```
**********************************************************
   32586 Thu Jan 17 14:46:46 2019
new/usr/src/lib/libzoneinfo/common/libzone.c
10110 get_tz_countries shouldn't check array for NULL
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */
#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <tzfile.h>
#include <fcntl.h>
#include <regex.h>
#include <errno.h>
#include <libintl.h>
#include <libzoneinfo.h>

#define DEFINIT         "/etc/default/init"
#define ZONEINFOTABDIR  "/usr/share/lib/zoneinfo/tab/"
#define CONTINENT_TAB   ZONEINFOTABDIR "continent.tab"
#define COUNTRY_TAB     ZONEINFOTABDIR "country.tab"
#define ZONE_SUN_TAB    ZONEINFOTABDIR "zone_sun.tab"

#define NEWLINE         "\n"
#define SLASH           "/"
#define WHITESPACE      "\t "
#define WHITESPACE_NL   "\t \n"
#define DIGITS          "0123456789"
#define BUFFLEN         1024

#define CCLEN           2               /* country code length */

#define GMT_MAX         (12*60*60)      /* The maximum GMT offset */
```



```
#define GMT_MIN         (-13*60*60)     /* The minimum GMT offset */
#define GMT_FMT_Q       "<GMT%c%d>%c%d"
#define GMT_FMT_Q_LEN   (11)            /* "<GMT+dd>+dd" - maximum 11 chars */
#define GMT0_FMT        "GMT0"          /* backwards compatibility name */
#define GMT_FMT_ZONE    ":Etc/GMT%c%d"  /* ":Etc/GMT+dd" */
#define GMT_FMT_ZONE_LEN        (11)    /* ":Etc/GMT+dd" - maximum 11 chars */

#define TZ_FMT          "TZ=%s\n"       /* format TZ entry init file */
#define TZ_FMT_Q        "TZ=\"%s\"\n"   /* format quoted TZ entry init file */

#define COORD_FMTLEN1   (sizeof ("+DDMM+DDDMM") - 1)
#define COORD_FMTLEN2   (sizeof ("+DDMMSS+DDDMMSS") - 1)
#define COORD_FMT1              (1)     /* flag for format 1 */
#define COORD_FMT2              (2)     /* flag for format 2 */
#define COORD_DLEN_LAT          (2)     /* length of DD for latitude */
#define COORD_DLEN_LONG         (3)     /* length of DDD for longtitude */
#define COORD_MLEN              (2)     /* length of MM */
#define COORD_SLEN              (2)     /* length of SS */

#define TRAILER         "/XXXXXX"
#define TR_LEN          (sizeof (TRAILER) -1)

/* Internal Declarations */
static char *skipwhite(char *);
static int skipline(char *);
static int trav_link(char **);
static void remove_component(char *);
static void strip_quotes(char *, char *);
static int compar(struct tz_country *, struct tz_country *);
static int get_coord(struct tz_timezone *, char *, size_t);
static int _tz_match(const char *, const char *);
static char *_conv_gmt_zoneinfo(int);
static char *_conv_gmt_posix(int);

/*
 * get_tz_continents() reads the continent.tab file, and
 * returns a list of continents.
 */
int
get_tz_continents(struct tz_continent **cont)
{
        FILE *fp;
        char buff[BUFFLEN];
        char *lp;               /* line pointer */
        char *lptr, *ptr;       /* temp pointer */
        struct tz_continent *head = NULL, *lcp, *prev = NULL;
        int sav_errno = 0, ncount, status;
        size_t len;

        /* open continents file */
        if ((fp = fopen(CONTINENT_TAB, "r")) == NULL) {
                /* fopen() sets errno */
                return (-1);
        }
        /* read and count continents */
        ncount = 0;
        /*CONSTANTCONDITION*/
        while (1) {
                if (fgets(buff, sizeof (buff), fp) == NULL) {
                        if (feof(fp) == 0) {
                                /* fgets() sets errno */
                                sav_errno = errno;
                                ncount = -1;
                        }
                        break;
                }
```

```
127                 /* Skip comments or blank/whitespace lines */
128                 if ((status = skipline(buff)) != 0) {
129                         if (status == 1)
130                                 continue;
131                         else {
132                                 sav_errno = EINVAL;
133                                 ncount = -1;
134                                 break;
135                         }
136                 }
137                 /* Get continent name */
138                 lp = skipwhite(&buff[0]);
139                 if ((len = strcspn(lp, WHITESPACE)) > _TZBUFLEN -1) {
140                         sav_errno = ENAMETOOLONG;
141                         ncount = -1;
142                         break;
143                 }
144                 /* create continent struct */
145                 if ((lcp = (struct tz_continent *)
146                         calloc(1, sizeof (struct tz_continent))) == NULL) {
147                         sav_errno = ENOMEM;
148                         ncount = -1;
149                         break;
150                 }
151                 (void) strncpy(lcp->ctnt_name, lp, len);
152                 lcp->ctnt_name[len] = '\0';

154                 /* Get continent description */
155                 lp = skipwhite(lp + len);
156                 len = strcspn(lp, NEWLINE);
157                 if ((ptr = malloc(len + 1)) == NULL) {
158                         (void) free_tz_continents(lcp);
159                         sav_errno = ENOMEM;
160                         ncount = -1;
161                         break;
162                 }
163                 (void) strncpy(ptr, lp, len);
164                 *(ptr + len) = '\0';
165                 lcp->ctnt_id_desc = ptr;

167                 /* Get localized continent description */
168                 lptr = dgettext(TEXT_DOMAIN, lcp->ctnt_id_desc);
169                 if ((ptr = strdup(lptr)) == NULL) {
170                         (void) free_tz_continents(lcp);
171                         sav_errno = ENOMEM;
172                         ncount = -1;
173                         break;
174                 }
175                 lcp->ctnt_display_desc = ptr;

177                 if (head == NULL) {
178                         head = lcp;
179                 } else {
180                         prev->ctnt_next = lcp;
181                 }
182                 prev = lcp;
183                 ncount++;
184         }
185         (void) fclose(fp);
186         if (ncount == -1) {
187                 if (head != NULL) {
188                         (void) free_tz_continents(head);
189                 }
190                 if (sav_errno)
191                         errno = sav_errno;
192         } else {
```



```
193                 *cont = head;
194         }
195         return (ncount);
196 }

198 /*
199  * get_tz_countries() finds the list of countries from the zone_sun.tab
200  * file, for the input continent, and retrieves the country
201  * names from the country.tab file.  It also retrieves the localized
202  * country names.  The returned list of countries is sorted by the
203  * countries' localized name fields.
204  */
205 int
206 get_tz_countries(struct tz_country **country, struct tz_continent *cont)
207 {
208         FILE *fp_zone, *fp_cc;
209         char buff[BUFFLEN], ccbuf[_CCBUFLEN], *ptr;
210         char *lp, *lptr, *lp_coord, *lp_cc, *lp_tz;     /* line pointer */
211         struct tz_country *head = NULL, *prev = NULL, *next, *cp, *cp2;
212         int sav_errno = 0, ncount, i;
213         int cmp, status;
214         size_t len, len_coord, len_ctnt;

214         if (cont->ctnt_name == NULL) {
215                 errno = EINVAL;
216                 return (-1);
217         }
216         len_ctnt = strlen(cont->ctnt_name);
217         ccbuf[0] = '\0';

219         /* open zone_sun.tab and country.tab files */
220         if ((fp_zone = fopen(ZONE_SUN_TAB, "r")) == NULL) {
221                 /* fopen() sets errno */
222                 return (-1);
223         }
224         if ((fp_cc = fopen(COUNTRY_TAB, "r")) == NULL) {
225                 /* fopen() sets errno */
226                 (void) fclose(fp_zone);
227                 return (-1);
228         }

230         /* read timezones to match continents, and get countries */
231         ncount = 0;
232         /*CONSTANTCONDITION*/
233         while (1) {
234                 if (fgets(buff, sizeof (buff), fp_zone) == NULL) {
235                         if (feof(fp_zone) == 0) {
236                                 /* fgets() error - errno set */
237                                 sav_errno = errno;
238                                 ncount = -1;
239                         }
240                         break;
241                 }
242                 /* Skip comments or blank/whitespace lines */
243                 if ((status = skipline(buff)) != 0) {
244                         if (status == 1)
245                                 continue;
246                         else {
247                                 sav_errno = EINVAL;
248                                 ncount = -1;
249                                 break;
250                         }
251                 }
252                 /*
253                  * If country matches previously *matched* country, skip
254                  * entry, since zone.tab is alphabetized by country code
```

```
255                  * (It should be a *matched* country, because the same country
256                  * can be in different continents.)
257                  */
258                 /* Get country code */
259                 lp_cc = skipwhite(&buff[0]);
260                 if (strcspn(lp_cc, WHITESPACE) != CCLEN) {
261                         ncount = -1;
262                         sav_errno = EINVAL;
263                         break;
264                 }
265                 /* Check country code cache; skip if already found */
266                 if (strncmp(ccbuf, lp_cc, CCLEN) == 0) {
267                         continue;
268                 }
269                 /* Get coordinates */
270                 lp_coord = skipwhite(lp_cc + CCLEN);
271                 if (((len_coord = strcspn(lp_coord, WHITESPACE)) !=
272                                 COORD_FMTLEN1) &&
273                                 (len_coord != COORD_FMTLEN2)) {
274                         ncount = -1;
275                         sav_errno = EINVAL;
276                         break;
277                 }

279                 /* Get timezone name (Skip timezone description) */
280                 lp_tz = skipwhite(lp_coord + len_coord);
281                 if ((len = strcspn(lp_tz, SLASH)) == 0) {
282                         ncount = -1;
283                         sav_errno = EINVAL;
284                         break;
285                 }
286                 /* If continents match, allocate a country struct */
287                 if ((len == len_ctnt) &&
288                                 (strncmp(cont->ctnt_name, lp_tz, len) == 0)) {
289                         if ((cp = (struct tz_country *)
290                             calloc(1, sizeof (struct tz_country))) == NULL) {
291                                 sav_errno = ENOMEM;
292                                 ncount = -1;
293                                 break;
294                         }
295                         /* Copy and save country code (len already checked) */
296                         (void) strncpy(cp->ctry_code, lp_cc, CCLEN);
297                         cp->ctry_code[CCLEN] = '\0';
298                         (void) strncpy(ccbuf, lp_cc, CCLEN);
299                         ccbuf[CCLEN] = '\0';

301                         /* Create linked list */
302                         if (head == NULL) {
303                                 head = cp;
304                         } else {
305                                 prev->ctry_next = cp;
306                         };
307                         prev = cp;
308                         ncount++;
309                 }
310         }       /* while */

312         if (ncount == -1)
313                 goto error;

315         /* Get country name from country.tab; get localized country name */
316         /* Read country list, match country codes to process entry */
317         cp = head;
318         /*CONSTANTCONDITION*/
319         while (1) {
320                 if (fgets(buff, sizeof (buff), fp_cc) == NULL) {
```



```
321                         if (feof(fp_cc) == 0) {
322                                 /* fgets() sets errno */
323                                 ncount = -1;
324                                 sav_errno = errno;
325                         }
326                         break;
327                 }
328                 /* Skip comments or blank/whitespace lines */
329                 if ((status = skipline(buff)) != 0) {
330                         if (status == 1)
331                                 continue;
332                         else {
333                                 sav_errno = EINVAL;
334                                 ncount = -1;
335                                 break;
336                         }
337                 }
338                 /* Match country codes */
339                 if ((len = strcspn(buff, WHITESPACE)) != CCLEN) {
340                         sav_errno = EINVAL;
341                         ncount = -1;
342                         break;
343                 }
344                 if ((cmp = strncmp(cp->ctry_code, buff, CCLEN)) == 0) {
345                         /* Get country description, and localized desc. */
346                         /* Skip to country description */
347                         lp = &buff[CCLEN];
348                         if ((len = strspn(lp, WHITESPACE)) == 0) {
349                                 sav_errno = EINVAL;
350                                 ncount = -1;
351                                 break;
352                         }
353                         lp += len;              /* lp points to country desc. */
354                         len = strcspn(lp, NEWLINE);
355                         if ((ptr = calloc(len + 1, 1)) == NULL) {
356                                 ncount = -1;
357                                 errno = ENOMEM;
358                                 break;
359                         }
360                         (void) strncpy(ptr, lp, len);
361                         *(ptr + len) = '\0';
362                         cp->ctry_id_desc = ptr;

364                         /* Get localized country description */
365                         lptr = dgettext(TEXT_DOMAIN, ptr);
366                         if ((ptr = strdup(lptr)) == NULL) {
367                                 ncount = -1;
368                                 errno = ENOMEM;
369                                 break;
370                         }
371                         cp->ctry_display_desc = ptr;
372                 } else if (cmp > 0) {
373                         /* Keep searching country.tab */
374                         continue;
375                 } else {
376                         /* Not found - should not happen */
377                         ncount = -1;
378                         errno = EILSEQ;
379                         break;
380                 }
381                 if (cp->ctry_next == NULL) {
382                         /* done with countries list */
383                         break;
384                 } else {
385                         cp = cp->ctry_next;
386                 }
```

```
387         }               /* while */

389         /* Now sort the list by ctry_display_desc field */
390         if ((ncount != -1) &&
391                 ((cp2 = calloc(ncount, sizeof (struct tz_country))) != NULL)) {
392                 /*
393                  * First copy list to a static array for qsort() to use.
394                  * Use the cnt_next field to point back to original structure.
395                  */
396                 cp = head;
397                 for (i = 0; i < ncount; i++) {
398                         next = cp->ctry_next;
399                         cp->ctry_next = cp;
400                         (void) memcpy(&cp2[i], cp, sizeof (struct tz_country));
401                         cp = next;
402                 }

404                 /* Next, call qsort() using strcoll() to order */
405                 qsort(cp2, ncount, sizeof (struct tz_country),
406                         (int (*)(const void *, const void *))compar);

408                 /* Rearrange the country list according to qsort order */
409                 head = cp2->ctry_next; /* ctry_next is pointer to orig struct */
410                 cp = head;
411                 for (i = 0; i < ncount; i++) {
412                         prev = cp;
413                         cp = cp2[i].ctry_next;
414                         prev->ctry_next = cp;
415                 }
416                 cp->ctry_next = NULL;

418                 /* Last, free the static buffer */
419                 free(cp2);

421         } else {
422                 if (ncount != -1)
423                         ncount = -1;
424         }

426 error:
427         (void) fclose(fp_zone);
428         (void) fclose(fp_cc);
429         if (ncount == -1) {
430                 /* free the linked list */
431                 if (head != NULL)
432                         (void) free_tz_countries(head);
433                 if (sav_errno)
434                         errno = sav_errno;
435         } else {
436                 *country = head;
437         }
438         return (ncount);
439 }
```

_____***unchanged_portion_omitted_***