NAME

Compress::Raw::Bzip2 - Low-Level Interface to bzip2 compression library

SYNOPSIS

```
use Compress::Raw::Bzip2 ;

my ($bz, $status) = new Compress::Raw::Bzip2 [OPTS]
    or die "Cannot create bzip2 object: $bzerno\n";

$status = $bz->bzdeflate($input, $output);
$status = $bz->bzflush($output);
$status = $bz->bzclose($output);

my ($bz, $status) = new Compress::Raw::Bunzip2 [OPTS]
    or die "Cannot create bunzip2 object: $bzerno\n";

$status = $bz->bzinflate($input, $output);

my $version = Compress::Raw::Bzip2::bzlibversion();
```

DESCRIPTION

`Compress::Raw::Bzip2` provides an interface to the in-memory compression/uncompression functions from the bzip2 compression library.

Although the primary purpose for the existence of `Compress::Raw::Bzip2` is for use by the `IO::Compress::Bzip2` and `IO::Compress::Bunzip2` modules, it can be used on its own for simple compression/uncompression tasks.

Compression

($z, $status) = new Compress::Raw::Bzip2 $appendOutput, $blockSize100k, $workfactor;

Creates a new compression object.

If successful, it will return the initialised compression object, `$z` and a `$status` of `BZ_OK` in a list context. In scalar context it returns the deflation object, `$z`, only.

If not successful, the returned compression object, `$z`, will be *undef* and `$status` will hold the a *bzip2* error code.

Below is a list of the valid options:

$appendOutput

Controls whether the compressed data is appended to the output buffer in the `bzdeflate`, `bzflush` and `bzclose` methods.

Defaults to 1.

$blockSize100k

To quote the bzip2 documentation

```
    blockSize100k specifies the block size to be used for
compression. It
    should be a value between 1 and 9 inclusive, and the actual
block size
    used is 100000 x this figure. 9 gives the best compression but
takes
    most memory.
```

Defaults to 1.

$workfactor

To quote the bzip2 documentation

```
    This parameter controls how the compression phase behaves when
    presented with worst case, highly repetitive, input data. If
    compression runs into difficulties caused by repetitive data,
the
    library switches from the standard sorting algorithm to a
fallback
    algorithm. The fallback is slower than the standard algorithm by
    perhaps a factor of three, but always behaves reasonably, no
matter how
    bad the input.

    Lower values of workFactor reduce the amount of effort the
standard
    algorithm will expend before resorting to the fallback. You
should set
    this parameter carefully; too low, and many inputs will be
handled by
    the fallback algorithm and so compress rather slowly, too high,
and
    your average-to-worst case compression times can become very
large. The
    default value of 30 gives reasonable behaviour over a wide range
 of
    circumstances.

    Allowable values range from 0 to 250 inclusive. 0 is a special
case,
    equivalent to using the default value of 30.
```

Defaults to 0.

$status = $bz->bzdeflate($input, $output);

Reads the contents of `$input`, compresses it and writes the compressed data to `$output`.

Returns `BZ_RUN_OK` on success and a `bzip2` error code on failure.

If `appendOutput` is enabled in the constructor for the bzip2 object, the compressed data will be appended to `$output`. If not enabled, `$output` will be truncated before the compressed data is written to it.

$status = $bz->bzflush($output);

Flushes any pending compressed data to `$output`.

Returns `BZ_RUN_OK` on success and a `bzip2` error code on failure.

$status = $bz->bzclose($output);

Terminates the compressed data stream and flushes any pending compressed data to `$output`.

Returns `BZ_STREAM_END` on success and a `bzip2` error code on failure.

Example

Uncompression

($z, $status) = new Compress::Raw::Bunzip2 $appendOutput, $consumeInput, $small, $limitOutput;

If successful, it will return the initialised uncompression object, `$z` and a `$status` of `BZ_OK` in a list context. In scalar context it returns the deflation object, `$z`, only.

If not successful, the returned uncompression object, `$z`, will be *undef* and `$status` will hold the a *bzip2* error code.

Below is a list of the valid options:

$appendOutput

Controls whether the compressed data is appended to the output buffer in the `bzinflate`, `bzflush` and `bzclose` methods.

Defaults to 1.

$consumeInput

$small

To quote the bzip2 documentation

```
    If small is nonzero, the library will use an alternative
decompression
    algorithm which uses less memory but at the cost of
decompressing more
    slowly (roughly speaking, half the speed, but the maximum memory
    requirement drops to around 2300k).
```

Defaults to 0.

$limitOutput

The `LimitOutput` option changes the behavior of the `$i->bzinflate` method so that the amount of memory used by the output buffer can be limited.

When `LimitOutput` is used the size of the output buffer used will either be the 16k or the amount of memory already allocated to `$output`, whichever is larger. Predicting the output size available is tricky, so don't rely on getting an exact output buffer size.

When `LimitOutout` is not specified `$i->bzinflate` will use as much memory as it takes to write all the uncompressed data it creates by uncompressing the input buffer.

If `LimitOutput` is enabled, the `ConsumeInput` option will also be enabled.

This option defaults to false.

$status = $z->bzinflate($input, $output);

Uncompresses `$input` and writes the uncompressed data to `$output`.

Returns `BZ_OK` if the uncompression was successful, but the end of the compressed data stream has not been reached. Returns `BZ_STREAM_END` on successful uncompression and the end of the compression stream has been reached.

If `consumeInput` is enabled in the constructor for the bunzip2 object, `$input` will have all compressed data removed from it after uncompression. On `BZ_OK` return this will mean that `$input` will be an empty string; when `BZ_STREAM_END` `$input` will either be an empty string or will contain whatever data immediately followed the compressed data stream.

If `appendOutput` is enabled in the constructor for the bunzip2 object, the uncompressed data will be appended to `$output`. If not enabled, `$output` will be truncated before the uncompressed data is written to it.

Misc

my $version = Compress::Raw::Bzip2::bzlibversion();

Returns the version of the underlying bzip2 library.

Constants

The following bzip2 constants are exported by this module

```
BZ_RUN
BZ_FLUSH
BZ_FINISH

BZ_OK
BZ_RUN_OK
BZ_FLUSH_OK
BZ_FINISH_OK
BZ_STREAM_END
BZ_SEQUENCE_ERROR
BZ_PARAM_ERROR
BZ_MEM_ERROR
BZ_DATA_ERROR
BZ_DATA_ERROR_MAGIC
BZ_IO_ERROR
BZ_UNEXPECTED_EOF
BZ_OUTBUFF_FULL
BZ_CONFIG_ERROR
```

SEE ALSO

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

Compress::Zlib::FAQ

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

The primary site for the bzip2 program is *http://www.bzip.org*.

See the module *Compress::Bzip2*

AUTHOR

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

MODIFICATION HISTORY

See the Changes file.

COPYRIGHT AND LICENSE