Commit c75ae9c

pmqs authored and **jkeenan** committed last month · 34 / 35

kid51

```
cpan/Compress-Raw-Zlib - Update to version 2.221

  2.221 27 February 2026

    * Merge remote-tracking branch 'refs/remotes/origin/master'
      Fri Feb 27 12:57:24 2026 +0000
      34f0318c9687924347284704c6f3b9b2566e07fd

    * Merge pull request #39 from jkeenan/correct-changes-date-20260227
      Fri Feb 27 12:50:53 2026 +0000
      e879453ec96d85b2917ed903a3f03cf8264238f1

    * Update to 2.221
      Fri Feb 27 12:44:53 2026 +0000
      17f707ed1986acf6773d91ebf6b5c7d6ba6d32be

    * Version number wrong in Changes file. Fixes #38
      Fri Feb 27 12:38:32 2026 +0000
      2c10524538077d20779c483c667e2ead4c65a87a

  2.220 25 February 2026

    * Update to version 2.220
      Wed Feb 25 13:50:23 2026 +0000
      f9ac640e71a54ad2f8156a075127f8f99ad45586

    * zlib 1.3.2: Update references to zlib 1.3.1 to use zlib 1.3.2
      Wed Feb 25 13:45:59 2026 +0000
      16b5fc92e830af8b10c37e6a3e66806cc0db937f

    * zlib 1.3.2: Add "Perl_crz" prefix to "z_errmsg". Fixes #32
      Wed Feb 25 13:41:07 2026 +0000
      36b42f013ed5d3e6b27ce956c97b8ce5733cfe24

    * Fix spelling typo
      Tue Feb 24 19:46:13 2026 +0000
      f73b8114faad246cf5c615c7e77e36d8978393bd

  2.219 23 February 2026

      * Update to version 2.219
        Mon Feb 23 15:19:30 2026 +0000
        c257a0b73e5f925549e20602cd42f9beea9db51e
```

```
    * zlib 1.3.2:  Add a few casts to allow zlib sources to build with C++
      Mon Feb 23 14:29:02 2026 +0000
      0049fc058a9b9c1de250b00af38edf3e91cb15eb

    * Test workflows with upstream zlib 1.32
      Mon Feb 23 14:15:02 2026 +0000
      d3b865abdb11c0479d04f8b66a6350b7aad9d5b3

      * zlib 1.3.2: Add "Perl_crz_" prefix  to exported symbols
      Mon Feb 23 14:02:50 2026 +0000
      37cec4a39cc6080a29060944db89f90c614c1cd4

    * zlib-1.3.2: Force include of <stddef.h> to get definition for NULL
      Mon Feb 23 13:38:39 2026 +0000
      8907f13c905eb3a9e65514bfbbf35304a92fa76a

    * Refresh zlib-src directory with unmodified zlib-1.3.2 files
      Mon Feb 23 13:31:42 2026 +0000
      48a5180b706c1066af212656e32d73c74850c408
```

blead · v5.43.9

1 parent bf65533 commit c75ae9c

19 files changed +939 -722 lines changed

Top

Filter files…

- Porting
  - Maintainers.pl
- cpan/Compress-Raw-Zlib
  - lib/Compress/Raw
    - Zlib.pm
  - zlib-src
    - compress.c
    - crc32.c
    - deflate.c
    - deflate.h
    - infback.c
    - inffast.c

- inffixed.h
- inflate.c
- inflate.h
- inftrees.c
- inftrees.h
- trees.c
- uncompr.c
- zconf.h
- zlib.h
- zutil.c
- zutil.h

19 files changed +939 -722 lines changed

Search within code

Porting/Maintainers.pl

```
@@ -231,8 +231,8 @@ package Maintainers;
231 231         },
232 232
233 233         'Compress::Raw::Zlib' => {
234     -           'DISTRIBUTION' => 'PMQS/Compress-Raw-Zlib-2.218.tar.gz',
235     -           'SYNCINFO'     => 'jkeenan on Wed Feb 11 11:27:39 2026',
    234 +           'DISTRIBUTION' => 'PMQS/Compress-Raw-Zlib-2.221.tar.gz',
    235 +           'SYNCINFO'     => 'jkeenan on Fri Feb 27 10:16:27 2026',
236 236             'FILES'    => q[cpan/Compress-Raw-Zlib],
237 237             'EXCLUDED' => [
238 238                 qr{^examples/},
```

.../Compress-Raw-Zlib/lib/Compress/Raw/Zlib.pm

```
@@ -10,7 +10,7 @@ use warnings ;
10 10  use bytes ;
11 11  our ($VERSION, $XS_VERSION, @ISA, @EXPORT, %EXPORT_TAGS, @EXPORT_OK, $AUTOLOAD,
       %DEFLATE_CONSTANTS, @DEFLATE_CONSTANTS);
12 12
13     - $VERSION = '2.218';
```

```
      13 + $VERSION = '2.221';
14    14   $XS_VERSION = $VERSION;
15    15   $VERSION = eval $VERSION;
16    16
```

cpan/Compress-Raw-Zlib/zlib-src/compress.c

```
      ••• @@ -1,5 +1,5 @@
1     1    /* compress.c -- compress a memory buffer
2           - * Copyright (C) 1995-2005, 2014, 2016 Jean-loup Gailly, Mark Adler
      2    + * Copyright (C) 1995-2026 Jean-loup Gailly, Mark Adler
3     3     * For conditions of distribution and use, see copyright notice in zlib.h
4     4     */
5     5
      @@ -18,13 +18,19 @@
18    18         compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
19    19       memory, Z_BUF_ERROR if there was not enough room in the output buffer,
20    20       Z_STREAM_ERROR if the level parameter is invalid.
      21   +
      22   +     The _z versions of the functions take size_t length arguments.
21    23   */
22          - int ZEXPORT compress2(Bytef *dest, uLongf *destLen, const Bytef *source,
23          - uLong sourceLen, int level) {
      24   + int ZEXPORT compress2_z(Bytef *dest, z_size_t *destLen, const Bytef *source,
      25   +                         z_size_t sourceLen, int level) {
24    26       z_stream stream;
25    27       int err;
26    28       const uInt max = (uInt)-1;
27          -    uLong left;
      29   +    z_size_t left;
      30   +
      31   +    if ((sourceLen > 0 && source == NULL) ||
      32   +        destLen == NULL || (*destLen > 0 && dest == NULL))
      33   +        return Z_STREAM_ERROR;
28    34
29    35       left = *destLen;
30    36       *destLen = 0;
      @@ -43,23 +49,36 @@ int ZEXPORT compress2(Bytef *dest, uLongf *destLen, const
      Bytef *source,
43    49
```

```
44  50          do {
45  51              if (stream.avail_out == 0) {
46      -               stream.avail_out = left > (uLong)max ? max : (uInt)left;
    52  +               stream.avail_out = left > (z_size_t)max ? max : (uInt)left;
47  53                  left -= stream.avail_out;
48  54              }
49  55              if (stream.avail_in == 0) {
50      -               stream.avail_in = sourceLen > (uLong)max ? max : (uInt)sourceLen;
    56  +               stream.avail_in = sourceLen > (z_size_t)max ? max :
    57  +                                                            (uInt)sourceLen;
51  58                  sourceLen -= stream.avail_in;
52  59              }
53  60              err = deflate(&stream, sourceLen ? Z_NO_FLUSH : Z_FINISH);
54  61          } while (err == Z_OK);
55  62
56      -       *destLen = stream.total_out;
    63  +       *destLen = (z_size_t)(stream.next_out - dest);
57  64          deflateEnd(&stream);
58  65          return err == Z_STREAM_END ? Z_OK : err;
59  66      }
60      -
    67  + int ZEXPORT compress2(Bytef *dest, uLongf *destLen, const Bytef *source,
    68  +                       uLong sourceLen, int level) {
    69  +     int ret;
    70  +     z_size_t got = *destLen;
    71  +     ret = compress2_z(dest, &got, source, sourceLen, level);
    72  +     *destLen = (uLong)got;
    73  +     return ret;
    74  + }
61  75    /* ===========================================================================
62  76     */
    77  + int ZEXPORT compress_z(Bytef *dest, z_size_t *destLen, const Bytef *source,
    78  +                        z_size_t sourceLen) {
    79  +     return compress2_z(dest, destLen, source, sourceLen,
    80  +                        Z_DEFAULT_COMPRESSION);
    81  + }
63  82    int ZEXPORT compress(Bytef *dest, uLongf *destLen, const Bytef *source,
64  83                         uLong sourceLen) {
65  84          return compress2(dest, destLen, source, sourceLen, Z_DEFAULT_COMPRESSION);
```

```
@@ -69,7 +88,12 @@ int ZEXPORT compress(Bytef *dest, uLongf *destLen, const Bytef *source,
 69  88       If the default memLevel or windowBits for deflateInit() is changed, then
 70  89     this function needs to be updated.
 71  90    */
     91  + z_size_t ZEXPORT compressBound_z(z_size_t sourceLen) {
     92  +     z_size_t bound = sourceLen + (sourceLen >> 12) + (sourceLen >> 14) +
     93  +                      (sourceLen >> 25) + 13;
     94  +     return bound < sourceLen ? (z_size_t)-1 : bound;
     95  + }
 72  96    uLong ZEXPORT compressBound(uLong sourceLen) {
 73      -     return sourceLen + (sourceLen >> 12) + (sourceLen >> 14) +
 74      -            (sourceLen >> 25) + 13;
     97  +     z_size_t bound = compressBound_z(sourceLen);
     98  +     return (uLong)bound != bound ? (uLong)-1 : (uLong)bound;
 75  99    }
```

cpan/Compress-Raw-Zlib/zlib-src/crc32.c

```
@@ -1,5 +1,5 @@
  1   1    /* crc32.c -- compute the CRC-32 of a data stream
  2       -  * Copyright (C) 1995-2022 Mark Adler
      2  +  * Copyright (C) 1995-2026 Mark Adler
  3   3     * For conditions of distribution and use, see copyright notice in zlib.h
  4   4     *
  5   5     * This interleaved implementation of a CRC makes use of pipelined multiple
@@ -24,11 +24,18 @@
 24  24    #  include <stdio.h>
 25  25    #  ifndef DYNAMIC_CRC_TABLE
 26  26    #    define DYNAMIC_CRC_TABLE
 27      - #  endif /* !DYNAMIC_CRC_TABLE */
 28      - #endif /* MAKECRCH */
     27  + #  endif
     28  + #endif
     29  + #ifdef DYNAMIC_CRC_TABLE
     30  + #  define Z_ONCE
     31  + #endif
 29  32
 30  33    #include "zutil.h"      /* for Z_U4, Z_U8, z_crc_t, and FAR definitions */
 31  34
     35  + #ifdef HAVE_S390X_VX
```

```
          36 + #  include "contrib/crc32vx/crc32_vx_hooks.h"
          37 + #endif
          38 +
 32       39    /*
 33       40      A CRC of a message is computed on N braids of words in the message, where
 34       41      each word consists of W bytes (4 or 8). If N is 3, for example, then three
                  @@ -99,7 +106,8 @@
 99      106    #endif
100      107
101      108    /* If available, use the ARM processor CRC32 instruction. */
102          - #if defined(__aarch64__) && defined(__ARM_FEATURE_CRC32) && W == 8
          109 + #if defined(__aarch64__) && defined(__ARM_FEATURE_CRC32) && \
          110 +     defined(W) && W == 8
103      111    #  define ARMCRC32
104      112    #endif
105      113
                  @@ -152,10 +160,10 @@ local z_word_t byte_swap(z_word_t word) {
152      160      Return a(x) multiplied by b(x) modulo p(x), where p(x) is the CRC
                  polynomial,
153      161      reflected. For speed, this requires that a not be zero.
154      162     */
155          - local z_crc_t multmodp(z_crc_t a, z_crc_t b) {
156          -     z_crc_t m, p;
          163 + local uLong multmodp(uLong a, uLong b) {
          164 +     uLong m, p;
157      165
158          -     m = (z_crc_t)1 << 31;
          166 +     m = (uLong)1 << 31;
159      167       p = 0;
160      168       for (;;) {
161      169           if (a & m) {
                  @@ -171,12 +179,12 @@ local z_crc_t multmodp(z_crc_t a, z_crc_t b) {
171      179
172      180    /*
173      181      Return x^(n * 2^k) modulo p(x). Requires that x2n_table[] has been
174          -   initialized.
          182 +   initialized. n must not be negative.
175      183     */
```

```
176       - local z_crc_t x2nmodp(z_off64_t n, unsigned k) {
177       -     z_crc_t p;
      184 + local uLong x2nmodp(z_off64_t n, unsigned k) {
      185 +     uLong p;
178   186
179       -     p = (z_crc_t)1 << 31;           /* x^0 == 1 */
      187 +     p = (uLong)1 << 31;             /* x^0 == 1 */
180   188       while (n) {
181   189           if (n & 1)
182   190               p = multmodp(x2n_table[k & 31], p);
          @@ -204,83 +212,8 @@ local z_crc_t FAR crc_table[256];
204   212       local void write_table64(FILE *, const z_word_t FAR *, int);
205   213   #endif /* MAKECRCH */
206   214
207       - /*
208       -   Define a once() function depending on the availability of atomics. If this
          is
209       -   compiled with DYNAMIC_CRC_TABLE defined, and if CRCs will be computed in
210       -   multiple threads, and if atomics are not available, then get_crc_table()
          must
211       -   be called to initialize the tables and must return before any threads are
212       -   allowed to compute or combine CRCs.
213       -  */
214       -
215       - /* Definition of once functionality. */
216       - typedef struct once_s once_t;
217       -
218       - /* Check for the availability of atomics. */
219       - #if defined(__STDC__) && __STDC_VERSION__ >= 201112L && \
220       -     !defined(__STDC_NO_ATOMICS__)
221       -
222       - #include <stdatomic.h>
223       -
224       - /* Structure for once(), which must be initialized with ONCE_INIT. */
225       - struct once_s {
226       -     atomic_flag begun;
227       -     atomic_int done;
228       - };
229       - #define ONCE_INIT {ATOMIC_FLAG_INIT, 0}
```

```
-
- /*
-   Run the provided init() function exactly once, even if multiple threads
-   invoke once() at the same time. The state must be a once_t initialized with
-   ONCE_INIT.
-  */
- local void once(once_t *state, void (*init)(void)) {
-     if (!atomic_load(&state->done)) {
-         if (atomic_flag_test_and_set(&state->begun))
-             while (!atomic_load(&state->done))
-                 ;
-         else {
-             init();
-             atomic_store(&state->done, 1);
-         }
-     }
- }
-
- #else   /* no atomics */
-
- /* Structure for once(), which must be initialized with ONCE_INIT. */
- struct once_s {
-     volatile int begun;
-     volatile int done;
- };
- #define ONCE_INIT {0, 0}
-
- /* Test and set. Alas, not atomic, but tries to minimize the period of
-    vulnerability. */
- local int test_and_set(int volatile *flag) {
-     int was;
-
-     was = *flag;
-     *flag = 1;
-     return was;
- }
-
- /* Run the provided init() function once. This is not thread-safe. */
- local void once(once_t *state, void (*init)(void)) {
-     if (!state->done) {
```

```
270      -          if (test_and_set(&state->begun))
271      -              while (!state->done)
272      -                  ;
273      -          else {
274      -              init();
275      -              state->done = 1;
276      -          }
277      -      }
278      - }
279      -
280      - #endif
281      -
282  215     /* State for once(). */
283      - local once_t made = ONCE_INIT;
     216 + local z_once_t made = Z_ONCE_INIT;
284  217
285  218     /*
286  219       Generate tables for a byte-wise 32-bit CRC calculation on the polynomial:
         ↓
         ↑   @@ -326,7 +259,7 @@ local void make_crc_table(void) {
326  259         p = (z_crc_t)1 << 30;         /* x^1 */
327  260         x2n_table[0] = p;
328  261         for (n = 1; n < 32; n++)
329      -           x2n_table[n] = p = multmodp(p, p);
     262 +           x2n_table[n] = p = (z_crc_t)multmodp(p, p);
330  263
331  264     #ifdef W
332  265         /* initialize the braiding tables -- needs x2n_table[] */
         ↓
         ↑   @@ -529,11 +462,11 @@ local void braid(z_crc_t ltl[][256], z_word_t big[]
             [256], int n, int w) {
529  462         int k;
530  463         z_crc_t i, p, q;
531  464         for (k = 0; k < w; k++) {
532      -           p = x2nmodp((n * w + 3 - k) << 3, 0);
     465 +           p = (z_crc_t)x2nmodp((n * w + 3 - k) << 3, 0);
533  466             ltl[k][0] = 0;
534  467             big[w - 1 - k][0] = 0;
535  468             for (i = 1; i < 256; i++) {
536      -               ltl[k][i] = q = multmodp(i << 24, p);
     469 +               ltl[k][i] = q = (z_crc_t)multmodp(i << 24, p);
```

```
537  470                 big[w - 1 - k][i] = byte_swap(q);
538  471             }
539  472         }
     @@ -548,7 +481,7 @@ local void braid(z_crc_t ltl[][256], z_word_t big[]
     [256], int n, int w) {
548  481      */
549  482     const z_crc_t FAR * ZEXPORT get_crc_table(void) {
550  483     #ifdef DYNAMIC_CRC_TABLE
551       -     once(&made, make_crc_table);
     484  +     z_once(&made, make_crc_table);
552  485     #endif /* DYNAMIC_CRC_TABLE */
553  486         return (const z_crc_t FAR *)crc_table;
554  487     }
     @@ -572,9 +505,8 @@ const z_crc_t FAR * ZEXPORT get_crc_table(void) {
572  505     #define Z_BATCH_ZEROS 0xa10d3d0c    /* computed from Z_BATCH = 3990 */
573  506     #define Z_BATCH_MIN 800             /* fewest words in a final batch */
574  507
575       - unsigned long ZEXPORT crc32_z(unsigned long crc, const unsigned char FAR
          *buf,
576       -                               z_size_t len) {
577       -     z_crc_t val;
     508  + uLong ZEXPORT crc32_z(uLong crc, const unsigned char FAR *buf, z_size_t len)
          {
     509  +     uLong val;
578  510         z_word_t crc1, crc2;
579  511         const z_word_t *word;
580  512         z_word_t val0, val1, val2;
     @@ -585,7 +517,7 @@ unsigned long ZEXPORT crc32_z(unsigned long crc,
     const unsigned char FAR *buf,
585  517         if (buf == Z_NULL) return 0;
586  518
587  519     #ifdef DYNAMIC_CRC_TABLE
588       -     once(&made, make_crc_table);
     520  +     z_once(&made, make_crc_table);
589  521     #endif /* DYNAMIC_CRC_TABLE */
590  522
591  523         /* Pre-condition the CRC */
     @@ -640,7 +572,7 @@ unsigned long ZEXPORT crc32_z(unsigned long crc,
     const unsigned char FAR *buf,
640  572             }
```

```
641  573            word += 3 * last;
642  574            num -= 3 * last;
643      -          val = x2nmodp(last, 6);
     575  +          val = x2nmodp((int)last, 6);
644  576            crc = multmodp(val, crc) ^ crc1;
645  577            crc = multmodp(val, crc) ^ crc2;
646  578        }
          @@ -691,13 +623,12 @@ local z_word_t crc_word_big(z_word_t data) {
691  623    #endif
692  624
693  625    /* ========================================================================
               */
694      - unsigned long ZEXPORT crc32_z(unsigned long crc, const unsigned char FAR
               *buf,
695      -                               z_size_t len) {
     626  + uLong ZEXPORT crc32_z(uLong crc, const unsigned char FAR *buf, z_size_t len)
               {
696  627        /* Return initial CRC, if requested. */
697  628        if (buf == Z_NULL) return 0;
698  629
699  630    #ifdef DYNAMIC_CRC_TABLE
700      -     once(&made, make_crc_table);
     631  +     z_once(&made, make_crc_table);
701  632    #endif /* DYNAMIC_CRC_TABLE */
702  633
703  634        /* Pre-condition the CRC */
          @@ -1012,38 +943,41 @@ unsigned long ZEXPORT crc32_z(unsigned long crc,
               const unsigned char FAR *buf,
1012 943    #endif
1013 944
1014 945    /* ========================================================================
               */
1015     - unsigned long ZEXPORT crc32(unsigned long crc, const unsigned char FAR *buf,
1016     -                             uInt len) {
     946  + uLong ZEXPORT crc32(uLong crc, const unsigned char FAR *buf, uInt len) {
     947  +     #ifdef HAVE_S390X_VX
     948  +     return crc32_z_hook(crc, buf, len);
     949  +     #endif
1017 950        return crc32_z(crc, buf, len);
```

```
1018  951     }
1019  952
1020  953     /* ========================================================================
                */
1021        - uLong ZEXPORT crc32_combine64(uLong crc1, uLong crc2, z_off64_t len2) {
      954   + uLong ZEXPORT crc32_combine_gen64(z_off64_t len2) {
      955   +     if (len2 < 0)
      956   +         return 0;
1022  957     #ifdef DYNAMIC_CRC_TABLE
1023        -     once(&made, make_crc_table);
      958   +     z_once(&made, make_crc_table);
1024  959     #endif /* DYNAMIC_CRC_TABLE */
1025        -     return multmodp(x2nmodp(len2, 3), crc1) ^ (crc2 & 0xffffffff);
      960   +     return x2nmodp(len2, 3);
1026  961     }
1027  962
1028  963     /* ========================================================================
                */
1029        - uLong ZEXPORT crc32_combine(uLong crc1, uLong crc2, z_off_t len2) {
1030        -     return crc32_combine64(crc1, crc2, (z_off64_t)len2);
      964   + uLong ZEXPORT crc32_combine_gen(z_off_t len2) {
      965   +     return crc32_combine_gen64((z_off64_t)len2);
1031  966     }
1032  967
1033  968     /* ========================================================================
                */
1034        - uLong ZEXPORT crc32_combine_gen64(z_off64_t len2) {
1035        - #ifdef DYNAMIC_CRC_TABLE
1036        -     once(&made, make_crc_table);
1037        - #endif /* DYNAMIC_CRC_TABLE */
1038        -     return x2nmodp(len2, 3);
      969   + uLong ZEXPORT crc32_combine_op(uLong crc1, uLong crc2, uLong op) {
      970   +     if (op == 0)
      971   +         return 0;
      972   +     return multmodp(op, crc1 & 0xffffffff) ^ (crc2 & 0xffffffff);
1039  973     }
1040  974
1041  975     /* ========================================================================
                */
1042        - uLong ZEXPORT crc32_combine_gen(z_off_t len2) {
```

```
1043      -     return crc32_combine_gen64((z_off64_t)len2);
          976 + uLong ZEXPORT crc32_combine64(uLong crc1, uLong crc2, z_off64_t len2) {
          977 +     return crc32_combine_op(crc1, crc2, crc32_combine_gen64(len2));
1044      978   }
1045      979
1046      980   /* ========================================================================
                */
1047      -  uLong ZEXPORT crc32_combine_op(uLong crc1, uLong crc2, uLong op) {
1048      -      return multmodp(op, crc1) ^ (crc2 & 0xffffffff);
          981 +  uLong ZEXPORT crc32_combine(uLong crc1, uLong crc2, z_off_t len2) {
          982 +      return crc32_combine64(crc1, crc2, (z_off64_t)len2);
1049      983   }
```

cpan/Compress-Raw-Zlib/zlib-src/deflate.c

Load Diff

Large diffs are not rendered by default.

cpan/Compress-Raw-Zlib/zlib-src/deflate.h

```
          ...   @@ -1,5 +1,5 @@
1         1     /* deflate.h -- internal compression state
2         -   * Copyright (C) 1995-2024 Jean-loup Gailly
          2   * Copyright (C) 1995-2026 Jean-loup Gailly
3         3     * For conditions of distribution and use, see copyright notice in zlib.h
4         4     */
5         5
                @@ -271,6 +271,9 @@ typedef struct internal_state {
271       271       /* Number of valid bits in bi_buf.  All bits above the last valid bit
272       272        * are always zero.
273       273        */
          274 +     int bi_used;
          275 +     /* Last number of used bits when going to a byte boundary.
          276 +      */
274       277
275       278       ulg high_water;
276       279       /* High water mark offset in window for initialized bytes -- bytes above
```

```
@@ -279,6 +282,9 @@ typedef struct internal_state {
279 282          * updated to the new high water mark.
280 283          */
281 284
    285 +     int slid;
    286 +     /* True if the hash table has been slid since it was cleared. */
    287 +
282 288     } FAR deflate_state;
283 289
284 290     /* Output a byte on the stream.
```

cpan/Compress-Raw-Zlib/zlib-src/infback.c

```
@@ -1,5 +1,5 @@
1 1     /* infback.c -- inflate using a call-back interface
2       -  * Copyright (C) 1995-2022 Mark Adler
    2 +  * Copyright (C) 1995-2026 Mark Adler
3 3      * For conditions of distribution and use, see copyright notice in zlib.h
4 4      */
5 5
@@ -46,7 +46,7 @@ int ZEXPORT inflateBackInit_(z_streamp strm, int
windowBits,
46 46     #ifdef Z_SOLO
47 47             return Z_STREAM_ERROR;
48 48     #else
49      -     strm->zfree = zcfree;
    49 +     strm->zfree = zcfree;
50 50     #endif
51 51         state = (struct inflate_state FAR *)ZALLOC(strm, 1,
52 52                                                    sizeof(struct inflate_state));
@@ -63,57 +63,6 @@ int ZEXPORT inflateBackInit_(z_streamp strm, int
windowBits,
63 63         return Z_OK;
64 64     }
65 65
66      - /*
67      -    Return state with length and distance decoding tables and index sizes set to
68      -    fixed code decoding.  Normally this returns fixed tables from inffixed.h.
69      -    If BUILDFIXED is defined, then instead this routine builds the tables the
70      -    first time it's called, and returns those tables the first time and
```

```
71  -    thereafter.  This reduces the size of the code by about 2K bytes, in
72  -    exchange for a little execution time.  However, BUILDFIXED should not be
73  -    used for threaded applications, since the rewriting of the tables and virgin
74  -    may not be thread-safe.
75  -  */
76  - local void fixedtables(struct inflate_state FAR *state) {
77  - #ifdef BUILDFIXED
78  -     static int virgin = 1;
79  -     static code *lenfix, *distfix;
80  -     static code fixed[544];
81  -
82  -     /* build fixed huffman tables if first call (may not be thread safe) */
83  -     if (virgin) {
84  -         unsigned sym, bits;
85  -         static code *next;
86  -
87  -         /* literal/length table */
88  -         sym = 0;
89  -         while (sym < 144) state->lens[sym++] = 8;
90  -         while (sym < 256) state->lens[sym++] = 9;
91  -         while (sym < 280) state->lens[sym++] = 7;
92  -         while (sym < 288) state->lens[sym++] = 8;
93  -         next = fixed;
94  -         lenfix = next;
95  -         bits = 9;
96  -         inflate_table(LENS, state->lens, 288, &(next), &(bits), state->work);
97  -
98  -         /* distance table */
99  -         sym = 0;
100 -         while (sym < 32) state->lens[sym++] = 5;
101 -         distfix = next;
102 -         bits = 5;
103 -         inflate_table(DISTS, state->lens, 32, &(next), &(bits), state->work);
104 -
105 -         /* do this just once */
106 -         virgin = 0;
107 -     }
108 - #else /* !BUILDFIXED */
109 - #   include "inffixed.h"
110 - #endif /* BUILDFIXED */
```

```
111        -     state->lencode = lenfix;
112        -     state->lenbits = 9;
113        -     state->distcode = distfix;
114        -     state->distbits = 5;
115        - }
116        -
117   66     /* Macros for inflateBack(): */
118   67
119   68     /* Load returned state from inflate_fast() */
           @@ -293,7 +242,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
           FAR *in_desc,
293   242                    state->mode = STORED;
294   243                    break;
295   244                case 1:                                /* fixed block */
296        -                  fixedtables(state);
       245 +                  inflate_fixed(state);
297   246                    Tracev((stderr, "inflate:     fixed codes block%s\n",
298   247                            state->last ? " (last)" : ""));
299   248                    state->mode = LEN;              /* decode codes */
           @@ -303,8 +252,8 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
           FAR *in_desc,
303   252                            state->last ? " (last)" : ""));
304   253                    state->mode = TABLE;
305   254                    break;
306        -              case 3:
307        -                  strm->msg = (char *)"invalid block type";
       255 +              default:
       256 +                  strm->msg = (z_const char *)"invalid block type";
308   257                    state->mode = BAD;
309   258                }
310   259                DROPBITS(2);
           @@ -315,7 +264,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
           FAR *in_desc,
315   264                BYTEBITS();                         /* go to byte boundary */
316   265                NEEDBITS(32);
317   266                if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
318        -                  strm->msg = (char *)"invalid stored block lengths";
       267 +                  strm->msg = (z_const char *)"invalid stored block lengths";
319   268                    state->mode = BAD;
320   269                    break;
```

```
321  270                }
     @@ -353,7 +302,8 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
     FAR *in_desc,
353  302                DROPBITS(4);
354  303    #ifndef PKZIP_BUG_WORKAROUND
355  304                if (state->nlen > 286 || state->ndist > 30) {
356       -                strm->msg = (char *)"too many length or distance symbols";
     305    +                strm->msg = (z_const char *)
     306    +                    "too many length or distance symbols";
357  307                    state->mode = BAD;
358  308                    break;
359  309                }
     @@ -375,7 +325,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
     FAR *in_desc,
375  325                ret = inflate_table(CODES, state->lens, 19, &(state->next),
376  326                                    &(state->lenbits), state->work);
377  327                if (ret) {
378       -                strm->msg = (char *)"invalid code lengths set";
     328    +                strm->msg = (z_const char *)"invalid code lengths set";
379  329                    state->mode = BAD;
380  330                    break;
381  331                }
     @@ -398,7 +348,8 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
     FAR *in_desc,
398  348                            NEEDBITS(here.bits + 2);
399  349                            DROPBITS(here.bits);
400  350                            if (state->have == 0) {
401       -                        strm->msg = (char *)"invalid bit length repeat";
     351    +                        strm->msg = (z_const char *)
     352    +                            "invalid bit length repeat";
402  353                            state->mode = BAD;
403  354                            break;
404  355                        }
     @@ -421,7 +372,8 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
     FAR *in_desc,
421  372                        DROPBITS(7);
422  373                    }
423  374                    if (state->have + copy > state->nlen + state->ndist) {
424       -                    strm->msg = (char *)"invalid bit length repeat";
     375    +                    strm->msg = (z_const char *)
```

```
      376 +                              "invalid bit length repeat";
425   377                             state->mode = BAD;
426   378                             break;
427   379                         }
    @@ -435,7 +387,8 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
    FAR *in_desc,
435   387
436   388               /* check for end-of-block code (better have one) */
437   389               if (state->lens[256] == 0) {
438       -                 strm->msg = (char *)"invalid code -- missing end-of-block";
      390 +                 strm->msg = (z_const char *)
      391 +                     "invalid code -- missing end-of-block";
439   392                   state->mode = BAD;
440   393                   break;
441   394               }
    @@ -449,7 +402,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
    FAR *in_desc,
449   402               ret = inflate_table(LENS, state->lens, state->nlen, &(state->next),
450   403                                   &(state->lenbits), state->work);
451   404               if (ret) {
452       -                 strm->msg = (char *)"invalid literal/lengths set";
      405 +                 strm->msg = (z_const char *)"invalid literal/lengths set";
453   406                   state->mode = BAD;
454   407                   break;
455   408               }
    @@ -458,7 +411,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
    FAR *in_desc,
458   411               ret = inflate_table(DISTS, state->lens + state->nlen, state->ndist,
459   412                                   &(state->next), &(state->distbits), state->work);
460   413               if (ret) {
461       -                 strm->msg = (char *)"invalid distances set";
      414 +                 strm->msg = (z_const char *)"invalid distances set";
462   415                   state->mode = BAD;
463   416                   break;
464   417               }
    @@ -517,7 +470,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
    FAR *in_desc,
517   470
518   471               /* invalid code */
519   472               if (here.op & 64) {
```

```
520      -                  strm->msg = (char *)"invalid literal/length code";
         473 +              strm->msg = (z_const char *)"invalid literal/length code";
521      474                state->mode = BAD;
522      475                break;
523      476            }
                    @@ -549,7 +502,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
                    FAR *in_desc,
549      502            }
550      503            DROPBITS(here.bits);
551      504            if (here.op & 64) {
552      -                  strm->msg = (char *)"invalid distance code";
         505 +              strm->msg = (z_const char *)"invalid distance code";
553      506                state->mode = BAD;
554      507                break;
555      508            }
                    @@ -564,7 +517,7 @@ int ZEXPORT inflateBack(z_streamp strm, in_func in, void
                    FAR *in_desc,
564      517            }
565      518            if (state->offset > state->wsize - (state->whave < state->wsize ?
566      519                                                left : 0)) {
567      -                  strm->msg = (char *)"invalid distance too far back";
         520 +              strm->msg = (z_const char *)"invalid distance too far back";
568      521                state->mode = BAD;
569      522                break;
570      523            }
```

cpan/Compress-Raw-Zlib/zlib-src/inffast.c

```
         ...   @@ -1,5 +1,5 @@
1        1     /* inffast.c -- fast decoding
2        -      * Copyright (C) 1995-2017 Mark Adler
         2 +    * Copyright (C) 1995-2026 Mark Adler
3        3      * For conditions of distribution and use, see copyright notice in zlib.h
4        4      */
5        5
                    @@ -155,7 +155,8 @@ void ZLIB_INTERNAL inflate_fast(z_streamp strm, unsigned
                    start) {
155      155                dist += (unsigned)hold & ((1U << op) - 1);
156      156    #ifdef INFLATE_STRICT
157      157                if (dist > dmax) {
```

```
158     -                    strm->msg = (char *)"invalid distance too far back";
        158 +                strm->msg = (z_const char *)
        159 +                    "invalid distance too far back";
159 160                      state->mode = BAD;
160 161                      break;
161 162                  }
        @@ -168,8 +169,8 @@ void ZLIB_INTERNAL inflate_fast(z_streamp strm, unsigned start) {
168 169                  op = dist - op;             /* distance back in window */
169 170                  if (op > whave) {
170 171                      if (state->sane) {
171     -                        strm->msg =
172     -                            (char *)"invalid distance too far back";
        172 +                        strm->msg = (z_const char *)
        173 +                            "invalid distance too far back";
173 174                          state->mode = BAD;
174 175                          break;
175 176                      }
        @@ -265,7 +266,7 @@ void ZLIB_INTERNAL inflate_fast(z_streamp strm, unsigned start) {
265 266                  goto dodist;
266 267              }
267 268              else {
268     -                strm->msg = (char *)"invalid distance code";
        269 +                strm->msg = (z_const char *)"invalid distance code";
269 270                  state->mode = BAD;
270 271                  break;
271 272              }
        @@ -280,7 +281,7 @@ void ZLIB_INTERNAL inflate_fast(z_streamp strm, unsigned start) {
280 281              break;
281 282          }
282 283          else {
283     -            strm->msg = (char *)"invalid literal/length code";
        284 +            strm->msg = (z_const char *)"invalid literal/length code";
284 285              state->mode = BAD;
285 286              break;
286 287          }
```

cpan/Compress-Raw-Zlib/zlib-src/inffixed.h

```
@@ -1,94 +1,94 @@
-  /* inffixed.h -- table for decoding fixed codes
-   * Generated automatically by makefixed().
-   */
+/* inffixed.h -- table for decoding fixed codes
+ * Generated automatically by makefixed().
+ */

-  /* WARNING: this file should *not* be used by applications.
-     It is part of the implementation of this library and is
-     subject to change. Applications should only use zlib.h.
-   */
+/* WARNING: this file should *not* be used by applications.
+   It is part of the implementation of this library and is
+   subject to change. Applications should only use zlib.h.
+ */

-  static const code lenfix[512] = {
-      {96,7,0},{0,8,80},{0,8,16},{20,8,115},{18,7,31},{0,8,112},{0,8,48},
-      {0,9,192},{16,7,10},{0,8,96},{0,8,32},{0,9,160},{0,8,0},{0,8,128},
-      {0,8,64},{0,9,224},{16,7,6},{0,8,88},{0,8,24},{0,9,144},{19,7,59},
-      {0,8,120},{0,8,56},{0,9,208},{17,7,17},{0,8,104},{0,8,40},{0,9,176},
-      {0,8,8},{0,8,136},{0,8,72},{0,9,240},{16,7,4},{0,8,84},{0,8,20},
-      {21,8,227},{19,7,43},{0,8,116},{0,8,52},{0,9,200},{17,7,13},{0,8,100},
-      {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},{16,7,8},
-      {0,8,92},{0,8,28},{0,9,152},{20,7,83},{0,8,124},{0,8,60},{0,9,216},
-      {18,7,23},{0,8,108},{0,8,44},{0,9,184},{0,8,12},{0,8,140},{0,8,76},
-      {0,9,248},{16,7,3},{0,8,82},{0,8,18},{21,8,163},{19,7,35},{0,8,114},
-      {0,8,50},{0,9,196},{17,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},
-      {0,8,130},{0,8,66},{0,9,228},{16,7,7},{0,8,90},{0,8,26},{0,9,148},
-      {20,7,67},{0,8,122},{0,8,58},{0,9,212},{18,7,19},{0,8,106},{0,8,42},
-      {0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},{16,7,5},{0,8,86},
-      {0,8,22},{64,8,0},{19,7,51},{0,8,118},{0,8,54},{0,9,204},{17,7,15},
-      {0,8,102},{0,8,38},{0,9,172},{0,8,6},{0,8,134},{0,8,70},{0,9,236},
-      {16,7,9},{0,8,94},{0,8,30},{0,9,156},{20,7,99},{0,8,126},{0,8,62},
-      {0,9,220},{18,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
-      {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{21,8,131},{18,7,31},
-      {0,8,113},{0,8,49},{0,9,194},{16,7,10},{0,8,97},{0,8,33},{0,9,162},
```

```
31  -       {0,8,1},{0,8,129},{0,8,65},{0,9,226},{16,7,6},{0,8,89},{0,8,25},
32  -       {0,9,146},{19,7,59},{0,8,121},{0,8,57},{0,9,210},{17,7,17},{0,8,105},
33  -       {0,8,41},{0,9,178},{0,8,9},{0,8,137},{0,8,73},{0,9,242},{16,7,4},
34  -       {0,8,85},{0,8,21},{16,8,258},{19,7,43},{0,8,117},{0,8,53},{0,9,202},
35  -       {17,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},{0,8,69},
36  -       {0,9,234},{16,7,8},{0,8,93},{0,8,29},{0,9,154},{20,7,83},{0,8,125},
37  -       {0,8,61},{0,9,218},{18,7,23},{0,8,109},{0,8,45},{0,9,186},{0,8,13},
38  -       {0,8,141},{0,8,77},{0,9,250},{16,7,3},{0,8,83},{0,8,19},{21,8,195},
39  -       {19,7,35},{0,8,115},{0,8,51},{0,9,198},{17,7,11},{0,8,99},{0,8,35},
40  -       {0,9,166},{0,8,3},{0,8,131},{0,8,67},{0,9,230},{16,7,7},{0,8,91},
41  -       {0,8,27},{0,9,150},{20,7,67},{0,8,123},{0,8,59},{0,9,214},{18,7,19},
42  -       {0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},{0,8,75},{0,9,246},
43  -       {16,7,5},{0,8,87},{0,8,23},{64,8,0},{19,7,51},{0,8,119},{0,8,55},
44  -       {0,9,206},{17,7,15},{0,8,103},{0,8,39},{0,9,174},{0,8,7},{0,8,135},
45  -       {0,8,71},{0,9,238},{16,7,9},{0,8,95},{0,8,31},{0,9,158},{20,7,99},
46  -       {0,8,127},{0,8,63},{0,9,222},{18,7,27},{0,8,111},{0,8,47},{0,9,190},
47  -       {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},{0,8,16},
48  -       {20,8,115},{18,7,31},{0,8,112},{0,8,48},{0,9,193},{16,7,10},{0,8,96},
49  -       {0,8,32},{0,9,161},{0,8,0},{0,8,128},{0,8,64},{0,9,225},{16,7,6},
50  -       {0,8,88},{0,8,24},{0,9,145},{19,7,59},{0,8,120},{0,8,56},{0,9,209},
51  -       {17,7,17},{0,8,104},{0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},
52  -       {0,9,241},{16,7,4},{0,8,84},{0,8,20},{21,8,227},{19,7,43},{0,8,116},
53  -       {0,8,52},{0,9,201},{17,7,13},{0,8,100},{0,8,36},{0,9,169},{0,8,4},
54  -       {0,8,132},{0,8,68},{0,9,233},{16,7,8},{0,8,92},{0,8,28},{0,9,153},
55  -       {20,7,83},{0,8,124},{0,8,60},{0,9,217},{18,7,23},{0,8,108},{0,8,44},
56  -       {0,9,185},{0,8,12},{0,8,140},{0,8,76},{0,9,249},{16,7,3},{0,8,82},
57  -       {0,8,18},{21,8,163},{19,7,35},{0,8,114},{0,8,50},{0,9,197},{17,7,11},
58  -       {0,8,98},{0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
59  -       {16,7,7},{0,8,90},{0,8,26},{0,9,149},{20,7,67},{0,8,122},{0,8,58},
60  -       {0,9,213},{18,7,19},{0,8,106},{0,8,42},{0,9,181},{0,8,10},{0,8,138},
61  -       {0,8,74},{0,9,245},{16,7,5},{0,8,86},{0,8,22},{64,8,0},{19,7,51},
62  -       {0,8,118},{0,8,54},{0,9,205},{17,7,15},{0,8,102},{0,8,38},{0,9,173},
63  -       {0,8,6},{0,8,134},{0,8,70},{0,9,237},{16,7,9},{0,8,94},{0,8,30},
64  -       {0,9,157},{20,7,99},{0,8,126},{0,8,62},{0,9,221},{18,7,27},{0,8,110},
65  -       {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},{96,7,0},
66  -       {0,8,81},{0,8,17},{21,8,131},{18,7,31},{0,8,113},{0,8,49},{0,9,195},
67  -       {16,7,10},{0,8,97},{0,8,33},{0,9,163},{0,8,1},{0,8,129},{0,8,65},
68  -       {0,9,227},{16,7,6},{0,8,89},{0,8,25},{0,9,147},{19,7,59},{0,8,121},
69  -       {0,8,57},{0,9,211},{17,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},
70  -       {0,8,137},{0,8,73},{0,9,243},{16,7,4},{0,8,85},{0,8,21},{16,8,258},
```

```
-       {19,7,43},{0,8,117},{0,8,53},{0,9,203},{17,7,13},{0,8,101},{0,8,37},
-       {0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},{16,7,8},{0,8,93},
-       {0,8,29},{0,9,155},{20,7,83},{0,8,125},{0,8,61},{0,9,219},{18,7,23},
-       {0,8,109},{0,8,45},{0,9,187},{0,8,13},{0,8,141},{0,8,77},{0,9,251},
-       {16,7,3},{0,8,83},{0,8,19},{21,8,195},{19,7,35},{0,8,115},{0,8,51},
-       {0,9,199},{17,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
-       {0,8,67},{0,9,231},{16,7,7},{0,8,91},{0,8,27},{0,9,151},{20,7,67},
-       {0,8,123},{0,8,59},{0,9,215},{18,7,19},{0,8,107},{0,8,43},{0,9,183},
-       {0,8,11},{0,8,139},{0,8,75},{0,9,247},{16,7,5},{0,8,87},{0,8,23},
-       {64,8,0},{19,7,51},{0,8,119},{0,8,55},{0,9,207},{17,7,15},{0,8,103},
-       {0,8,39},{0,9,175},{0,8,7},{0,8,135},{0,8,71},{0,9,239},{16,7,9},
-       {0,8,95},{0,8,31},{0,9,159},{20,7,99},{0,8,127},{0,8,63},{0,9,223},
-       {18,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},{0,8,79},
-       {0,9,255}
-   };
+ static const code lenfix[512] = {
+     {96,7,0},{0,8,80},{0,8,16},{20,8,115},{18,7,31},{0,8,112},{0,8,48},
+     {0,9,192},{16,7,10},{0,8,96},{0,8,32},{0,9,160},{0,8,0},{0,8,128},
+     {0,8,64},{0,9,224},{16,7,6},{0,8,88},{0,8,24},{0,9,144},{19,7,59},
+     {0,8,120},{0,8,56},{0,9,208},{17,7,17},{0,8,104},{0,8,40},{0,9,176},
+     {0,8,8},{0,8,136},{0,8,72},{0,9,240},{16,7,4},{0,8,84},{0,8,20},
+     {21,8,227},{19,7,43},{0,8,116},{0,8,52},{0,9,200},{17,7,13},{0,8,100},
+     {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},{16,7,8},
+     {0,8,92},{0,8,28},{0,9,152},{20,7,83},{0,8,124},{0,8,60},{0,9,216},
+     {18,7,23},{0,8,108},{0,8,44},{0,9,184},{0,8,12},{0,8,140},{0,8,76},
+     {0,9,248},{16,7,3},{0,8,82},{0,8,18},{21,8,163},{19,7,35},{0,8,114},
+     {0,8,50},{0,9,196},{17,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},
+     {0,8,130},{0,8,66},{0,9,228},{16,7,7},{0,8,90},{0,8,26},{0,9,148},
+     {20,7,67},{0,8,122},{0,8,58},{0,9,212},{18,7,19},{0,8,106},{0,8,42},
+     {0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},{16,7,5},{0,8,86},
+     {0,8,22},{64,8,0},{19,7,51},{0,8,118},{0,8,54},{0,9,204},{17,7,15},
+     {0,8,102},{0,8,38},{0,9,172},{0,8,6},{0,8,134},{0,8,70},{0,9,236},
+     {16,7,9},{0,8,94},{0,8,30},{0,9,156},{20,7,99},{0,8,126},{0,8,62},
+     {0,9,220},{18,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
+     {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{21,8,131},{18,7,31},
+     {0,8,113},{0,8,49},{0,9,194},{16,7,10},{0,8,97},{0,8,33},{0,9,162},
+     {0,8,1},{0,8,129},{0,8,65},{0,9,226},{16,7,6},{0,8,89},{0,8,25},
+     {0,9,146},{19,7,59},{0,8,121},{0,8,57},{0,9,210},{17,7,17},{0,8,105},
+     {0,8,41},{0,9,178},{0,8,9},{0,8,137},{0,8,73},{0,9,242},{16,7,4},
+     {0,8,85},{0,8,21},{16,8,258},{19,7,43},{0,8,117},{0,8,53},{0,9,202},
```

```
35 +     {17,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},{0,8,69},
36 +     {0,9,234},{16,7,8},{0,8,93},{0,8,29},{0,9,154},{20,7,83},{0,8,125},
37 +     {0,8,61},{0,9,218},{18,7,23},{0,8,109},{0,8,45},{0,9,186},{0,8,13},
38 +     {0,8,141},{0,8,77},{0,9,250},{16,7,3},{0,8,83},{0,8,19},{21,8,195},
39 +     {19,7,35},{0,8,115},{0,8,51},{0,9,198},{17,7,11},{0,8,99},{0,8,35},
40 +     {0,9,166},{0,8,3},{0,8,131},{0,8,67},{0,9,230},{16,7,7},{0,8,91},
41 +     {0,8,27},{0,9,150},{20,7,67},{0,8,123},{0,8,59},{0,9,214},{18,7,19},
42 +     {0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},{0,8,75},{0,9,246},
43 +     {16,7,5},{0,8,87},{0,8,23},{64,8,0},{19,7,51},{0,8,119},{0,8,55},
44 +     {0,9,206},{17,7,15},{0,8,103},{0,8,39},{0,9,174},{0,8,7},{0,8,135},
45 +     {0,8,71},{0,9,238},{16,7,9},{0,8,95},{0,8,31},{0,9,158},{20,7,99},
46 +     {0,8,127},{0,8,63},{0,9,222},{18,7,27},{0,8,111},{0,8,47},{0,9,190},
47 +     {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},{0,8,16},
48 +     {20,8,115},{18,7,31},{0,8,112},{0,8,48},{0,9,193},{16,7,10},{0,8,96},
49 +     {0,8,32},{0,9,161},{0,8,0},{0,8,128},{0,8,64},{0,9,225},{16,7,6},
50 +     {0,8,88},{0,8,24},{0,9,145},{19,7,59},{0,8,120},{0,8,56},{0,9,209},
51 +     {17,7,17},{0,8,104},{0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},
52 +     {0,9,241},{16,7,4},{0,8,84},{0,8,20},{21,8,227},{19,7,43},{0,8,116},
53 +     {0,8,52},{0,9,201},{17,7,13},{0,8,100},{0,8,36},{0,9,169},{0,8,4},
54 +     {0,8,132},{0,8,68},{0,9,233},{16,7,8},{0,8,92},{0,8,28},{0,9,153},
55 +     {20,7,83},{0,8,124},{0,8,60},{0,9,217},{18,7,23},{0,8,108},{0,8,44},
56 +     {0,9,185},{0,8,12},{0,8,140},{0,8,76},{0,9,249},{16,7,3},{0,8,82},
57 +     {0,8,18},{21,8,163},{19,7,35},{0,8,114},{0,8,50},{0,9,197},{17,7,11},
58 +     {0,8,98},{0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
59 +     {16,7,7},{0,8,90},{0,8,26},{0,9,149},{20,7,67},{0,8,122},{0,8,58},
60 +     {0,9,213},{18,7,19},{0,8,106},{0,8,42},{0,9,181},{0,8,10},{0,8,138},
61 +     {0,8,74},{0,9,245},{16,7,5},{0,8,86},{0,8,22},{64,8,0},{19,7,51},
62 +     {0,8,118},{0,8,54},{0,9,205},{17,7,15},{0,8,102},{0,8,38},{0,9,173},
63 +     {0,8,6},{0,8,134},{0,8,70},{0,9,237},{16,7,9},{0,8,94},{0,8,30},
64 +     {0,9,157},{20,7,99},{0,8,126},{0,8,62},{0,9,221},{18,7,27},{0,8,110},
65 +     {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},{96,7,0},
66 +     {0,8,81},{0,8,17},{21,8,131},{18,7,31},{0,8,113},{0,8,49},{0,9,195},
67 +     {16,7,10},{0,8,97},{0,8,33},{0,9,163},{0,8,1},{0,8,129},{0,8,65},
68 +     {0,9,227},{16,7,6},{0,8,89},{0,8,25},{0,9,147},{19,7,59},{0,8,121},
69 +     {0,8,57},{0,9,211},{17,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},
70 +     {0,8,137},{0,8,73},{0,9,243},{16,7,4},{0,8,85},{0,8,21},{16,8,258},
71 +     {19,7,43},{0,8,117},{0,8,53},{0,9,203},{17,7,13},{0,8,101},{0,8,37},
72 +     {0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},{16,7,8},{0,8,93},
73 +     {0,8,29},{0,9,155},{20,7,83},{0,8,125},{0,8,61},{0,9,219},{18,7,23},
74 +     {0,8,109},{0,8,45},{0,9,187},{0,8,13},{0,8,141},{0,8,77},{0,9,251},
```

```
      75 +     {16,7,3},{0,8,83},{0,8,19},{21,8,195},{19,7,35},{0,8,115},{0,8,51},
      76 +     {0,9,199},{17,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
      77 +     {0,8,67},{0,9,231},{16,7,7},{0,8,91},{0,8,27},{0,9,151},{20,7,67},
      78 +     {0,8,123},{0,8,59},{0,9,215},{18,7,19},{0,8,107},{0,8,43},{0,9,183},
      79 +     {0,8,11},{0,8,139},{0,8,75},{0,9,247},{16,7,5},{0,8,87},{0,8,23},
      80 +     {64,8,0},{19,7,51},{0,8,119},{0,8,55},{0,9,207},{17,7,15},{0,8,103},
      81 +     {0,8,39},{0,9,175},{0,8,7},{0,8,135},{0,8,71},{0,9,239},{16,7,9},
      82 +     {0,8,95},{0,8,31},{0,9,159},{20,7,99},{0,8,127},{0,8,63},{0,9,223},
      83 +     {18,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},{0,8,79},
      84 +     {0,9,255}
      85 + };
86    86
87        - static const code distfix[32] = {
88        -     {16,5,1},{23,5,257},{19,5,17},{27,5,4097},{17,5,5},{25,5,1025},
89        -     {21,5,65},{29,5,16385},{16,5,3},{24,5,513},{20,5,33},{28,5,8193},
90        -     {18,5,9},{26,5,2049},{22,5,129},{64,5,0},{16,5,2},{23,5,385},
91        -     {19,5,25},{27,5,6145},{17,5,7},{25,5,1537},{21,5,97},{29,5,24577},
92        -     {16,5,4},{24,5,769},{20,5,49},{28,5,12289},{18,5,13},{26,5,3073},
93        -     {22,5,193},{64,5,0}
94        - };
      87 + static const code distfix[32] = {
      88 +     {16,5,1},{23,5,257},{19,5,17},{27,5,4097},{17,5,5},{25,5,1025},
      89 +     {21,5,65},{29,5,16385},{16,5,3},{24,5,513},{20,5,33},{28,5,8193},
      90 +     {18,5,9},{26,5,2049},{22,5,129},{64,5,0},{16,5,2},{23,5,385},
      91 +     {19,5,25},{27,5,6145},{17,5,7},{25,5,1537},{21,5,97},{29,5,24577},
      92 +     {16,5,4},{24,5,769},{20,5,49},{28,5,12289},{18,5,13},{26,5,3073},
      93 +     {22,5,193},{64,5,0}
      94 + };
```

cpan/Compress-Raw-Zlib/zlib-src/inflate.c

Load Diff

Large diffs are not rendered by default.

cpan/Compress-Raw-Zlib/zlib-src/inflate.h

```
@@ -100,7 +100,7 @@ struct inflate_state {
100   100         unsigned char FAR *window;  /* allocated sliding window, if needed */
```

```
101  101          /* bit accumulator */
102  102      unsigned long hold;         /* input bit accumulator */
103       -  unsigned bits;               /* number of bits in "in" */
     103  +  unsigned bits;               /* number of bits in hold */
104  104          /* for string and stored block copying */
105  105      unsigned length;            /* literal or length of data to copy */
106  106      unsigned offset;            /* distance back to copy string from */
```

cpan/Compress-Raw-Zlib/zlib-src/inftrees.c

```
     ...  @@ -1,15 +1,29 @@
1    1    /* inftrees.c -- generate Huffman trees for efficient decoding
2         - * Copyright (C) 1995-2024 Mark Adler
     2    + * Copyright (C) 1995-2026 Mark Adler
3    3     * For conditions of distribution and use, see copyright notice in zlib.h
4    4     */
5    5
     6    + #ifdef MAKEFIXED
     7    + #  ifndef BUILDFIXED
     8    + #    define BUILDFIXED
     9    + #  endif
     10   + #endif
     11   + #ifdef BUILDFIXED
     12   + #  define Z_ONCE
     13   + #endif
     14   +
6    15   #include "zutil.h"
7    16   #include "inftrees.h"
     17   + #include "inflate.h"
     18   +
     19   + #ifndef NULL
     20   + #  define NULL 0
     21   + #endif
8    22
9    23   #define MAXBITS 15
10   24
11   25   const char inflate_copyright[] =
12        -    " inflate 1.3.1 Copyright 1995-2024 Mark Adler ";
     26   +    " inflate 1.3.2 Copyright 1995-2026 Mark Adler ";
13   27   /*
```

```
14  28        If you use the zlib library in a product, an acknowledgment is welcome
15  29        in the documentation of your product. If for some reason you cannot
        @@ -47,17 +61,17 @@ int ZLIB_INTERNAL inflate_table(codetype type, unsigned
        short FAR *lens,
47  61            unsigned mask;              /* mask for low root bits */
48  62            code here;                  /* table entry for duplication */
49  63            code FAR *next;             /* next available space in table */
50      -         const unsigned short FAR *base;     /* base value table to use */
51      -         const unsigned short FAR *extra;    /* extra bits table to use */
52      -         unsigned match;             /* use base and extra for symbol >= match */
    64  +         const unsigned short FAR *base = NULL;  /* base value table to use */
    65  +         const unsigned short FAR *extra = NULL; /* extra bits table to use */
    66  +         unsigned match = 0;         /* use base and extra for symbol >= match */
53  67            unsigned short count[MAXBITS+1];    /* number of codes of each length */
54  68            unsigned short offs[MAXBITS+1];     /* offsets in table for each length */
55  69            static const unsigned short lbase[31] = { /* Length codes 257..285 base */
56  70                3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
57  71                35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258, 0, 0};
58  72            static const unsigned short lext[31] = { /* Length codes 257..285 extra */
59  73                16, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 18, 18, 18, 18,
60      -             19, 19, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 16, 203, 77};
    74  +             19, 19, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 16, 199, 75};
61  75            static const unsigned short dbase[32] = { /* Distance codes 0..29 base */
62  76                1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
63  77                257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        @@ -175,18 +189,16 @@ int ZLIB_INTERNAL inflate_table(codetype type,
        unsigned short FAR *lens,
175 189           /* set up for code type */
176 190           switch (type) {
177 191           case CODES:
178     -             base = extra = work;    /* dummy value--not used */
179 192               match = 20;
180 193               break;
181 194           case LENS:
182 195               base = lbase;
183 196               extra = lext;
184 197               match = 257;
185 198               break;
186     -         default:    /* DISTS */
    199 +         case DISTS:
```

```
187  200            base = dbase;
188  201            extra = dext;
189       -          match = 0;
190  202        }
191  203
192  204        /* initialize state for loop */
          @@ -297,3 +309,116 @@ int ZLIB_INTERNAL inflate_table(codetype type,
          unsigned short FAR *lens,
297  309        *bits = root;
298  310        return 0;
299  311    }
     312  +
     313  + #ifdef BUILDFIXED
     314  + /*
     315  +   If this is compiled with BUILDFIXED defined, and if inflate will be used in
     316  +   multiple threads, and if atomics are not available, then inflate() must be
     317  +   called with a fixed block (e.g. 0x03 0x00) to initialize the tables and must
     318  +   return before any other threads are allowed to call inflate.
     319  +  */
     320  +
     321  + static code *lenfix, *distfix;
     322  + static code fixed[544];
     323  +
     324  + /* State for z_once(). */
     325  + local z_once_t built = Z_ONCE_INIT;
     326  +
     327  + local void buildtables(void) {
     328  +     unsigned sym, bits;
     329  +     static code *next;
     330  +     unsigned short lens[288], work[288];
     331  +
     332  +     /* literal/length table */
     333  +     sym = 0;
     334  +     while (sym < 144) lens[sym++] = 8;
     335  +     while (sym < 256) lens[sym++] = 9;
     336  +     while (sym < 280) lens[sym++] = 7;
     337  +     while (sym < 288) lens[sym++] = 8;
     338  +     next = fixed;
     339  +     lenfix = next;
     340  +     bits = 9;
```

```
341 +     inflate_table(LENS, lens, 288, &(next), &(bits), work);
342 +
343 +     /* distance table */
344 +     sym = 0;
345 +     while (sym < 32) lens[sym++] = 5;
346 +     distfix = next;
347 +     bits = 5;
348 +     inflate_table(DISTS, lens, 32, &(next), &(bits), work);
349 + }
350 + #else /* !BUILDFIXED */
351 + #  include "inffixed.h"
352 + #endif /* BUILDFIXED */
353 +
354 + /*
355 +    Return state with length and distance decoding tables and index sizes set to
356 +    fixed code decoding.  Normally this returns fixed tables from inffixed.h.
357 +    If BUILDFIXED is defined, then instead this routine builds the tables the
358 +    first time it's called, and returns those tables the first time and
359 +    thereafter.  This reduces the size of the code by about 2K bytes, in
360 +    exchange for a little execution time.  However, BUILDFIXED should not be
361 +    used for threaded applications if atomics are not available, as it will
362 +    not be thread-safe.
363 +  */
364 + void inflate_fixed(struct inflate_state FAR *state) {
365 + #ifdef BUILDFIXED
366 +     z_once(&built, buildtables);
367 + #endif /* BUILDFIXED */
368 +     state->lencode = lenfix;
369 +     state->lenbits = 9;
370 +     state->distcode = distfix;
371 +     state->distbits = 5;
372 + }
373 +
374 + #ifdef MAKEFIXED
375 + #include <stdio.h>
376 +
377 + /*
378 +    Write out the inffixed.h that will be #include'd above.  Defining MAKEFIXED
379 +    also defines BUILDFIXED, so the tables are built on the fly.  main() writes
380 +    those tables to stdout, which would directed to inffixed.h. Compile this
```

```
381 +    along with zutil.c:
382 +
383 +        cc -DMAKEFIXED -o fix inftrees.c zutil.c
384 +        ./fix > inffixed.h
385 +  */
386 + int main(void) {
387 +     unsigned low, size;
388 +     struct inflate_state state;
389 +
390 +     inflate_fixed(&state);
391 +     puts("/* inffixed.h -- table for decoding fixed codes");
392 +     puts(" * Generated automatically by makefixed().");
393 +     puts(" */");
394 +     puts("");
395 +     puts("/* WARNING: this file should *not* be used by applications.");
396 +     puts("   It is part of the implementation of this library and is");
397 +     puts("   subject to change. Applications should only use zlib.h.");
398 +     puts(" */");
399 +     puts("");
400 +     size = 1U << 9;
401 +     printf("static const code lenfix[%u] = {", size);
402 +     low = 0;
403 +     for (;;) {
404 +         if ((low % 7) == 0) printf("\n    ");
405 +         printf("{%u,%u,%d}", (low & 127) == 99 ? 64 : state.lencode[low].op,
406 +                state.lencode[low].bits, state.lencode[low].val);
407 +         if (++low == size) break;
408 +         putchar(',');
409 +     }
410 +     puts("\n};");
411 +     size = 1U << 5;
412 +     printf("\nstatic const code distfix[%u] = {", size);
413 +     low = 0;
414 +     for (;;) {
415 +         if ((low % 6) == 0) printf("\n    ");
416 +         printf("{%u,%u,%d}", state.distcode[low].op, state.distcode[low].bits,
417 +                state.distcode[low].val);
418 +         if (++low == size) break;
419 +         putchar(',');
420 +     }
```

```
421 +     puts("\n};");
422 +     return 0;
423 + }
424 + #endif /* MAKEFIXED */
```

cpan/Compress-Raw-Zlib/zlib-src/inftrees.h

```
    @@ -1,5 +1,5 @@
1   1   /* inftrees.h -- header to use inftrees.c
2       - * Copyright (C) 1995-2005, 2010 Mark Adler
    2   + * Copyright (C) 1995-2026 Mark Adler
3   3    * For conditions of distribution and use, see copyright notice in zlib.h
4   4    */
5   5
    @@ -60,3 +60,5 @@ typedef enum {
60  60  int ZLIB_INTERNAL inflate_table(codetype type, unsigned short FAR *lens,
61  61                                  unsigned codes, code FAR * FAR *table,
62  62                                  unsigned FAR *bits, unsigned short FAR *work);
    63  + struct inflate_state;
    64  + void ZLIB_INTERNAL inflate_fixed(struct inflate_state FAR *state);
```

cpan/Compress-Raw-Zlib/zlib-src/trees.c

```
    @@ -1,5 +1,5 @@
1   1   /* trees.c -- output deflated data using Huffman coding
2       - * Copyright (C) 1995-2024 Jean-loup Gailly
    2   + * Copyright (C) 1995-2026 Jean-loup Gailly
3   3    * detect_data_type() function provided freely by Cosmin Truta, 2006
4   4    * For conditions of distribution and use, see copyright notice in zlib.h
5   5    */
    @@ -112,7 +112,7 @@ local int base_dist[D_CODES];
112 112
113 113   #else
114 114   #  include "trees.h"
115     - #endif /* GEN_TREES_H */
    115 + #endif /* defined(GEN_TREES_H) || !defined(STDC) */
116 116
117 117   struct static_tree_desc_s {
118 118       const ct_data *static_tree;  /* static tree or NULL */
```

```
@@ -152,7 +152,7 @@ local TCONST static_tree_desc static_bl_desc =
152 152      * IN assertion: 1 <= len <= 15
153 153      */
154 154     local unsigned bi_reverse(unsigned code, int len) {
155     -     register unsigned res = 0;
    155 +     unsigned res = 0;
156 156         do {
157 157             res |= code & 1;
158 158             code >>= 1, res <<= 1;
@@ -184,10 +184,11 @@ local void bi_windup(deflate_state *s) {
184 184         } else if (s->bi_valid > 0) {
185 185             put_byte(s, (Byte)s->bi_buf);
186 186         }
    187 +     s->bi_used = ((s->bi_valid - 1) & 7) + 1;
187 188         s->bi_buf = 0;
188 189         s->bi_valid = 0;
189 190     #ifdef ZLIB_DEBUG
190     -     s->bits_sent = (s->bits_sent + 7) & ~7;
    191 +     s->bits_sent = (s->bits_sent + 7) & ~(ulg)7;
191 192     #endif
192 193     }
193 194
@@ -466,6 +467,7 @@ void ZLIB_INTERNAL _tr_init(deflate_state *s) {
466 467
467 468         s->bi_buf = 0;
468 469         s->bi_valid = 0;
    470 +     s->bi_used = 0;
469 471     #ifdef ZLIB_DEBUG
470 472         s->compressed_len = 0L;
471 473         s->bits_sent = 0L;
@@ -724,7 +726,7 @@ local void scan_tree(deflate_state *s, ct_data *tree,
    int max_code) {
724 726             if (++count < max_count && curlen == nextlen) {
725 727                 continue;
726 728             } else if (count < min_count) {
727     -               s->bl_tree[curlen].Freq += count;
    729 +               s->bl_tree[curlen].Freq += (ush)count;
```

```
728  730            } else if (curlen != 0) {
729  731                if (curlen != prevlen) s->bl_tree[curlen].Freq++;
730  732                s->bl_tree[REP_3_6].Freq++;
     @@ -817,7 +819,7 @@ local int build_bl_tree(deflate_state *s) {
817  819        }
818  820        /* Update opt_len to include the bit length tree and counts */
819  821        s->opt_len += 3*((ulg)max_blindex + 1) + 5 + 5 + 4;
820       -    Tracev((stderr, "\ndyn trees: dyn %ld, stat %ld",
     822  +    Tracev((stderr, "\ndyn trees: dyn %lu, stat %lu",
821  823                s->opt_len, s->static_len));
822  824
823  825        return max_blindex;
     @@ -843,13 +845,13 @@ local void send_all_trees(deflate_state *s, int
     lcodes, int dcodes,
843  845            Tracev((stderr, "\nbl code %2d ", bl_order[rank]));
844  846            send_bits(s, s->bl_tree[bl_order[rank]].Len, 3);
845  847        }
846       -    Tracev((stderr, "\nbl tree: sent %ld", s->bits_sent));
     848  +    Tracev((stderr, "\nbl tree: sent %lu", s->bits_sent));
847  849
848  850        send_tree(s, (ct_data *)s->dyn_ltree, lcodes - 1);  /* literal tree */
849       -    Tracev((stderr, "\nlit tree: sent %ld", s->bits_sent));
     851  +    Tracev((stderr, "\nlit tree: sent %lu", s->bits_sent));
850  852
851  853        send_tree(s, (ct_data *)s->dyn_dtree, dcodes - 1);  /* distance tree */
852       -    Tracev((stderr, "\ndist tree: sent %ld", s->bits_sent));
     854  +    Tracev((stderr, "\ndist tree: sent %lu", s->bits_sent));
853  855    }
854  856
855  857    /*
       ================================================================================
     @@ -932,7 +934,7 @@ local void compress_block(deflate_state *s, const
     ct_data *ltree,
932  934                extra = extra_dbits[code];
933  935                if (extra != 0) {
934  936                    dist -= (unsigned)base_dist[code];
935       -                send_bits(s, dist, extra);   /* send the extra distance bits
     */
     937  +                send_bits(s, (int)dist, extra); /* send the extra bits */
```

```
936  938                }
937  939              } /* literal or match pair ? */
938  940
     @@ -1006,11 +1008,11 @@ void ZLIB_INTERNAL _tr_flush_block(deflate_state
     *s, charf *buf,
1006 1008
1007 1009            /* Construct the literal and distance trees */
1008 1010            build_tree(s, (tree_desc *)(&(s->l_desc)));
1009      -          Tracev((stderr, "\nlit data: dyn %ld, stat %ld", s->opt_len,
     1011 +          Tracev((stderr, "\nlit data: dyn %lu, stat %lu", s->opt_len,
1010 1012                  s->static_len));
1011 1013
1012 1014            build_tree(s, (tree_desc *)(&(s->d_desc)));
1013      -          Tracev((stderr, "\ndist data: dyn %ld, stat %ld", s->opt_len,
     1015 +          Tracev((stderr, "\ndist data: dyn %lu, stat %lu", s->opt_len,
1014 1016                  s->static_len));
1015 1017            /* At this point, opt_len and static_len are the total bit lengths of
1016 1018             * the compressed block data, excluding the tree representations.
     @@ -1083,7 +1085,7 @@ void ZLIB_INTERNAL _tr_flush_block(deflate_state
     *s, charf *buf,
1083 1085    #endif
1084 1086        }
1085 1087        Tracev((stderr,"\ncomprlen %lu(%lu) ", s->compressed_len >> 3,
1086      -            s->compressed_len - 7*last));
     1088 +            s->compressed_len - 7*(ulg)last));
1087 1089    }
1088 1090
1089 1091    /*
     ================================================================================
```

cpan/Compress-Raw-Zlib/zlib-src/uncompr.c

```
     ...  @@ -1,5 +1,5 @@
1    1    /* uncompr.c -- decompress a memory buffer
2         - * Copyright (C) 1995-2003, 2010, 2014, 2016 Jean-loup Gailly, Mark Adler
     2    + * Copyright (C) 1995-2026 Jean-loup Gailly, Mark Adler
3    3     * For conditions of distribution and use, see copyright notice in zlib.h
4    4     */
5    5
     @@ -23,24 +23,24 @@
```

```
23  23        memory, Z_BUF_ERROR if there was not enough room in the output buffer, or
24  24        Z_DATA_ERROR if the input data was corrupted, including if the input data is
25  25        an incomplete zlib stream.
        26  +
        27  +      The _z versions of the functions take size_t length arguments.
26  28      */
27      -  int ZEXPORT uncompress2(Bytef *dest, uLongf *destLen, const Bytef *source,
28      -                          uLong *sourceLen) {
        29  +  int ZEXPORT uncompress2_z(Bytef *dest, z_size_t *destLen, const Bytef *source,
        30  +                            z_size_t *sourceLen) {
29  31        z_stream stream;
30  32        int err;
31  33        const uInt max = (uInt)-1;
32      -     uLong len, left;
33      -     Byte buf[1];    /* for detection of incomplete stream when *destLen == 0 */
        34  +     z_size_t len, left;
        35  +
        36  +     if (sourceLen == NULL || (*sourceLen > 0 && source == NULL) ||
        37  +         destLen == NULL || (*destLen > 0 && dest == NULL))
        38  +         return Z_STREAM_ERROR;
34  39
35  40        len = *sourceLen;
36      -     if (*destLen) {
37      -         left = *destLen;
38      -         *destLen = 0;
39      -     }
40      -     else {
41      -         left = 1;
42      -         dest = buf;
43      -     }
        41  +     left = *destLen;
        42  +     if (left == 0 && dest == Z_NULL)
        43  +         dest = (Bytef *)&stream.reserved;       /* next_out cannot be NULL */
44  44
45  45        stream.next_in = (z_const Bytef *)source;
46  46        stream.avail_in = 0;
          @@ -56,30 +56,46 @@ int ZEXPORT uncompress2(Bytef *dest, uLongf *destLen,
          const Bytef *source,
56  56
57  57        do {
```

```
58  58            if (stream.avail_out == 0) {
59      -             stream.avail_out = left > (uLong)max ? max : (uInt)left;
    59  +             stream.avail_out = left > (z_size_t)max ? max : (uInt)left;
60  60                left -= stream.avail_out;
61  61            }
62  62            if (stream.avail_in == 0) {
63      -             stream.avail_in = len > (uLong)max ? max : (uInt)len;
    63  +             stream.avail_in = len > (z_size_t)max ? max : (uInt)len;
64  64                len -= stream.avail_in;
65  65            }
66  66            err = inflate(&stream, Z_NO_FLUSH);
67  67        } while (err == Z_OK);
68  68
69      -     *sourceLen -= len + stream.avail_in;
70      -     if (dest != buf)
71      -         *destLen = stream.total_out;
72      -     else if (stream.total_out && err == Z_BUF_ERROR)
73      -         left = 1;
    69  +     /* Set len and left to the unused input data and unused output space. Set
    70  +        *sourceLen to the amount of input consumed. Set *destLen to the amount
    71  +        of data produced. */
    72  +     len += stream.avail_in;
    73  +     left += stream.avail_out;
    74  +     *sourceLen -= len;
    75  +     *destLen -= left;
74  76
75  77        inflateEnd(&stream);
76  78        return err == Z_STREAM_END ? Z_OK :
77  79               err == Z_NEED_DICT ? Z_DATA_ERROR  :
78      -            err == Z_BUF_ERROR && left + stream.avail_out ? Z_DATA_ERROR :
    80  +            err == Z_BUF_ERROR && len == 0 ? Z_DATA_ERROR :
79  81               err;
80  82    }
81      -
    83  + int ZEXPORT uncompress2(Bytef *dest, uLongf *destLen, const Bytef *source,
    84  +                         uLong *sourceLen) {
    85  +     int ret;
    86  +     z_size_t got = *destLen, used = *sourceLen;
    87  +     ret = uncompress2_z(dest, &got, source, &used);
    88  +     *sourceLen = (uLong)used;
```

```
        89 +     *destLen = (uLong)got;
        90 +     return ret;
        91 + }
        92 + int ZEXPORT uncompress_z(Bytef *dest, z_size_t *destLen, const Bytef *source,
        93 +                          z_size_t sourceLen) {
        94 +     z_size_t used = sourceLen;
        95 +     return uncompress2_z(dest, destLen, source, &used);
        96 + }
82      97   int ZEXPORT uncompress(Bytef *dest, uLongf *destLen, const Bytef *source,
83      98                          uLong sourceLen) {
84          - -     return uncompress2(dest, destLen, source, &sourceLen);
        99 +     uLong used = sourceLen;
       100 +     return uncompress2(dest, destLen, source, &used);
85     101   }
```

cpan/Compress-Raw-Zlib/zlib-src/zconf.h

```
       ...   @@ -1,5 +1,5 @@
1       1    /* zconf.h -- configuration of the zlib compression library
2           - * Copyright (C) 1995-2024 Jean-loup Gailly, Mark Adler
        2 +  * Copyright (C) 1995-2026 Jean-loup Gailly, Mark Adler
3       3    * For conditions of distribution and use, see copyright notice in zlib.h
4       4    */
5       5
       ...   @@ -8,6 +8,8 @@
8       8    #ifndef ZCONF_H
9       9    #define ZCONF_H
10     10
       11 + #include <stddef.h>         /* Compress::Raw::Zlib change -- Needed for NULL */
       12 +
11     13    /*
12     14     * If you *really* need a unique prefix for all types and library functions,
13     15     * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
       ...   @@ -33,7 +35,10 @@
33     35    #  ifndef Z_SOLO
34     36    #    define compress              z_compress
35     37    #    define compress2             z_compress2
       38 + #    define compress_z            z_compress_z
       39 + #    define compress2_z           z_compress2_z
36     40    #    define compressBound         z_compressBound
       41 + #    define compressBound_z       z_compressBound_z
```

```
37  42      #  endif
38  43      #  define crc32                Perl_crz_crc32
39  44      #  define crc32_combine        Perl_crz_crc32_combine
        @@ -44,6 +49,7 @@
44  49      #  define crc32_z              Perl_crz_crc32_z
45  50      #  define deflate              Perl_crz_deflate
46  51      #  define deflateBound         Perl_crz_deflateBound
    52  +   #  define deflateBound_z       Perl_crz_deflateBound_z
47  53      #  define deflateCopy          Perl_crz_deflateCopy
48  54      #  define deflateEnd           Perl_crz_deflateEnd
49  55      #  define deflateGetDictionary Perl_crz_deflateGetDictionary
        @@ -59,6 +65,7 @@
59  65      #  define deflateSetDictionary Perl_crz_deflateSetDictionary
60  66      #  define deflateSetHeader     Perl_crz_deflateSetHeader
61  67      #  define deflateTune          Perl_crz_deflateTune
    68  +   #  define deflateUsed          Perl_crz_deflateUsed
62  69      #  define deflate_copyright    Perl_crz_deflate_copyright
63  70      #  define get_crc_table        Perl_crz_get_crc_table
64  71      #  ifndef Z_SOLO
        @@ -128,11 +135,15 @@
128 135     #  define inflate_copyright    Perl_crz_inflate_copyright
129 136     #  define inflate_fast         Perl_crz_inflate_fast
130 137     #  define inflate_table        Perl_crz_inflate_table
    138 +   #  define inflate_fixed        Perl_crz_inflate_fixed
131 139     #  ifndef Z_SOLO
132 140     #    define uncompress           z_uncompress
133 141     #    define uncompress2          z_uncompress2
    142 +   #    define uncompress_z         z_uncompress_z
    143 +   #    define uncompress2_z        z_uncompress2_z
134 144     #  endif
135 145     #  define zError               Perl_crz_zError
    146 +   #  define z_errmsg             Perl_crz_errmsg
136 147     #  ifndef Z_SOLO
137 148     #    define zcalloc              z_zcalloc
138 149     #    define zcfree               z_zcfree
        @@ -149,8 +160,8 @@
149 160     #  ifndef Z_SOLO
150 161     #    define gzFile               z_gzFile
151 162     #  endif
```

```
152      - #  define gz_header             Perl_crz_gz_header
153      - #  define gz_headerp            Perl_crz_gz_headerp
         163 + #  define gz_header             Perl_crz_gPerl_crz_header
         164 + #  define gz_headerp            Perl_crz_gPerl_crz_headerp
154      165   #  define in_func              Perl_crz_in_func
155      166   #  define intf                 Perl_crz_intf
156      167   #  define out_func             Perl_crz_out_func
                @@ -234,10 +245,12 @@
234      245   #  endif
235      246   #endif
236      247
237      - #if defined(ZLIB_CONST) && !defined(z_const)
238      - #  define z_const const
239      - #else
240      - #  define z_const
         248 + #ifndef z_const
         249 + #  ifdef ZLIB_CONST
         250 + #    define z_const const
         251 + #  else
         252 + #    define z_const
         253 + #  endif
241      254   #endif
242      255
243      256   #ifdef Z_SOLO
                @@ -408,12 +421,12 @@ typedef uLong FAR uLongf;
408      421
409      422   #ifdef STDC
410      423      typedef void const *voidpc;
411      -    typedef Bytef      *voidpf;
412      -    typedef Bytef      *voidp;
         424 +    typedef void FAR   *voidpf;
         425 +    typedef void       *voidp;
413      426   #else
414      427      typedef Byte const *voidpc;
415      -    typedef Bytef      *voidpf;
416      -    typedef Bytef      *voidp;
         428 +    typedef Byte FAR   *voidpf;
         429 +    typedef Byte       *voidp;
```

```
417  430    #endif
418  431
419  432    #if !defined(Z_U4) && !defined(Z_SOLO) && defined(STDC)
          @@ -433,11 +446,11 @@ typedef uLong FAR uLongf;
433  446       typedef unsigned long z_crc_t;
434  447    #endif
435  448
436       - #ifdef HAVE_UNISTD_H    /* may be set to #if 1 by ./configure */
     449  + #if HAVE_UNISTD_H-0    /* may be set to #if 1 by ./configure */
437  450    #  define Z_HAVE_UNISTD_H
438  451    #endif
439  452
440       - #ifdef HAVE_STDARG_H    /* may be set to #if 1 by ./configure */
     453  + #if HAVE_STDARG_H-0    /* may be set to #if 1 by ./configure */
441  454    #  define Z_HAVE_STDARG_H
442  455    #endif
443  456
          @@ -470,12 +483,8 @@ typedef uLong FAR uLongf;
470  483    #endif
471  484
472  485    #ifndef Z_HAVE_UNISTD_H
473       - #  ifdef __WATCOMC__
474       - #    define Z_HAVE_UNISTD_H
475       - #  endif
476       - #endif
477       - #ifndef Z_HAVE_UNISTD_H
478       - #  if defined(_LARGEFILE64_SOURCE) && !defined(_WIN32)
     486  + #  if defined(__WATCOMC__) || defined(__GO32__) || \
     487  +       (defined(_LARGEFILE64_SOURCE) && !defined(_WIN32))
479  488    #    define Z_HAVE_UNISTD_H
480  489    #  endif
481  490    #endif
          @@ -510,17 +519,19 @@ typedef uLong FAR uLongf;
510  519    #endif
511  520
512  521    #ifndef z_off_t
513       - #  define z_off_t long
     522  + #  define z_off_t long long
```

```
514  523    #endif
515  524
516  525    #if !defined(_WIN32) && defined(Z_LARGE64)
517  526    #  define z_off64_t off64_t
     527  + #elif defined(__MINGW32__)
     528  + #  define z_off64_t long long
     529  + #elif defined(_WIN32) && !defined(__GNUC__)
     530  + #  define z_off64_t __int64
     531  + #elif defined(__GO32__)
     532  + #  define z_off64_t offset_t
518  533    #else
519       - #  if defined(_WIN32) && !defined(__GNUC__)
520       - #    define z_off64_t __int64
521       - #  else
522       - #    define z_off64_t z_off_t
523       - #  endif
     534  + #  define z_off64_t z_off_t
524  535    #endif
525  536
526  537    /* MVS linker does not support external names larger than 8 bytes */
```

cpan/Compress-Raw-Zlib/zlib-src/zlib.h

Load Diff

Large diffs are not rendered by default.

cpan/Compress-Raw-Zlib/zlib-src/zutil.c

```
     •••   @@ -1,5 +1,5 @@
1    1     /* zutil.c -- target dependent utility functions for the compression library
2         -  * Copyright (C) 1995-2017 Jean-loup Gailly
     2     +  * Copyright (C) 1995-2026 Jean-loup Gailly
3    3      * For conditions of distribution and use, see copyright notice in zlib.h
4    4      */
5    5
     @@ -86,28 +86,36 @@ uLong ZEXPORT zlibCompileFlags(void) {
```

```
86  86          flags += 1L << 21;
87  87      #endif
88  88      #if defined(STDC) || defined(Z_HAVE_STDARG_H)
89      -   #  ifdef NO_vsnprintf
90      -       flags += 1L << 25;
91      -   #    ifdef HAS_vsprintf_void
92      -       flags += 1L << 26;
93      -   #    endif
94      -   #  else
95      -   #    ifdef HAS_vsnprintf_void
96      -       flags += 1L << 26;
97      -   #    endif
98      -   #  endif
    89  +   #   ifdef NO_vsnprintf
    90  +   #       ifdef ZLIB_INSECURE
    91  +               flags += 1L << 25;
    92  +   #       else
    93  +               flags += 1L << 27;
    94  +   #       endif
    95  +   #       ifdef HAS_vsprintf_void
    96  +               flags += 1L << 26;
    97  +   #       endif
    98  +   #   else
    99  +   #       ifdef HAS_vsnprintf_void
   100  +               flags += 1L << 26;
   101  +   #       endif
   102  +   #   endif
99  103     #else
100 104         flags += 1L << 24;
101     -   #  ifdef NO_snprintf
102     -       flags += 1L << 25;
103     -   #    ifdef HAS_sprintf_void
104     -       flags += 1L << 26;
105     -   #    endif
106     -   #  else
107     -   #    ifdef HAS_snprintf_void
108     -       flags += 1L << 26;
109     -   #    endif
110     -   #  endif
   105  +   #   ifdef NO_snprintf
```

```
                 + #       ifdef ZLIB_INSECURE
                 +             flags += 1L << 25;
                 + #       else
                 +             flags += 1L << 27;
                 + #       endif
                 + #       ifdef HAS_sprintf_void
                 +             flags += 1L << 26;
                 + #       endif
                 + #   else
                 + #       ifdef HAS_snprintf_void
                 +             flags += 1L << 26;
                 + #       endif
                 + #   endif
111  119           #endif
112  120               return flags;
113  121           }
                   @@ -142,28 +150,34 @@ const char * ZEXPORT zError(int err) {
142  150
143  151           #ifndef HAVE_MEMCPY
144  152
145       - void ZLIB_INTERNAL zmemcpy(Bytef* dest, const Bytef* source, uInt len) {
146       -     if (len == 0) return;
147       -     do {
148       -         *dest++ = *source++; /* ??? to be unrolled */
149       -     } while (--len != 0);
                 + void ZLIB_INTERNAL zmemcpy(void FAR *dst, const void FAR *src, z_size_t n) {
                 +     uchf *p = (uchf *)dst;
                 +     const uchf *q = (const uchf *)src;
                 +     while (n) {
                 +         *p++ = *q++;
                 +         n--;
                 +     }
150  160           }
151  161
152       - int ZLIB_INTERNAL zmemcmp(const Bytef* s1, const Bytef* s2, uInt len) {
153       -     uInt j;
154       -
155       -     for (j = 0; j < len; j++) {
156       -         if (s1[j] != s2[j]) return 2*(s1[j] > s2[j])-1;
```

```
                 + int ZLIB_INTERNAL zmemcmp(const void FAR *s1, const void FAR *s2, z_size_t n) {
                 +     const uchf *p = (const uchf *)s1, *q = (const uchf *)s2;
                 +     while (n) {
                 +         if (*p++ != *q++)
                 +             return (int)p[-1] - (int)q[-1];
                 +         n--;
157  168           }
158  169           return 0;
159  170       }
160  171
161           - void ZLIB_INTERNAL zmemzero(Bytef* dest, uInt len) {
                 + void ZLIB_INTERNAL zmemzero(void FAR *b, z_size_t len) {
                 +     uchf *p = (uchf *)b;
162  174           if (len == 0) return;
163           -     do {
164           -         *dest++ = 0;  /* ??? to be unrolled */
165           -     } while (--len != 0);
                 +     while (len) {
                 +         *p++ = 0;
                 +         len--;
                 +     }
166  179       }
                 +
167  181       #endif
168  182
169  183       #ifndef Z_SOLO
```

cpan/Compress-Raw-Zlib/zlib-src/zutil.h

```
                 @@ -1,5 +1,5 @@
1    1        /* zutil.h -- internal interface and configuration of the compression library
2             - * Copyright (C) 1995-2024 Jean-loup Gailly, Mark Adler
                 + * Copyright (C) 1995-2026 Jean-loup Gailly, Mark Adler
3    3         * For conditions of distribution and use, see copyright notice in zlib.h
4    4         */
5    5
                 @@ -36,6 +36,10 @@
36   36          define "local" for the non-static meaning of "static", for readability
37   37          (compile with -Dlocal if your debugger can't find static symbols) */
```

```
38  38
    39  + extern const char deflate_copyright[];
    40  + extern const char inflate_copyright[];
    41  + extern const char inflate9_copyright[];
    42  +
39  43    typedef unsigned char  uch;
40  44    typedef uch FAR uchf;
41  45    typedef unsigned short ush;
        @@ -48,6 +52,8 @@ typedef unsigned long ulg;
48  52    #    define Z_U8 unsigned long
49  53    #  elif (ULLONG_MAX == 0xffffffffffffffff)
50  54    #    define Z_U8 unsigned long long
    55  + #  elif (ULONG_LONG_MAX == 0xffffffffffffffff)
    56  + #    define Z_U8 unsigned long long
51  57    #  elif (UINT_MAX == 0xffffffffffffffff)
52  58    #    define Z_U8 unsigned
53  59    #  endif
        @@ -63,7 +69,9 @@ extern z_const char * const z_errmsg[10]; /* indexed by 2-
        zlib_error */
63  69    /* To be used only when the state is known to be valid */
64  70
65  71            /* common constants */
66      - 
    72  + #if MAX_WBITS < 9 || MAX_WBITS > 15
    73  + #  error MAX_WBITS must be in 9..15
    74  + #endif
67  75    #ifndef DEF_WBITS
68  76    #  define DEF_WBITS MAX_WBITS
69  77    #endif
        @@ -141,7 +149,7 @@ extern z_const char * const z_errmsg[10]; /* indexed by
        2-zlib_error */
141 149    #  define OS_CODE  7
142 150    #endif
143 151
144     - #ifdef __acorn
    152 + #if defined(__acorn) || defined(__riscos)
145 153    #  define OS_CODE 13
146 154    #endif
147 155
```

```
@@ -168,11 +176,10 @@ extern z_const char * const z_errmsg[10]; /* indexed by 2-zlib_error */
168 176  #endif
169 177
170 178  /* provide prototypes for these when building zlib without LFS */
171     - #if !defined(_WIN32) && \
172     -     (!defined(_LARGEFILE64_SOURCE) || _LFS64_LARGEFILE-0 == 0)
173     -     ZEXTERN uLong ZEXPORT adler32_combine64(uLong, uLong, z_off_t);
174     -     ZEXTERN uLong ZEXPORT crc32_combine64(uLong, uLong, z_off_t);
175     -     ZEXTERN uLong ZEXPORT crc32_combine_gen64(z_off_t);
    179 + #ifndef Z_LARGE64
    180 +    ZEXTERN uLong ZEXPORT adler32_combine64(uLong, uLong, z_off64_t);
    181 +    ZEXTERN uLong ZEXPORT crc32_combine64(uLong, uLong, z_off64_t);
    182 +    ZEXTERN uLong ZEXPORT crc32_combine_gen64(z_off64_t);
176 183  #endif
177 184
178 185          /* common defaults */
@@ -211,9 +218,9 @@ extern z_const char * const z_errmsg[10]; /* indexed by 2-zlib_error */
211 218  #    define zmemzero(dest, len) memset(dest, 0, len)
212 219  #  endif
213 220  #else
214     -    void ZLIB_INTERNAL zmemcpy(Bytef* dest, const Bytef* source, uInt len);
215     -    int ZLIB_INTERNAL zmemcmp(const Bytef* s1, const Bytef* s2, uInt len);
216     -    void ZLIB_INTERNAL zmemzero(Bytef* dest, uInt len);
    221 +    void ZLIB_INTERNAL zmemcpy(void FAR *, const void FAR *, z_size_t);
    222 +    int ZLIB_INTERNAL zmemcmp(const void FAR *, const void FAR *, z_size_t);
    223 +    void ZLIB_INTERNAL zmemzero(void FAR *, z_size_t);
217 224  #endif
218 225
219 226  /* Diagnostic functions */
@@ -251,4 +258,74 @@ extern z_const char * const z_errmsg[10]; /* indexed by 2-zlib_error */
251 258  #define ZSWAP32(q) ((((q) >> 24) & 0xff) + (((q) >> 8) & 0xff00) + \
252 259                      (((q) & 0xff00) << 8) + (((q) & 0xff) << 24))
253 260
    261 + #ifdef Z_ONCE
    262 + /*
    263 +   Create a local z_once() function depending on the availability of atomics.
    264 +  */
```

```
265 +
266 + /* Check for the availability of atomics. */
267 + #if defined(__STDC__) && __STDC_VERSION__ >= 201112L && \
268 +     !defined(__STDC_NO_ATOMICS__)
269 +
270 + #include <stdatomic.h>
271 + typedef struct {
272 +     atomic_flag begun;
273 +     atomic_int done;
274 + } z_once_t;
275 + #define Z_ONCE_INIT {ATOMIC_FLAG_INIT, 0}
276 +
277 + /*
278 +   Run the provided init() function exactly once, even if multiple threads
279 +   invoke once() at the same time. The state must be a once_t initialized with
280 +   Z_ONCE_INIT.
281 +  */
282 + local void z_once(z_once_t *state, void (*init)(void)) {
283 +     if (!atomic_load(&state->done)) {
284 +         if (atomic_flag_test_and_set(&state->begun))
285 +             while (!atomic_load(&state->done))
286 +                 ;
287 +         else {
288 +             init();
289 +             atomic_store(&state->done, 1);
290 +         }
291 +     }
292 + }
293 +
294 + #else   /* no atomics */
295 +
296 + #warning zlib not thread-safe
297 +
298 + typedef struct z_once_s {
299 +     volatile int begun;
300 +     volatile int done;
301 + } z_once_t;
302 + #define Z_ONCE_INIT {0, 0}
303 +
304 + /* Test and set. Alas, not atomic, but tries to limit the period of
```

```
305 +    vulnerability. */
306 + local int test_and_set(int volatile *flag) {
307 +     int was;
308 +
309 +     was = *flag;
310 +     *flag = 1;
311 +     return was;
312 + }
313 +
314 + /* Run the provided init() function once. This is not thread-safe. */
315 + local void z_once(z_once_t *state, void (*init)(void)) {
316 +     if (!state->done) {
317 +         if (test_and_set(&state->begun))
318 +             while (!state->done)
319 +                 ;
320 +         else {
321 +             init();
322 +             state->done = 1;
323 +         }
324 +     }
325 + }
326 +
327 + #endif /* ?atomics */
328 +
329 + #endif /* Z_ONCE */
330 +
254 331   #endif /* ZUTIL_H */
```

Comments 0

Please sign in to comment.