

```

*****
86490 Fri Jan 3 13:06:05 2014
new/usr/src/cmd/cron/cron.c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23  * Use is subject to license terms.
24  *
25  * Copyright 2013 Joshua M. Clulow <josh@sysmgr.org>
26  *
27  * Copyright (c) 2014 Gary Mills
28  */

30 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
31 /*      All Rights Reserved */

33 /*      Copyright (c) 1987, 1988 Microsoft Corporation */
34 /*      All Rights Reserved */

36 #ifndef lint
37 /* make lint happy */
38 #define __EXTENSIONS__
39 #endif

41 #include <sys/contract/process.h>
42 #include <sys/ctfs.h>
43 #include <sys/param.h>
44 #include <sys/resource.h>
45 #include <sys/stat.h>
46 #include <sys/task.h>
47 #include <sys/time.h>
48 #include <sys/types.h>
49 #include <sys/utsname.h>
50 #include <sys/wait.h>

52 #include <security/pam_appl.h>

54 #include <alloca.h>
55 #include <ctype.h>
56 #include <deflt.h>
57 #include <dirent.h>
58 #include <errno.h>
59 #include <fcntl.h>
60 #include <grp.h>

```

```

61 #include <libcontract.h>
62 #include <libcontract_priv.h>
63 #include <limits.h>
64 #include <locale.h>
65 #include <poll.h>
66 #include <project.h>
67 #include <pwd.h>
68 #include <signal.h>
69 #include <stdarg.h>
70 #include <stdio.h>
71 #include <stdlib.h>
72 #include <string.h>
73 #include <stropts.h>
74 #include <time.h>
75 #include <unistd.h>
76 #include <libzoneinfo.h>

78 #include "cron.h"

80 /*
81  * #define      DEBUG
82  */

84 #define MAIL      "/usr/bin/mail" /* mail program to use */
85 #define CONSOLE   "/dev/console" /* where messages go when cron dies */

87 #define TMPINFILE "/tmp/crinXXXXXX" /* file to put stdin in for cmd */
88 #define TMPDIR    "/tmp"
89 #define PFX       "crout"
90 #define TMPOUTFILE "/tmp/croutXXXXXX" /* file to place stdout, stderr */

92 #define INMODE    00400 /* mode for stdin file */
93 #define OUTMODE   00600 /* mode for stdout file */
94 #define ISUID     S_ISUID /* mode for verifying at jobs */

96 #define INFINITY  2147483647L /* upper bound on time */
97 #define CUSHION   180L
98 #define ZOMB      100 /* proc slot used for mailing output */

100 #define JOBF      'j'
101 #define NICEF      'n'
102 #define USERF     'u'
103 #define WAITF     'w'

105 #define BCHAR     '>'
106 #define ECHAR     '<'

108 #define DEFAULT   0
109 #define LOAD      1
110 #define QBUFSIZ   80

112 /* Defined actions for crabort() routine */
113 #define NO_ACTION 000
114 #define REMOVE_FIFO 001
115 #define CONSOLE_MSG 002

117 #define BADCD     "can't change directory to the crontab directory."
118 #define NOREADDIR "can't read the crontab directory."

120 #define BADJOBOPEN "unable to read your at job."
121 #define BADSHELL  "because your login shell \
122 isn't /usr/bin/sh, you can't use cron."

124 #define BADSTAT   "can't access your crontab or at-job file. Resubmit it."
125 #define BADPROJID "can't set project id for your job."
126 #define CANTCDHOME "can't change directory to %s.\"

```



```

363 * BSM hooks
364 */
365 extern int audit_cron_session(char *, char *, uid_t, gid_t, char *);
366 extern void audit_cron_new_job(char *, int, void *);
367 extern void audit_cron_bad_user(char *);
368 extern void audit_cron_user_acct_expired(char *);
369 extern int audit_cron_create_anc_file(char *, char *, char *, uid_t);
370 extern int audit_cron_delete_anc_file(char *, char *);
371 extern int audit_cron_is_anc_name(char *);
372 extern int audit_cron_mode();

374 static int cron_conv(int, struct pam_message **,
375 struct pam_response **, void *);

377 static struct pam_conv pam_conv = {cron_conv, NULL};
378 static pam_handle_t *pamh; /* Authentication handle */

380 /*
381 * Function to help check a user's credentials.
382 */

384 static int verify_user_cred(struct usr *u);

386 /*
387 * Values returned by verify_user_cred and set_user_cred:
388 */

390 #define VUC_OK 0
391 #define VUC_BADUSER 1
392 #define VUC_NOTINGROUP 2
393 #define VUC_EXPIRED 3
394 #define VUC_NEW_AUTH 4

396 /*
397 * Modes of process_anc_files function
398 */
399 #define CRON_ANC_DELETE 1
400 #define CRON_ANC_CREATE 0

402 /*
403 * Functions to remove a user or job completely from the running database.
404 */
405 static void clean_out_atjobs(struct usr *u);
406 static void clean_out_ctab(struct usr *u);
407 static void clean_out_user(struct usr *u);
408 static void cron_unlink(char *name);
409 static void process_anc_files(int);

411 /*
412 * functions in elm.c
413 */
414 extern void el_init(int, time_t, time_t, int);
415 extern int el_add(void *, time_t, int);
416 extern void el_remove(int, int);
417 extern int el_empty(void);
418 extern void *el_first(void);
419 extern void el_delete(void);

421 static int valid_entry(char *, int);
422 static struct usr *create_ulist(char *, int);
423 static void init_cronevent(char *, int);
424 static void init_atevent(char *, time_t, int, int);
425 static void update_atevent(struct usr *, char *, time_t, int);

427 int
428 main(int argc, char *argv[])

```

```

429 {
430     time_t t;
431     time_t ne_time; /* amt of time until next event execution */
432     time_t newtime, lastmtime = 0L;
433     struct usr *u;
434     struct event *e, *e2, *eprev;
435     struct stat buf;
436     pid_t rfork;
437     struct sigaction act;

439     /*
440     * reset_needed is set to 1 whenever el_add() finds out that a cron
441     * job is scheduled to be run before the time when cron(lm) daemon
442     * initialized.
443     * Other cases where a reset is needed is when ex() finds that the
444     * event to be executed is being run at the wrong time, or when idle()
445     * determines that time was reset.
446     * We immediately return to the top of the while (TRUE) loop in
447     * main() where the event list is cleared and rebuilt, and reset_needed
448     * is set back to 0.
449     */
450     reset_needed = 0;

452     /*
453     * Only the privileged user can run this command.
454     */
455     if (getuid() != 0)
456         crabort(NOTALLOWED, 0);

458 begin:
459     (void) setlocale(LC_ALL, "");
460     /* fork unless 'nofork' is specified */
461     if ((argc <= 1) || (strcmp(argv[1], "nofork"))) {
462         if (rfork = fork()) {
463             if (rfork == (pid_t)-1) {
464                 (void) sleep(30);
465                 goto begin;
466             }
467             return (0);
468         }
469         didfork++;
470         (void) setpgrp(); /* detach cron from console */
471     }

473     (void) umask(022);
474     (void) signal(SIGHUP, SIG_IGN);
475     (void) signal(SIGINT, SIG_IGN);
476     (void) signal(SIGQUIT, SIG_IGN);
477     (void) signal(SIGTERM, cronend);

479     defaults();
480     initialize(1);
481     quedefs(DEFAULT); /* load default queue definitions */
482     cron_pid = getpid();
483     msg("*** cron started *** pid = %d", cron_pid);

485     /* setup THAW handler */
486     act.sa_handler = thaw_handler;
487     act.sa_flags = 0;
488     (void) sigemptyset(&act.sa_mask);
489     (void) sigaction(SIGTHAW, &act, NULL);

491     /* setup CHLD handler */
492     act.sa_handler = child_handler;
493     act.sa_flags = 0;
494     (void) sigemptyset(&act.sa_mask);

```



```

626     }
627 #ifdef DEBUG
628     cftime(timebuf, "%+", &next_event->time);
629     cftime(timebuf, "%C", &next_event->time);
630     (void) fprintf(stderr,
631     "pushing back cron event %s at %ld (%s)\n",
632     next_event->cmd, next_event->time, timebuf);
633 #endif
634
635     switch (el_add(next_event, next_event->time,
636     (next_event->u)->ctid)) {
637     case -1:
638         ignore_msg("main", "cron", next_event);
639         break;
640     case -2: /* event time lower than init time */
641         reset_needed = 1;
642         break;
643     }
644     default:
645         /* remove at or batch job from system */
646         if (delayed) {
647             delayed = 0;
648             break;
649         }
650         eprev = NULL;
651         e = (next_event->u)->atevents;
652         while (e != NULL) {
653             if (e == next_event) {
654                 if (eprev == NULL)
655                     (e->u)->atevents = e->link;
656                 else
657                     eprev->link = e->link;
658                 free(e->cmd);
659                 free(e);
660                 break;
661             } else {
662                 eprev = e;
663                 e = e->link;
664             }
665         }
666         break;
667     }
668     next_event = NULL;
669 }
670
671 /*NOTREACHED*/
672 }

```

unchanged portion omitted

```

1116 static char line[CTLINE_SIZE]; /* holds a line from a crontab file */
1117 static int cursor; /* cursor for the above line */
1118
1119 static void
1120 readcron(struct usr *u, time_t reftime)
1121 {
1122     /*
1123     * readcron reads in a crontab file for a user (u). The list of
1124     * events for user u is built, and u->events is made to point to
1125     * this list. Each event is also entered into the main event
1126     * list.
1127     */
1128     FILE *cf; /* cf will be a user's crontab file */
1129     struct event *e;
1130     int start;
1131     unsigned int i;

```

```

1132     char namebuf[PATH_MAX];
1133     char *pname;
1134     struct shared *tz = NULL;
1135     struct shared *home = NULL;
1136     struct shared *shell = NULL;
1137     int lineno = 0;
1138
1139     /* read the crontab file */
1140     cte_init(); /* Init error handling */
1141     if (cwd != CRON) {
1142         if (snprintf(namebuf, sizeof (namebuf), "%s/%s",
1143         CRONDIR, u->name) >= sizeof (namebuf)) {
1144             return;
1145         }
1146     } else {
1147         pname = namebuf;
1148     }
1149     if ((cf = fopen(pname, "r")) == NULL) {
1150         mail(u->name, NOREAD, ERR_UNIXERR);
1151         return;
1152     }
1153     while (fgets(line, CTLINE_SIZE, cf) != NULL) {
1154         char *tmp;
1155         /* process a line of a crontab file */
1156         lineno++;
1157         if (cte_istoomany())
1158             break;
1159         cursor = 0;
1160         while (line[cursor] == ' ' || line[cursor] == '\t')
1161             cursor++;
1162         if (line[cursor] == '#' || line[cursor] == '\n')
1163             continue;
1164
1165         if (strncmp(&line[cursor], ENV_TZ,
1166         strlen(ENV_TZ)) == 0) {
1167             if ((tmp = strchr(&line[cursor], '\n')) != NULL) {
1168                 *tmp = NULL;
1169             }
1170
1171             if (!isvalid_tz(&line[cursor + strlen(ENV_TZ)], NULL,
1172             _VTZ_ALL)) {
1173                 cte_add(lineno, line);
1174                 break;
1175             }
1176             if (tz == NULL || strcmp(&line[cursor], get_obj(tz))) {
1177                 rel_shared(tz);
1178                 tz = create_shared_str(&line[cursor]);
1179             }
1180             continue;
1181         }
1182
1183         if (strncmp(&line[cursor], ENV_HOME,
1184         strlen(ENV_HOME)) == 0) {
1185             if ((tmp = strchr(&line[cursor], '\n')) != NULL) {
1186                 *tmp = NULL;
1187             }
1188             if (home == NULL ||
1189             strcmp(&line[cursor], get_obj(home))) {
1190                 rel_shared(home);
1191                 home = create_shared_str(
1192                 &line[cursor + strlen(ENV_HOME)]);
1193             }
1194             continue;
1195         }
1196     }

```

```

1198     if (strcmp(&line[cursor], ENV_SHELL,
1199             strlen(ENV_SHELL)) == 0) {
1200         if ((tmp = strchr(&line[cursor], '\n')) != NULL) {
1201             *tmp = NULL;
1202         }
1203         if (shell == NULL ||
1204             strcmp(&line[cursor], get_obj(shell))) {
1205             rel_shared(shell);
1206             shell = create_shared_str(&line[cursor]);
1207         }
1208         continue;
1209     }
1210
1211     e = xmalloc(sizeof (struct event));
1212     e->etype = CRONEVENT;
1213     if (!((e->of.ct.minute = next_field(0, 59)) != NULL) &&
1214         ((e->of.ct.hour = next_field(0, 23)) != NULL) &&
1215         ((e->of.ct.daymon = next_field(1, 31)) != NULL) &&
1216         ((e->of.ct.month = next_field(1, 12)) != NULL) &&
1217         ((e->of.ct.dayweek = next_field(0, 6)) != NULL)) {
1218         free(e);
1219         cte_add(lineno, line);
1220         continue;
1221     }
1222     while (line[cursor] == ' ' || line[cursor] == '\t')
1223         cursor++;
1224     if (line[cursor] == '\n' || line[cursor] == '\0')
1225         continue;
1226     /* get the command to execute */
1227     start = cursor;
1228 again:
1229     while ((line[cursor] != '%' &&
1230           (line[cursor] != '\n') &&
1231           (line[cursor] != '\0') &&
1232           (line[cursor] != '\\'))
1233         cursor++;
1234     if (line[cursor] == '\\') {
1235         cursor += 2;
1236         goto again;
1237     }
1238     e->cmd = xmalloc(cursor-start + 1);
1239     (void) strncpy(e->cmd, line + start, cursor-start);
1240     e->cmd[cursor-start] = '\0';
1241     /* see if there is any standard input */
1242     if (line[cursor] == '%') {
1243         e->of.ct.input = xmalloc(strlen(line)-cursor + 1);
1244         (void) strcpy(e->of.ct.input, line + cursor + 1);
1245         for (i = 0; i < strlen(e->of.ct.input); i++) {
1246             if (e->of.ct.input[i] == '%')
1247                 e->of.ct.input[i] = '\n';
1248         }
1249     } else {
1250         e->of.ct.input = NULL;
1251     }
1252     /* set the timezone of this entry */
1253     e->of.ct.tz = dup_shared(tz);
1254     /* set the shell of this entry */
1255     e->of.ct.shell = dup_shared(shell);
1256     /* set the home of this entry */
1257     e->of.ct.home = dup_shared(home);
1258     /* have the event point to it's owner */
1259     e->u = u;
1260     /* insert this event at the front of this user's event list */
1261     e->link = u->ctevents;
1262     u->ctevents = e;
1263     /* set the time for the first occurrence of this event */

```

```

1264     e->time = next_time(e, reftime);
1265     /* finally, add this event to the main event list */
1266     switch (el_add(e, e->time, u->ctid)) {
1267     case -1:
1268         ignore_msg("readcron", "cron", e);
1269         break;
1270     case -2: /* event time lower than init time */
1271         reset_needed = 1;
1272         break;
1273     }
1274     cte_valid();
1275 #ifdef DEBUG
1276     cftime(timebuf, "%+", &e->time);
1277     cftime(timebuf, "%C", &e->time);
1278     (void) fprintf(stderr, "inserting cron event %s at %ld (%s)\n",
1279                  e->cmd, e->time, timebuf);
1280 #endif
1281     }
1282     cte_sendmail(u->name); /* mail errors if any to user */
1283     (void) fclose(cf);
1284     rel_shared(tz);
1285     rel_shared(shell);
1286     rel_shared(home);
1287 }

```

unchanged portion omitted

new/usr/src/cmd/fruadm/fruadm.c

1

```
*****
21434 Fri Jan 3 13:06:05 2014
new/usr/src/cmd/fruadm/fruadm.c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23 * Copyright (c) 2014 Gary Mills
24 *
25 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
26 * Use is subject to license terms.
27 */

29 #include <limits.h>
30 #include <stdio.h>
31 #include <stdlib.h>
32 #include <string.h>
33 #include <libintl.h>
34 #include <libfru.h>
35 #include <errno.h>
36 #include <math.h>
37 #include <alloca.h>
38 #include <assert.h>
39 #include <sys/systeminfo.h>

41 #define NUM_OF_SEGMENT 1
42 #define SEGMENT_NAME_SIZE 2

44 #define FD_SEGMENT_SIZE 2949

46 static char *command, *customer_data = NULL, *frupath = NULL, **svccargv;

48 /* DataElement supported in the customer operation */
49 static char *cust_data_list[] = {"Customer_DataR"};

51 /* DataElement supported in the service operation */
52 static char *serv_data_list[] = {"InstallationR", "ECO_CurrentR"};

54 /* currently supported segment name */
55 static char *segment_name[] = {"FD"};

57 static int found_frupath = 0, list_only = 0, recursive = 0,
58 service_mode = 0, svccargc, update = 0;
```

new/usr/src/cmd/fruadm/fruadm.c

2

```
61 static void
62 usage(void)
63 {
64     (void) fprintf(stderr,
65         gettext("Usage: %s [ -l ] [ [ -r ] frupath [ text ] ]\n"),
66         command);
67 }
unchanged_portion_omitted

134 static void
135 displayBinary(unsigned char *data, size_t length, fru_elemdef_t *def)
136 {
137     int i = 0;
138     uint64_t lldata;
139     uint64_t mask;

141     if (def->disp_type == FDISP_Hex) {
142         for (i = 0; i < length; i++) {
143             (void) printf("%02X", data[i]);
144         }
145         return;
146     }

148     (void) memcpy(&lldata, data, sizeof (lldata));
149     switch (def->disp_type) {
150     case FDISP_Binary:
151         {
152             mask = 0x8000000000000000ULL;
153             for (i = 0; i < (sizeof (uint64_t) * 8); i++) {
154                 if (lldata & (mask >> i)) {
155                     (void) printf("1");
156                 } else {
157                     (void) printf("0");
158                 }
159             }
160             return;
161         }
162     case FDISP_Octal:
163         {
164             (void) printf("%llo", lldata);
165             return;
166         }
167     case FDISP_Decimal:
168         {
169             (void) printf("%lld", lldata);
170             return;
171         }
172     case FDISP_Time:
173         {
174             char buffer[PATH_MAX];
175             time_t time;
176             time = (time_t)lldata;
177             (void) strftime(buffer, PATH_MAX, "%+",
178                 (void) strftime(buffer, PATH_MAX, "%C",
179                     localtime(&time)));
179             (void) printf("%s", buffer);
180             return;
181         }
182     }
183 }
unchanged_portion_omitted
```

```

*****
2683 Fri Jan 3 13:06:06 2014
new/usr/src/lib/libc/port/gen/cftime.c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 2014 Gary Mills
24  *
25  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
26  * Use is subject to license terms.
27  */

29 /*      Copyright (c) 1988 AT&T */
30 /*      All Rights Reserved */

30 #pragma ident      "%Z%M% %I%      %E% SMI"

32 /*
33  * This routine converts time as follows. The epoch is 0000 Jan 1
34  * 1970 GMT. The argument time is in seconds since then. The
35  * localtime(t) entry returns a pointer to an array containing:
36  *
37  *          seconds (0-59)
38  *          minutes (0-59)
39  *          hours (0-23)
40  *          day of month (1-31)
41  *          month (0-11)
42  *          year
43  *          weekday (0-6, Sun is 0)
44  *          day of the year
45  *          daylight savings flag
46  *
47  * The routine corrects for daylight saving time and will work in
48  * any time zone provided "timezone" is adjusted to the difference
49  * between Greenwich and local standard time (measured in seconds).
50  *
51  *          ascftime(buf, format, t)      -> where t is produced by localtime
52  *                                         and returns a ptr to a character
53  *                                         string that has the ascii time in
54  *                                         the format specified by the format
55  *                                         argument (see date(1) for format
56  *                                         syntax).
57  *
58  *          cftime(buf, format, t)      -> just calls ascftime.

```

```

59  *
60  *
61  *
62  */

64 #include      "lint.h"
65 #include      <mtlib.h>
66 #include      <stddef.h>
67 #include      <time.h>
68 #include      <limits.h>
69 #include      <stdlib.h>
70 #include      <thread.h>
71 #include      <synch.h>

73 int
74 cftime(char *buf, char *format, const time_t *t)
75 {
76     struct tm res;
77     struct tm *p;

79     p = localtime_r(t, &res);
80     if (p == NULL) {
81         *buf = '\0';
82         return (0);
83     }
84     /* LINTED do not use ascftime() */
85     return (ascftime(buf, format, p));
86 }

88 int
89 ascftime(char *buf, const char *format, const struct tm *tm)
90 {
91     /* Set format string, if not already set */
92     if (format == NULL || *format == '\0')
93         if (((format = getenv("CFTIME")) == 0) || *format == 0)
94             format = "%+";
95             format = "%C";

96     return ((int)strftime(buf, LONG_MAX, format, tm));
97 }

```

unchanged_portion_omitted


```

*****
9318 Fri Jan 3 13:06:06 2014
new/usr/src/lib/libfru/libnvfru/nvfru.c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23 * Copyright (c) 2014 Gary Mills
24 *
25 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
26 */

28 #include <stdio.h>
29 #include <stdlib.h>
30 #include <stdint.h>
31 #include <strings.h>
32 #include <assert.h>
33 #include <pthread.h>
34 #include <sys/byteorder.h>
35 #include <sys/types.h>
36 #include <sys/nvpair.h>

38 #include "libfru.h"
39 #include "libfrup.h"
40 #include "fru_tag.h"
41 #include "libfrureg.h"
42 #include "nvfru.h"

44 #define NUM_ITER_BYTES 4
45 #define HEAD_ITER 0
46 #define TAIL_ITER 1
47 #define NUM_ITER 2
48 #define MAX_ITER 3
49 #define TIMESTRINGLEN 128

51 #define PARSE_TIME 1

53 static pthread_mutex_t gLock = PTHREAD_MUTEX_INITIALIZER;

57 static void
58 convert_field(const uint8_t *field, const fru_regdef_t *def, const char *path,
59 nvlist_t *nv)
60 {

```

```

61 char timestring[TIMESTRINGLEN];
62 int i;
63 uint64_t value;
64 time_t timefield;

66 switch (def->dataType) {
67 case FDTYPE_Binary:
68     assert(def->payloadLen <= sizeof (value));
69     switch (def->dispType) {
70 #if PARSE_TIME == 1
71     case FDISP_Time:
72         if (def->payloadLen > sizeof (timefield)) {
73             /* too big for formatting */
74             return;
75         }
76         (void) memcpy(&timefield, field, sizeof (timefield));
77         timefield = BE_32(timefield);
78         if (strftime(timestring, sizeof (timestring), "%c",
79             localtime(&timefield)) == 0) {
80             /* buffer too small */
81             return;
82         }
83         (void) nvlist_add_string(nv, path, timestring);
84         return;
85 #endif

87     case FDISP_Binary:
88     case FDISP_Octal:
89     case FDISP_Decimal:
90     case FDISP_Hex:
91     default:
92         value = 0;
93         (void) memcpy((((uint8_t *)&value) +
94             sizeof (value) - def->payloadLen),
95             field, def->payloadLen);
96         value = BE_64(value);
97         switch (def->payloadLen) {
98         case 1:
99             (void) nvlist_add_uint8(nv, path,
100                 (uint8_t)value);
101             break;
102         case 2:
103             (void) nvlist_add_uint16(nv, path,
104                 (uint16_t)value);
105             break;
106         case 4:
107             (void) nvlist_add_uint32(nv, path,
108                 (uint32_t)value);
109             break;
110         default:
111             (void) nvlist_add_uint64(nv, path, value);
112         }
113         return;
114     }

116 case FDTYPE_ASCII:
117     (void) nvlist_add_string(nv, path, (char *)field);
118     return;

120 case FDTYPE_Enumeration:
121     value = 0;
122     (void) memcpy((((uint8_t *)&value) + sizeof (value) -
123         def->payloadLen), field, def->payloadLen);
124     value = BE_64(value);
125     for (i = 0; i < def->enumCount; i++) {

```

```
126             if (def->enumTable[i].value == value) {
127                 (void) nvlist_add_string(nv, path,
128                     def->enumTable[i].text);
129                 return;
130             }
131         }
132     }
133
134     /* nothing matched above, use byte array */
135     (void) nvlist_add_byte_array(nv, path, (uchar_t *)field,
136         def->payloadLen);
137 }
unchanged_portion_omitted
```

```

*****
25581 Fri Jan 3 13:06:06 2014
new/usr/src/lib/print/libpapi-common/common/attribute.c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 2014 Gary Mills
24  *
25  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
26  * Use is subject to license terms.
27  *
28  */

30 /* $Id: attribute.c 157 2006-04-26 15:07:55Z ktou $ */

30 #pragma ident      "%Z%M% %I%      %E% SMI"

32 /*LINTLIBRARY*/

34 #include <stdio.h>
35 #include <stdlib.h>
36 #include <stdarg.h>
37 #include <string.h>
38 #include <ctype.h>
39 #include <alloca.h>
40 #include <papi.h>
41 #include <regex.h>

43 #define MAX_PAGES 32767
44 /*
45  * Assuming the maximum number of pages in
46  * a document to be 32767
47  */

49 static void papiAttributeFree(papi_attribute_t *attribute);

51 static void
52 papiAttributeValueFree(papi_attribute_value_type_t type,
53                       papi_attribute_value_t *value)
54 {
55     if (value != NULL) {
56         switch (type) {
57             case PAPI_STRING:
58                 if (value->string != NULL)

```

```

59         free(value->string);
60         break;
61     case PAPI_COLLECTION:
62         if (value->collection != NULL) {
63             int i;

65             for (i = 0; value->collection[i] != NULL; i++)
66                 papiAttributeFree(value->collection[i]);

68             free(value->collection);
69         }
70         break;
71     default: /* don't need to free anything extra */
72         break;
73 }

75     free(value);
76 }
77 }

unchanged_portion_omitted_

889 static papi_status_t
890 papiAttributeToString(papi_attribute_t *attribute, char *delim,
891                      char *buffer, size_t buflen)
892 {
893     papi_attribute_value_t **values = attribute->values;
894     int rc, i;

896     if ((attribute->type == PAPI_BOOLEAN) && (values[1] == NULL)) {
897         if (values[0]->boolean == PAPI_FALSE) {
898             if (isupper(attribute->name[0]) == 0)
899                 strcat(buffer, "no", buflen);
900             else
901                 strcat(buffer, "No", buflen);
902         }
903         rc = strcat(buffer, attribute->name, buflen);
904     } else {
905         strcat(buffer, attribute->name, buflen);
906         rc = strcat(buffer, "=", buflen);
907     }

909     if (values == NULL)
910         return (PAPI_OK);

912     for (i = 0; values[i] != NULL; i++) {
913         switch (attribute->type) {
914             case PAPI_STRING:
915                 rc = strcat(buffer, values[i]->string, buflen);
916                 break;
917             case PAPI_INTEGER: {
918                 char string[24];

920                 snprintf(string, sizeof (string), "%d",
921                          values[i]->integer);
922                 rc = strcat(buffer, string, buflen);
923             }
924             break;
925             case PAPI_BOOLEAN:
926                 if (values[1] != NULL)
927                     rc = strcat(buffer, (values[i]->boolean ?
928                                         "true" : "false"), buflen);
929                 break;
930             case PAPI_RANGE: {
931                 char string[24];

933                 if (values[i]->range.lower == values[i]->range.upper)

```

```
934         snprintf(string, sizeof (string), "%d",
935                  values[i]->range.lower);
936     else
937         snprintf(string, sizeof (string), "%d-%d",
938                  values[i]->range.lower,
939                  values[i]->range.upper);
940     rc = strlcat(buffer, string, buflen);
941     }
942     break;
943 case PAPI_RESOLUTION: {
944     char string[24];
945
946     snprintf(string, sizeof (string), "%dx%ddp%c",
947              values[i]->resolution.xres,
948              values[i]->resolution.yres,
949              (values[i]->resolution.units == PAPI_RES_PER_CM
950               ? 'c' : 'i'));
951     rc = strlcat(buffer, string, buflen);
952     }
953     break;
954 case PAPI_DATETIME: {
955     struct tm *tm = localtime(&values[i]->datetime);
956
957     if (tm != NULL) {
958         char string[64];
959
960         strftime(string, sizeof (string), "%c", tm);
960         strftime(string, sizeof (string), "%C", tm);
961         rc = strlcat(buffer, string, buflen);
962     }}
963     break;
964 case PAPI_COLLECTION: {
965     char *string = alloca(buflen);
966
967     papiAttributeListToString(values[i]->collection,
968                               delim, string, buflen);
969     rc = strlcat(buffer, string, buflen);
970     }
971     break;
972 default: {
973     char string[32];
974
975     snprintf(string, sizeof (string), "unknown-type-0x%x",
976              attribute->type);
977     rc = strlcat(buffer, string, buflen);
978     }
979     }
980     if (values[i+1] != NULL)
981         rc = strlcat(buffer, ",", buflen);
982
983     if (rc >= buflen)
984         return (PAPI_NOT_POSSIBLE);
985
986 }
987
988     return (PAPI_OK);
989 }
```

unchanged portion omitted

```

*****
15758 Fri Jan 3 13:06:06 2014
new/usr/src/man/man3c/strftime.3c
4378 Clean up %C in *time() functions
438 need documentation for strftime %s flag
*****
1 \" te
2.\" Copyright (c) 2014 Gary Mills
3.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
4.\" Copyright 1989 AT&T
5.\" Portions Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.
6.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7.\" http://www.opengroup.org/bookstore/.
8.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
9.\" This notice shall appear on any product containing this material.
10.\" The contents of this file are subject to the terms of the Common Development
11.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
12.\" When distributing Covered Code, include this CDDL HEADER in each file and in
13.\"TH STRFTIME 3C \"Jan 3, 2014\"
12.\"TH STRFTIME 3C \"Sep 5, 2006\"
14.SH NAME
15.strftime, cftime, ascftime \- convert date and time to string
16.SH SYNOPSIS
17.LP
18.nf
19#include <time.h>

21 \fBsize_t\fR \fBstrftime\fR(\fBchar *restrict\fR \fIs\fR, \fBsize_t\fR \fImaxsiz
22 \fBconst char *restrict\fR \fIformat\fR,
23 \fBconst struct tm *restrict\fR \fItimeptr\fR);
24.fi

26.LP
27.nf
28 \fBint\fR \fBcftime\fR(\fBchar *fR\fIs\fR, \fBchar *fR\fIformat\fR, \fBconst t
29.fi

31.LP
32.nf
33 \fBint\fR \fBascftime\fR(\fBchar *fR\fIs\fR, \fBconst char *fR\fIformat\fR,
34 \fBconst struct tm *fR\fItimeptr\fR);
35.fi

37.SH DESCRIPTION
38.sp
39.LP
40 The \fBstrftime()\fR, \fBascftime()\fR, and \fBcftime()\fR functions place
41 bytes into the array pointed to by \fIs\fR as controlled by the string pointed
42 to by \fIformat\fR. The \fIformat\fR string consists of zero or more conversion
43 specifications and ordinary characters. A conversion specification consists of
44 a '\fB%\fR' (percent) character and one or two terminating conversion
45 characters that determine the conversion specification's behavior. All
46 ordinary characters (including the terminating null byte) are copied unchanged
47 into the array pointed to by \fIs\fR. If copying takes place between objects
48 that overlap, the behavior is undefined. For \fBstrftime()\fR, no more than
49 \fImaxsize\fR bytes are placed into the array.
50.sp
51.LP
52 If \fIformat\fR is \fB(char *)0\fR, then the locale's default format is used.
53 For \fBstrftime()\fR the default format is the same as \fB%c\fR; for
54 \fBcftime()\fR and \fBascftime()\fR the default format is the same as \fB%+fR.
55 \fBcftime()\fR and \fBascftime()\fR the default format is the same as \fB%C\fR.
56 \fBcftime()\fR and \fBascftime()\fR first try to use the value of the
56 environment variable \fBCFTIME\fR, and if that is undefined or empty, the
57 default format is used.
58.sp

```

```

59.LP
60 Each conversion specification is replaced by appropriate characters as
61 described in the following list. The appropriate characters are determined by
62 the \fBLC_TIME\fR category of the program's locale and by the values contained
63 in the structure pointed to by \fItimeptr\fR for \fBstrftime()\fR and
64 \fBascftime()\fR, and by the time represented by \fIclock\fR for
65 \fBcftime()\fR.
66.sp
67.ne 2
68.na
69 \fB\fB%\fR\fR
70.ad
71.RS 6n
72 Same as \fB%\fR.
73.RE

75.sp
76.ne 2
77.na
78 \fB\fB%A\fR\fR
79.ad
80.RS 6n
81 Locale's abbreviated weekday name.
82.RE

84.sp
85.ne 2
86.na
87 \fB\fB%A\fR\fR
88.ad
89.RS 6n
90 Locale's full weekday name.
91.RE

93.sp
94.ne 2
95.na
96 \fB\fB%b\fR\fR
97.ad
98.RS 6n
99 Locale's abbreviated month name.
100.RE

102.sp
103.ne 2
104.na
105 \fB\fB%B\fR\fR
106.ad
107.RS 6n
108 Locale's full month name.
109.RE

110.SS "Default"
111.sp
112.ne 2
113.na
114 \fB\fB%c\fR\fR
115.ad
116.RS 6n
117 Locale's appropriate date and time representation.
118 In the C locale, this format is:
117 Locale's appropriate date and time represented as:
119.sp
120.in +2
121.nf
121 %a %b %d %H:%M:%S %Y

```

```

122 .fi
123 .in -2

125 This is the default behavior as well as standard-conforming behavior for
126 standards first supported by releases prior to Solaris 2.4. See
127 \fBstandards\fR(5).
128 .RE

130 .SS "Standard conforming"
131 .sp
132 .ne 2
133 .na
134 \fB\fB%c\fR\fR
135 .ad
136 .RS 6n
137 Locale's appropriate date and time represented as:
138 .sp
139 .in +2
140 .nf
122 %a %b %e %H:%M:%S %Y
123 .fi
124 .in -2

145 This is standard-conforming behavior for standards first supported by Solaris
146 2.4 through Solaris 10.
147 .RE

149 .SS "Default"
125 .sp
126 Other locales may have different locale-specific formats.
151 .ne 2
152 .na
153 \fB\fB%C\fR\fR
154 .ad
155 .RS 6n
156 Locale's date and time representation as produced by \fBdate\fR(1).
157 .sp
158 This is the default behavior as well as standard-conforming behavior for
159 standards first supported by releases prior to Solaris 2.4.
127 .RE

162 .SS "Standard conforming"
129 .sp
130 .ne 2
131 .na
132 \fB\fB%C\fR\fR
133 .ad
134 .RS 6n
135 Century number (the year divided by 100 and truncated to an integer as a
136 decimal number [01,99]).
171 .sp
172 This is standard-conforming behavior for standards first supported by Solaris
173 2.4 through Solaris 10.
137 .RE

139 .sp
140 .ne 2
141 .na
142 \fB\fB%d\fR\fR
143 .ad
144 .RS 6n
145 Day of month [01,31].
146 .RE

148 .sp
149 .ne 2

```

```

150 .na
151 \fB\fB%D\fR\fR
152 .ad
153 .RS 6n
154 Date as \fB%m\fR/\fB%d\fR/\fB%y\fR.
155 .RE

157 .sp
158 .ne 2
159 .na
160 \fB\fB%e\fR\fR
161 .ad
162 .RS 6n
163 Day of month [1,31]; single digits are preceded by a space.
164 .RE

166 .sp
167 .ne 2
168 .na
169 \fB\fB%F\fR\fR
170 .ad
171 .RS 6n
172 Equivalent to \fB%Y\fR-\fB%m\fR-\fB%d\fR (the ISO 8601:2000 standard date
173 format).
174 .RE

176 .sp
177 .ne 2
178 .na
179 \fB\fB%g\fR\fR
180 .ad
181 .RS 6n
182 Week-based year within century [00,99].
183 .RE

185 .sp
186 .ne 2
187 .na
188 \fB\fB%G\fR\fR
189 .ad
190 .RS 6n
191 Week-based year, including the century [0000,9999].
192 .RE

194 .sp
195 .ne 2
196 .na
197 \fB\fB%h\fR\fR
198 .ad
199 .RS 6n
200 Locale's abbreviated month name.
201 .RE

203 .sp
204 .ne 2
205 .na
206 \fB\fB%H\fR\fR
207 .ad
208 .RS 6n
209 Hour (24-hour clock) [00,23].
210 .RE

212 .sp
213 .ne 2
214 .na
215 \fB\fB%I\fR\fR

```

```

216 .ad
217 .RS 6n
218 Hour (12-hour clock) [01,12].
219 .RE

221 .sp
222 .ne 2
223 .na
224 \fB\fB%j\fR\fR
225 .ad
226 .RS 6n
227 Day number of year [001,366].
228 .RE

230 .sp
231 .ne 2
232 .na
233 \fB\fB%k\fR\fR
234 .ad
235 .RS 6n
236 Hour (24-hour clock) [0,23]; single digits are preceded by a space.
237 .RE

239 .sp
240 .ne 2
241 .na
242 \fB\fB%l\fR\fR
243 .ad
244 .RS 6n
245 Hour (12-hour clock) [1,12]; single digits are preceded by a space.
246 .RE

248 .sp
249 .ne 2
250 .na
251 \fB\fB%m\fR\fR
252 .ad
253 .RS 6n
254 Month number [01,12].
255 .RE

257 .sp
258 .ne 2
259 .na
260 \fB\fB%M\fR\fR
261 .ad
262 .RS 6n
263 Minute [00,59].
264 .RE

266 .sp
267 .ne 2
268 .na
269 \fB\fB%n\fR\fR
270 .ad
271 .RS 6n
272 Insert a NEWLINE.
273 .RE

275 .sp
276 .ne 2
277 .na
278 \fB\fB%p\fR\fR
279 .ad
280 .RS 6n
281 Locale's equivalent of either a.m. or p.m.

```

```

282 .RE

284 .sp
285 .ne 2
286 .na
287 \fB\fB%r\fR\fR
288 .ad
289 .RS 6n
290 Appropriate time representation in 12-hour clock format with \fB%p\fR.
291 .RE

293 .sp
294 .ne 2
295 .na
296 \fB\fB%R\fR\fR
297 .ad
298 .RS 6n
299 Time as \fB%H\fR:\fB%M\fR.
300 .RE

302 .sp
303 .ne 2
304 .na
305 \fB\fB%s\fR\fR
306 .ad
307 .RS 6n
308 Seconds since 00:00:00 UTC, January 1, 1970.
309 .RE

311 .sp
312 .ne 2
313 .na
314 \fB\fB%S\fR\fR
315 .ad
316 .RS 6n
317 Seconds [00,60]; the range of values is [00,60] rather than [00,59] to allow
318 for the occasional leap second.
319 .RE

321 .sp
322 .ne 2
323 .na
324 \fB\fB%t\fR\fR
325 .ad
326 .RS 6n
327 Insert a TAB.
328 .RE

330 .sp
331 .ne 2
332 .na
333 \fB\fB%T\fR\fR
334 .ad
335 .RS 6n
336 Time as \fB%H\fR:\fB%M\fR:\fB%S\fR.
337 .RE

339 .sp
340 .ne 2
341 .na
342 \fB\fB%u\fR\fR
343 .ad
344 .RS 6n
345 Weekday as a decimal number [1,7], with 1 representing Monday. See \fBNOTES\fR
346 below.
347 .RE

```

```

349 .sp
350 .ne 2
351 .na
352 \fB\fB%U\fR\fR
353 .ad
354 .RS 6n
355 Week number of year as a decimal number [00,53], with Sunday as the first day
356 of week 1.
357 .RE

359 .sp
360 .ne 2
361 .na
362 \fB\fB%v\fR\fR
363 .ad
364 .RS 6n
365 Date as \fB%e\fR-\fB%b\fR-\fB%Y\fR.
366 .RE

368 .sp
369 .ne 2
370 .na
371 \fB\fB%V\fR\fR
372 .ad
373 .RS 6n
374 The ISO 8601 week number as a decimal number [01,53]. In the ISO 8601
375 week-based system, weeks begin on a Monday and week 1 of the year is the week
376 that includes both January 4th and the first Thursday of the year. If the
377 first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of
378 the last week of the preceding year. See \fBNOTES\fR below.
379 .RE

381 .sp
382 .ne 2
383 .na
384 \fB\fB%w\fR\fR
385 .ad
386 .RS 6n
387 Weekday as a decimal number [0,6], with 0 representing Sunday.
388 .RE

390 .sp
391 .ne 2
392 .na
393 \fB\fB%W\fR\fR
394 .ad
395 .RS 6n
396 Week number of year as a decimal number [00,53], with Monday as the first day
397 of week 1.
398 .RE

400 .sp
401 .ne 2
402 .na
403 \fB\fB%x\fR\fR
404 .ad
405 .RS 6n
406 Locale's appropriate date representation.
407 .RE

409 .sp
410 .ne 2
411 .na
412 \fB\fB%X\fR\fR
413 .ad

```

```

414 .RS 6n
415 Locale's appropriate time representation.
416 .RE

418 .sp
419 .ne 2
420 .na
421 \fB\fB%y\fR\fR
422 .ad
423 .RS 6n
424 Year within century [00,99].
425 .RE

427 .sp
428 .ne 2
429 .na
430 \fB\fB%Y\fR\fR
431 .ad
432 .RS 6n
433 Year, including the century (for example 1993).
434 .RE

436 .sp
437 .ne 2
438 .na
439 \fB\fB%z\fR\fR
440 .ad
441 .RS 6n
442 Replaced by offset from UTC in ISO 8601:2000 standard format (\fB+h:mm\fR or
443 \fB-h:mm\fR), or by no characters if no time zone is determinable. For example,
444 "-0430" means 4 hours 30 minutes behind UTC (west of Greenwich). If
445 \fB%tm_isdst\fR is zero, the standard time offset is used. If \fB%tm_isdst\fR is
446 greater than zero, the daylight savings time offset is used. If \fB%tm_isdst\fR
447 is negative, no characters are returned.
448 .RE

450 .sp
451 .ne 2
452 .na
453 \fB\fB%Z\fR\fR
454 .ad
455 .RS 6n
456 Time zone name or abbreviation, or no bytes if no time zone information exists.
457 .RE

459 .sp
460 .ne 2
461 .na
462 \fB\fB%+ \fR\fR
463 .ad
464 .RS 6n
465 Locale's date and time representation as produced by \fBdate\fR(1).
466 .RE

468 .sp
469 .LP
470 If a conversion specification does not correspond to any of the above or to any
471 of the modified conversion specifications listed below, the behavior is
472 undefined and \fB% \fR is returned.
473 .sp
474 .LP
475 The difference between \fB%U\fR and \fB%W\fR (and also between modified
476 conversion specifications \fB%OU\fR and \fB%OW\fR) lies in which day is counted
477 as the first of the week. Week number 1 is the first week in January starting
478 with a Sunday for \fB%U\fR or a Monday for \fB%W\fR. Week number 0 contains
479 those days before the first Sunday or Monday in January for \fB%U\fR and

```



```

480 \fB%W\fR, respectively.
481 .SS "Modified Conversion Specifications"
482 .sp
483 .LP
484 Some conversion specifications can be modified by the \fBE\fR and \fBO\fR
485 modifiers to indicate that an alternate format or specification should be used
486 rather than the one normally used by the unmodified conversion specification.
487 If the alternate format or specification does not exist in the current locale,
488 the behavior will be as if the unmodified specification were used.
489 .sp
490 .ne 2
491 .na
492 \fB\fB%Ec\fR\fR
493 .ad
494 .RS 7n
495 Locale's alternate appropriate date and time representation.
496 .RE

498 .sp
499 .ne 2
500 .na
501 \fB\fB%EC\fR\fR
502 .ad
503 .RS 7n
504 Name of the base year (period) in the locale's alternate representation.
505 .RE

507 .sp
508 .ne 2
509 .na
510 \fB\fB%Eg\fR\fR
511 .ad
512 .RS 7n
513 Offset from \fB%EC\fR of the week-based year in the locale's alternative
514 representation.
515 .RE

517 .sp
518 .ne 2
519 .na
520 \fB\fB%EG\fR\fR
521 .ad
522 .RS 7n
523 Full alternative representation of the week-based year.
524 .RE

526 .sp
527 .ne 2
528 .na
529 \fB\fB%Ex\fR\fR
530 .ad
531 .RS 7n
532 Locale's alternate date representation.
533 .RE

535 .sp
536 .ne 2
537 .na
538 \fB\fB%EX\fR\fR
539 .ad
540 .RS 7n
541 Locale's alternate time representation.
542 .RE

544 .sp
545 .ne 2

```

```

546 .na
547 \fB\fB%Ey\fR\fR
548 .ad
549 .RS 7n
550 Offset from \fB%EC\fR (year only) in the locale's alternate representation.
551 .RE

553 .sp
554 .ne 2
555 .na
556 \fB\fB%EY\fR\fR
557 .ad
558 .RS 7n
559 Full alternate year representation.
560 .RE

562 .sp
563 .ne 2
564 .na
565 \fB\fB%Od\fR\fR
566 .ad
567 .RS 7n
568 Day of the month using the locale's alternate numeric symbols.
569 .RE

571 .sp
572 .ne 2
573 .na
574 \fB\fB%Oe\fR\fR
575 .ad
576 .RS 7n
577 Same as \fB%Od\fR.
578 .RE

580 .sp
581 .ne 2
582 .na
583 \fB\fB%Og\fR\fR
584 .ad
585 .RS 7n
586 Week-based year (offset from \fB%C\fR) in the locale's alternate representation
587 and using the locale's alternate numeric symbols.
588 .RE

590 .sp
591 .ne 2
592 .na
593 \fB\fB%OH\fR\fR
594 .ad
595 .RS 7n
596 Hour (24-hour clock) using the locale's alternate numeric symbols.
597 .RE

599 .sp
600 .ne 2
601 .na
602 \fB\fB%OI\fR\fR
603 .ad
604 .RS 7n
605 Hour (12-hour clock) using the locale's alternate numeric symbols.
606 .RE

608 .sp
609 .ne 2
610 .na
611 \fB\fB%Om\fR\fR

```

```

612 .ad
613 .RS 7n
614 Month using the locale's alternate numeric symbols.
615 .RE

617 .sp
618 .ne 2
619 .na
620 \fB\fB%OM\fR\fR
621 .ad
622 .RS 7n
623 Minutes using the locale's alternate numeric symbols.
624 .RE

626 .sp
627 .ne 2
628 .na
629 \fB\fB%OS\fR\fR
630 .ad
631 .RS 7n
632 Seconds using the locale's alternate numeric symbols.
633 .RE

635 .sp
636 .ne 2
637 .na
638 \fB\fB%Ou\fR\fR
639 .ad
640 .RS 7n
641 Weekday as a number in the locale's alternate numeric symbols.
642 .RE

644 .sp
645 .ne 2
646 .na
647 \fB\fB%OU\fR\fR
648 .ad
649 .RS 7n
650 Week number of the year (Sunday as the first day of the week) using the
651 locale's alternate numeric symbols.
652 .RE

654 .sp
655 .ne 2
656 .na
657 \fB\fB%Ow\fR\fR
658 .ad
659 .RS 7n
660 Number of the weekday (Sunday=0) using the locale's alternate numeric symbols.
661 .RE

663 .sp
664 .ne 2
665 .na
666 \fB\fB%OW\fR\fR
667 .ad
668 .RS 7n
669 Week number of the year (Monday as the first day of the week) using the
670 locale's alternate numeric symbols.
671 .RE

673 .sp
674 .ne 2
675 .na
676 \fB\fB%Oy\fR\fR
677 .ad

```

```

678 .RS 7n
679 Year (offset from \fB%\fR) in the locale's alternate representation and using
680 the locale's alternate numeric symbols.
681 .RE

683 .SS "Selecting the Output Language"
684 .sp
685 .LP
686 By default, the output of \fBstrftime()\fR, \fBcftime()\fR, and
687 \fBascftime()\fR appear in U.S. English. The user can request that the output
688 of \fBstrftime()\fR, \fBcftime()\fR, or \fBascftime()\fR be in a specific
689 language by setting the \fBLC_TIME\fR category using \fBsetlocale()\fR.
690 .SS "Time Zone"
691 .sp
692 .LP
693 Local time zone information is used as though \fBtzset\fR(3C) were called.
694 .SH RETURN VALUES
695 .sp
696 .LP
697 The \fBstrftime()\fR, \fBcftime()\fR, and \fBascftime()\fR functions return the
698 number of characters placed into the array pointed to by \fRis\fR, not including
699 the terminating null character. If the total number of resulting characters
700 including the terminating null character is more than \fBImaxsize\fR,
701 \fBstrftime()\fR returns \fB0\fR and the contents of the array are
702 indeterminate.
703 .SH EXAMPLES
704 .LP
705 \fBExample 1\fR An example of the \fBstrftime()\fR function.
706 .sp
707 .LP
708 The following example illustrates the use of \fBstrftime()\fR for the
709 \fBPOSIX\fR locale. It shows what the string in \fBistr\fR would look like if
710 the structure pointed to by \fBtmpr\fR contains the values corresponding to
711 Thursday, August 28, 1986 at 12:44:36.

713 .sp
714 .in +2
715 .nf
716 \fBstrftime (str, strsize, "%A %b %d %j", tmpr)\fR
717 .fi
718 .in -2

720 .sp
721 .LP
722 This results in \fBistr\fR containing "Thursday Aug 28 240".

724 .SH ATTRIBUTES
725 .sp
726 .LP
727 See \fBattributes\fR(5) for descriptions of the following attributes:
728 .sp

730 .sp
731 .TS
732 box;
733 c | c
734 l | l .
735 ATTRIBUTE TYPE ATTRIBUTE VALUE
736 -
737 CSI Enabled
738 -
739 Interface Stability Committed
740 -
741 MT-Level MT-Safe
742 -
743 Standard See below.

```

744 .TE

746 .sp
747 .LP
748 For \fBstrftime()\fR, see \fBstandards\fR(5).
749 .SH SEE ALSO
750 .sp
751 .LP
752 \fBdate\fR(1), \fBctime\fR(3C), \fBmktime\fR(3C), \fBsetlocale\fR(3C),
753 \fBstrptime\fR(3C), \fBtzset\fR(3C), \fBTIMEZONE\fR(4), \fBzoneinfo\fR(4),
754 \fBattributes\fR(5), \fBenviron\fR(5), \fBstandards\fR(5)

755 .SH NOTES
756 .sp
757 .LP
758 The conversion specification for \fB%V\fR was changed in the Solaris 7 release.
759 This change was based on the public review draft of the ISO C9x standard at
760 that time. Previously, the specification stated that if the week containing 1
761 January had fewer than four days in the new year, it became week 53 of the
762 previous year. The ISO C9x standard committee subsequently recognized that that
763 specification had been incorrect.

764 .sp
765 .LP
766 The conversion specifications for \fB%g\fR, \fB%G\fR, \fB%Eg\fR, \fB%EG\fR, and
767 \fB%Og\fR were added in the Solaris 7 release. This change was based on the
768 public review draft of the ISO C9x standard at that time. These specifications
769 are evolving. If the ISO C9x standard is finalized with a different
770 conclusion, these specifications will change to conform to the ISO C9x standard
771 decision.

772 .sp
773 .LP
774 The conversion specification for \fB%u\fR was changed in the Solaris 8 release.
775 This change was based on the XPG4 specification.

776 .sp
777 .LP
778 If using the \fB%Z\fR specifier and \fBzoneinfo\fR timezones and if the input
779 date is outside the range 20:45:52 UTC, December 13, 1901 to 03:14:07 UTC,
780 January 19, 2038, the timezone name may not be correct.

781 .sp
782 .LP
783 **The conversion specification for \fB%+\fR was added in illumos.**
784 **It is not part of any standard, although it is available on a number**
785 **of other platforms.**
786 **Its use is discouraged for conforming applications.**