHOR

This module was written by Paul Marquess, *pmqs@cpan.org*.

MODIFICATION HISTORY

See the Changes file.

COPYRIGHT AND LICENSE