```
**********************************************************
   10564 Sun Jan 12 10:08:59 2014
new/usr/src/lib/libc/port/locale/strptime.c
3141 strptime() doesn't support %t
**********************************************************
```

```
/*
 * Copyright (c) 2014 Gary Mills
 * Copyright 2011, Nexenta Systems, Inc.  All rights reserved.
 * Copyright (c) 1994 Powerdog Industries.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer
 *    in the documentation and/or other materials provided with the
 *    distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY POWERDOG INDUSTRIES ``AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE POWERDOG INDUSTRIES BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
 * BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * The views and conclusions contained in the software and documentation
 * are those of the authors and should not be interpreted as representing
 * official policies, either expressed or implied, of Powerdog Industries.
 */

#include "lint.h"
#include <time.h>
#include <ctype.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <alloca.h>
#include "timelocal.h"

#define asizeof(a)      (sizeof (a) / sizeof ((a)[0]))

#define F_GMT           (1 << 0)
#define F_ZERO          (1 << 1)
#define F_RECURSE       (1 << 2)

static char *
__strptime(const char *buf, const char *fmt, struct tm *tm, int *flagsp)
{
        char    c;
        const char *ptr;
        int     i, len, recurse = 0;
        int Ealternative, Oalternative;
        struct lc_time_T *tptr = __get_current_time_locale();

        if (*flagsp & F_RECURSE)
                recurse = 1;
```



```
        *flagsp |= F_RECURSE;

        if (*flagsp & F_ZERO)
                (void) memset(tm, 0, sizeof (*tm));
        *flagsp &= ~F_ZERO;

        ptr = fmt;
        while (*ptr != 0) {
                if (*buf == 0)
                        break;

                c = *ptr++;

                if (c != '%') {
                        if (isspace(c))
                                while (isspace(*buf))
                                        buf++;
                        else if (c != *buf++)
                                return (NULL);
                        continue;
                }

                Ealternative = 0;
                Oalternative = 0;
label:
                c = *ptr++;
                switch (c) {
                case 0:
                case '%':
                        if (*buf++ != '%')
                                return (NULL);
                        break;

                case '+':
                        buf = __strptime(buf, tptr->date_fmt, tm, flagsp);
                        if (buf == NULL)
                                return (NULL);
                        break;

                case 'C':
                        if (!isdigit(*buf))
                                return (NULL);

                        /* XXX This will break for 3-digit centuries. */
                        len = 2;
                        for (i = 0; len && isdigit(*buf); buf++) {
                                i *= 10;
                                i += *buf - '0';
                                len--;
                        }
                        if (i < 19)
                                return (NULL);

                        tm->tm_year = i * 100 - 1900;
                        break;

                case 'c':
                        buf = __strptime(buf, tptr->c_fmt, tm, flagsp);
                        if (buf == NULL)
                                return (NULL);
                        break;

                case 'D':
                        buf = __strptime(buf, "%m/%d/%y", tm, flagsp);
                        if (buf == NULL)
                                return (NULL);
```

```
128                         break;

130                 case 'E':
131                         if (Ealternative || Oalternative)
132                                 break;
133                         Ealternative++;
134                         goto label;

136                 case 'O':
137                         if (Ealternative || Oalternative)
138                                 break;
139                         Oalternative++;
140                         goto label;

142                 case 'F':
143                         buf = __strptime(buf, "%Y-%m-%d", tm, flagsp);
144                         if (buf == NULL)
145                                 return (NULL);
146                         break;

148                 case 'R':
149                         buf = __strptime(buf, "%H:%M", tm, flagsp);
150                         if (buf == NULL)
151                                 return (NULL);
152                         break;

154                 case 'r':
155                         buf = __strptime(buf, tptr->ampm_fmt, tm, flagsp);
156                         if (buf == NULL)
157                                 return (NULL);
158                         break;

160                 case 'T':
161                         buf = __strptime(buf, "%H:%M:%S", tm, flagsp);
162                         if (buf == NULL)
163                                 return (NULL);
164                         break;

166                 case 'X':
167                         buf = __strptime(buf, tptr->X_fmt, tm, flagsp);
168                         if (buf == NULL)
169                                 return (NULL);
170                         break;

172                 case 'x':
173                         buf = __strptime(buf, tptr->x_fmt, tm, flagsp);
174                         if (buf == NULL)
175                                 return (NULL);
176                         break;

178                 case 'j':
179                         if (!isdigit(*buf))
180                                 return (NULL);

182                         len = 3;
183                         for (i = 0; len && isdigit(*buf); buf++) {
184                                 i *= 10;
185                                 i += *buf - '0';
186                                 len--;
187                         }
188                         if (i < 1 || i > 366)
189                                 return (NULL);

191                         tm->tm_yday = i - 1;
192                         break;
```



```
194                 case 'M':
195                 case 'S':
196                         if (*buf == 0 || isspace(*buf))
197                                 break;

199                         if (!isdigit(*buf))
200                                 return (NULL);

202                         len = 2;
203                         for (i = 0; len && isdigit(*buf); buf++) {
204                                 i *= 10;
205                                 i += *buf - '0';
206                                 len--;
207                         }

209                         if (c == 'M') {
210                                 if (i > 59)
211                                         return (NULL);
212                                 tm->tm_min = i;
213                         } else {
214                                 if (i > 60)
215                                         return (NULL);
216                                 tm->tm_sec = i;
217                         }

218                         if (isspace(*buf))
219                                 while (*ptr != 0 && !isspace(*ptr))
220                                         ptr++;
219                         break;

221                 case 'H':
222                 case 'I':
223                 case 'k':
224                 case 'l':
225                         /*
226                          * Of these, %l is the only specifier explicitly
227                          * documented as not being zero-padded.  However,
228                          * there is no harm in allowing zero-padding.
229                          *
230                          * XXX The %l specifier may gobble one too many
231                          * digits if used incorrectly.
232                          */
233                         if (!isdigit(*buf))
234                                 return (NULL);

236                         len = 2;
237                         for (i = 0; len && isdigit(*buf); buf++) {
238                                 i *= 10;
239                                 i += *buf - '0';
240                                 len--;
241                         }
242                         if (c == 'H' || c == 'k') {
243                                 if (i > 23)
244                                         return (NULL);
245                         } else if (i > 12)
246                                 return (NULL);

248                         tm->tm_hour = i;

252                         if (isspace(*buf))
253                                 while (*ptr != 0 && !isspace(*ptr))
254                                         ptr++;
250                         break;

252                 case 'p':
253                         /*
```

```
254                          * XXX This is bogus if parsed before hour-related
255                          * specifiers.
256                          */
257                         len = strlen(tptr->am);
258                         if (strncasecmp(buf, tptr->am, len) == 0) {
259                                 if (tm->tm_hour > 12)
260                                         return (NULL);
261                                 if (tm->tm_hour == 12)
262                                         tm->tm_hour = 0;
263                                 buf += len;
264                                 break;
265                         }

267                         len = strlen(tptr->pm);
268                         if (strncasecmp(buf, tptr->pm, len) == 0) {
269                                 if (tm->tm_hour > 12)
270                                         return (NULL);
271                                 if (tm->tm_hour != 12)
272                                         tm->tm_hour += 12;
273                                 buf += len;
274                                 break;
275                         }

277                         return (NULL);

279                 case 'A':
280                 case 'a':
281                         for (i = 0; i < asizeof(tptr->weekday); i++) {
282                                 len = strlen(tptr->weekday[i]);
283                                 if (strncasecmp(buf, tptr->weekday[i], len) ==
284                                     0)
285                                         break;
286                                 len = strlen(tptr->wday[i]);
287                                 if (strncasecmp(buf, tptr->wday[i], len) == 0)
288                                         break;
289                         }
290                         if (i == asizeof(tptr->weekday))
291                                 return (NULL);

293                         tm->tm_wday = i;
294                         buf += len;
295                         break;

297                 case 'U':
298                 case 'W':
299                         /*
300                          * XXX This is bogus, as we can not assume any valid
301                          * information present in the tm structure at this
302                          * point to calculate a real value, so just check the
303                          * range for now.
304                          */
305                         if (!isdigit(*buf))
306                                 return (NULL);

308                         len = 2;
309                         for (i = 0; len && isdigit(*buf); buf++) {
310                                 i *= 10;
311                                 i += *buf - '0';
312                                 len--;
313                         }
314                         if (i > 53)
315                                 return (NULL);

322                         if (isspace(*buf))
323                                 while (*ptr != 0 && !isspace(*ptr))
324                                         ptr++;
```



```
317                         break;

319                 case 'w':
320                         if (!isdigit(*buf))
321                                 return (NULL);

323                         i = *buf - '0';
324                         if (i > 6)
325                                 return (NULL);

327                         tm->tm_wday = i;

337                         if (isspace(*buf))
338                                 while (*ptr != 0 && !isspace(*ptr))
339                                         ptr++;
329                         break;

331                 case 'd':
332                 case 'e':
333                         /*
334                          * The %e format has a space before single digits
335                          * which we need to skip.
336                          */
337                         if (isspace(*buf))
338                                 buf++;
349                         /* FALLTHROUGH */
350                 case 'd':
339                         /*
340                          * The %e specifier is explicitly documented as not
341                          * being zero-padded but there is no harm in allowing
342                          * such padding.
343                          *
344                          * XXX The %e specifier may gobble one too many
345                          * digits if used incorrectly.
346                          */
347                         if (!isdigit(*buf))
348                                 return (NULL);

350                         len = 2;
351                         for (i = 0; len && isdigit(*buf); buf++) {
352                                 i *= 10;
353                                 i += *buf - '0';
354                                 len--;
355                         }
356                         if (i > 31)
357                                 return (NULL);

359                         tm->tm_mday = i;

373                         if (isspace(*buf))
374                                 while (*ptr != 0 && !isspace(*ptr))
375                                         ptr++;
361                         break;

363                 case 'B':
364                 case 'b':
365                 case 'h':
366                         for (i = 0; i < asizeof(tptr->month); i++) {
367                                 len = strlen(tptr->month[i]);
368                                 if (strncasecmp(buf, tptr->month[i], len) == 0)
369                                         break;
370                         }
371                         /*
372                          * Try the abbreviated month name if the full name
373                          * wasn't found.
374                          */
```

```
375                         if (i == asizeof(tptr->month)) {
376                                 for (i = 0; i < asizeof(tptr->month); i++) {
377                                         len = strlen(tptr->mon[i]);
378                                         if (strncasecmp(buf, tptr->mon[i],
379                                             len) == 0)
380                                                 break;
381                                 }
382                         }
383                         if (i == asizeof(tptr->month))
384                                 return (NULL);

386                         tm->tm_mon = i;
387                         buf += len;
388                         break;

390                 case 'm':
391                         if (!isdigit(*buf))
392                                 return (NULL);

394                         len = 2;
395                         for (i = 0; len && isdigit(*buf); buf++) {
396                                 i *= 10;
397                                 i += *buf - '0';
398                                 len--;
399                         }
400                         if (i < 1 || i > 12)
401                                 return (NULL);

403                         tm->tm_mon = i - 1;

420                         if (isspace(*buf))
421                                 while (*ptr != NULL && !isspace(*ptr))
422                                         ptr++;
405                         break;

407                 case 's':
408                         {
409                         char *cp;
410                         int sverrno;
411                         time_t t;

413                         sverrno = errno;
414                         errno = 0;
415                         t = strtol(buf, &cp, 10);
416                         if (errno == ERANGE) {
417                                 errno = sverrno;
418                                 return (NULL);
419                         }
420                         errno = sverrno;
421                         buf = cp;
422                         (void) gmtime_r(&t, tm);
423                         *flagsp |= F_GMT;
424                         }
425                         break;

427                 case 'Y':
428                 case 'y':
429                         if (*buf == NULL || isspace(*buf))
430                                 break;

432                         if (!isdigit(*buf))
433                                 return (NULL);

435                         len = (c == 'Y') ? 4 : 2;
436                         for (i = 0; len && isdigit(*buf); buf++) {
437                                 i *= 10;
```



```
438                                 i += *buf - '0';
439                                 len--;
440                         }
441                         if (c == 'Y')
442                                 i -= 1900;
443                         if (c == 'y' && i < 69)
444                                 i += 100;
445                         if (i < 0)
446                                 return (NULL);

448                         tm->tm_year = i;

468                         if (isspace(*buf))
469                                 while (*ptr != 0 && !isspace(*ptr))
470                                         ptr++;
450                         break;

452                 case 'Z':
453                         {
454                         const char *cp = buf;
455                         char *zonestr;

457                         while (isupper(*cp))
458                                 ++cp;
459                         if (cp - buf) {
460                                 zonestr = alloca(cp - buf + 1);
461                                 (void) strncpy(zonestr, buf, cp - buf);
462                                 zonestr[cp - buf] = '\0';
463                                 tzset();
464                                 if (strcmp(zonestr, "GMT") == 0) {
465                                         *flagsp |= F_GMT;
466                                 } else if (0 == strcmp(zonestr, tzname[0])) {
467                                         tm->tm_isdst = 0;
468                                 } else if (0 == strcmp(zonestr, tzname[1])) {
469                                         tm->tm_isdst = 1;
470                                 } else {
471                                         return (NULL);
472                                 }
473                                 buf += cp - buf;
474                         }
475                         }
476                         break;

478                 case 'z':
479                         {
480                         int sign = 1;

482                         if (*buf != '+') {
483                                 if (*buf == '-')
484                                         sign = -1;
485                                 else
486                                         return (NULL);
487                         }
488                         buf++;
489                         i = 0;
490                         for (len = 4; len > 0; len--) {
491                                 if (!isdigit(*buf))
492                                         return (NULL);
493                                 i *= 10;
494                                 i += *buf - '0';
495                                 buf++;
496                         }

498                         tm->tm_hour -= sign * (i / 100);
499                         tm->tm_min -= sign * (i % 100);
500                         *flagsp |= F_GMT;
```

```
501                         }
502                         break;
503                 case 'n':
504                 case 't':
505                         while (isspace(*buf))
506                                 buf++;
507                         break;
508                 }
509         }

511         if (!recurse) {
512                 if (buf && (*flagsp & F_GMT)) {
513                         time_t t = timegm(tm);
514                         (void) localtime_r(&t, tm);
515                 }
516         }

518         return ((char *)buf);
519 }
```

*unchanged_portion_omitted_*