```
**********************************************************
    8528 Sat Feb 10 09:35:49 2018
new/usr/src/cmd/rtc/rtc.c
8980 BIOS clock is sometimes one hour fast
Reviewed by: Toomas Soome <tsoome@me.com>
Reviewed by: C Fraire <cfraire@me.com>
**********************************************************
```

```
/*
 * 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 2018 Gary Mills
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/sysi86.h>
#include <errno.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>

/* RTC modes */
#define M_UNSET 0 /* Mode never set */
#define M_VAR   1 /* Tracks local time including DST */
#define M_UTC   2 /* Clock runs in UTC */
#define M_STD   3 /* Clock runs in local standard time */

static char *progname;
static char *zonefile = "/etc/rtc_config";
static FILE *zonefptr;
static char zone_info[256];
static char zone_lag[256];
static char tz[256] = "TZ=";
static char *utc_zone = "UTC";
int debug = 0;
int rtc_mode = M_UNSET;
int lag;
int errors_ok = 0; /* allow "rtc no-args" to be quiet when not configured */
static time_t clock_val;
static char zone_comment[] =
        "#\n"
```



```
        "#      This file (%s) contains information used to manage the\n"
        "#      x86 real time clock hardware.  The hardware is kept in\n"
        "#      the machine's local time for compatibility with other x86\n"
        "#      operating systems.  This file is read by the kernel at\n"
        "#      boot time.  It is set and updated by the /usr/sbin/rtc\n"
        "#      command.  The 'zone_info' field designates the local\n"
        "#      time zone.  The 'zone_lag' field indicates the number\n"
        "#      of seconds between local time and Greenwich Mean Time.\n"
        "#\n";

/*
 *      Open the configuration file and extract the
 *      zone_info and the zone_lag.  Return 0 if successful.
 */
int
open_zonefile()
{
        char b[256], *s;
        int lag_hrs;

        if ((zonefptr = fopen(zonefile, "r")) == NULL) {
                if (errors_ok == 0)
                        (void) fprintf(stderr,
                            "%s: cannot open %s: errno = %d\n",
                            progname, zonefile, errno);
                return (1);
        }

        for (;;) {
                if ((s = fgets(b, sizeof (b), zonefptr)) == NULL)
                        break;
                if ((s = strchr(s, 'z')) == NULL)
                        continue;
                if (strncmp(s, "zone_info", 9) == 0) {
                        s += 9;
                        while (*s != 0 && *s != '=')
                                s++;
                        if (*s == '=') {
                                s++;
                                while (*s != 0 && (*s == ' ' || *s == '\t'))
                                        s++;
                                (void) strncpy(zone_info, s,
                                    sizeof (zone_info));
                                s = zone_info;
                                while (*s != 0 && *s != '\n')
                                        s++;
                                if (*s == '\n')
                                        *s = 0;
                        }
                } else if (strncmp(s, "zone_lag", 8) == 0) {
                        s += 8;
                        while (*s != 0 && *s != '=')
                                s++;
                        if (*s == '=') {
                                s++;
                                while (*s != 0 && (*s == ' ' || *s == '\t'))
                                        s++;
                                (void) strncpy(zone_lag, s, sizeof (zone_lag));
                                s = zone_lag;
                                while (*s != 0 && *s != '\n')
                                        s++;
                                if (*s == '\n')
                                        *s = 0;
                        }
                }
        }
```

```
124         lag = atoi(zone_lag);
125         lag_hrs = lag / 3600;
126         if (zone_info[0] == 0) {
127                 (void) fprintf(stderr, "%s: zone_info field is invalid\n",
128                     progname);
129                 zone_info[0] = 0;
130                 zone_lag[0] = 0;
131                 return (1);
132         }
133         if (zone_lag[0] == 0) {
134                 (void) fprintf(stderr, "%s: zone_lag field is invalid\n",
135                     progname);
136                 zone_lag[0] = 0;
137                 return (1);
138         }
139         if ((lag_hrs < -24) || (lag_hrs > 24)) {
140                 (void) fprintf(stderr, "%s: a GMT lag of %d is out of range\n",
141                     progname, lag_hrs);
142                 zone_info[0] = 0;
143                 zone_lag[0] = 0;
144                 return (1);
145         }
146         if (debug)
147                 (void) fprintf(stderr, "zone_info = %s,   zone_lag = %s\n",
148                     zone_info, zone_lag);
149         if (debug)
150                 (void) fprintf(stderr, "lag (decimal) is %d\n", lag);

152         (void) fclose(zonefptr);
153         zonefptr = NULL;
154         return (0);
155 }
```

*unchanged_portion_omitted_*

```
166 int
167 get_local(char *z)
159 long
160 set_zone(char *zone_string)
168 {
169         struct tm *tm;
163         long current_lag;

165         (void) umask(0022);
166         if ((zonefptr = fopen(zonefile, "w")) == NULL) {
167                 (void) fprintf(stderr, "%s: cannot open %s: errno = %d\n",
168                         progname, zonefile, errno);
169                 return (0);
170         }

171         tz[3] = 0;
172         (void) strncat(tz, z, 253);
173         (void) strncat(tz, zone_string, 253);
173         if (debug)
174                 (void) fprintf(stderr, "Time Zone string is '%s'\n", tz);

176         (void) putenv(tz);
177         if (debug)
178                 (void) system("env | grep TZ");

180         (void) time(&clock_val);

182         tm = localtime(&clock_val);
183         return (tm->tm_isdst);
184 }

186 long
```



```
187 set_zone(char *zone_string)
188 {
189         int isdst;
190         long current_lag;

192         (void) umask(0022);
193         if ((zonefptr = fopen(zonefile, "w")) == NULL) {
194                 (void) fprintf(stderr, "%s: cannot open %s: errno = %d\n",
195                     progname, zonefile, errno);
196                 return (0);
197         }

199         switch (rtc_mode) {
200         case M_VAR:
201                 isdst = get_local(zone_string);
202                 current_lag = isdst ? altzone : timezone;
203                 break;
204         case M_STD:
205                 isdst = get_local(zone_string);
206                 current_lag = timezone;
207                 break;
208         default:        /* Includes M_UTC */
209                 isdst = 0;
210                 current_lag = 0;
211                 zone_string = utc_zone;
212                 break;
213         }
184         current_lag = tm->tm_isdst ? altzone : timezone;
214         if (debug)
215                 (void) printf("%s DST.    Lag is %ld.\n", isdst ? "Is" :
216                     "Is NOT",  current_lag);
186                 (void) printf("%s DST.    Lag is %ld.\n", tm->tm_isdst ? "Is" :
187                     "Is NOT",  tm->tm_isdst ? altzone : timezone);

218         (void) fprintf(zonefptr, zone_comment, zonefile);
219         (void) fprintf(zonefptr, "zone_info=%s\n", zone_string);
220         (void) fprintf(zonefptr, "zone_lag=%ld\n", current_lag);
191         (void) fprintf(zonefptr, "zone_lag=%ld\n",
192             tm->tm_isdst ? altzone : timezone);
221         (void) fclose(zonefptr);
222         zonefptr = NULL;
223         return (current_lag);
224 }

226 void
227 correct_rtc_and_lag()
228 {
229         int isdst;
201         struct tm *tm;
230         long kernels_lag;
231         long current_lag;

233         if (open_zonefile())
234                 return;

236         switch (rtc_mode) {
237         case M_VAR:
238                 isdst = get_local(zone_info);
239                 current_lag = isdst ? altzone : timezone;
240                 break;
241         case M_STD:
242                 (void) get_local(zone_info);
243                 current_lag = timezone;
244                 break;
245         default:        /* Includes M_UTC */
246                 current_lag = 0;
```

```
247                     break;
248             }
208             tz[3] = 0;
209             (void) strncat(tz, zone_info, 253);
210             if (debug)
211                     (void) fprintf(stderr, "Time Zone string is '%s'\n", tz);

213             (void) putenv(tz);
214             if (debug)
215                     (void) system("env | grep TZ");

217             (void) time(&clock_val);
218             tm = localtime(&clock_val);
219             current_lag = tm->tm_isdst ? altzone : timezone;

250             if (current_lag != lag) {       /* if file is wrong */
251                     if (debug)
252                             (void) fprintf(stderr, "correcting file\n");
223                             (void) fprintf(stderr, "correcting file");
253                     (void) set_zone(zone_info);     /* then rewrite file */
254             }

256             (void) sysi86(GGMTL, &kernels_lag);
257             if (current_lag != kernels_lag) {
258                     if (debug)
259                             (void) fprintf(stderr, "correcting kernel's lag\n");
230                             (void) fprintf(stderr, "correcting kernel's lag");
260                     (void) sysi86(SGMTL, current_lag);      /* correct the lag */
261                     (void) sysi86(WTODC);                   /* set the rtc to */
262                                                             /* new local time */
263             }
264 }
```

*unchanged_portion_omitted_*

```
285 void
286 usage()
287 {
288         static char Usage[] = "Usage:\n\
289 rtc [-w] [-s|-u|-v] [-c] [-z time_zone] [-?]\n";
260 rtc [-c] [-z time_zone] [-?]\n";

291         (void) fprintf(stderr, Usage);
292 }

294 void
295 verbose_usage()
296 {
297         static char Usage1[] = "\
298         Options:\n\
299             -w\t\tDoes nothing.\n\
300             -s\t\tRTC runs in local standard time.\n\
301             -u\t\tRTC runs in UTC time.\n\
302             -v\t\tRTC tracks local time (with cron command).\n\
303             -c\t\tCheck and correct for daylight savings time rollover.\n\
304             -z [zone]\tRecord the zone info in the config file.\n";

306         (void) fprintf(stderr, Usage1);
307 }

309 void
310 set_default()
311 {
312         switch (rtc_mode) {
313         default:        /* Includes M_UNSET */
314                 rtc_mode = M_VAR;
315                 break;
```



```
316         case M_VAR:
317                 /*FALLTHROUGH*/
318         case M_UTC:
319                 /*FALLTHROUGH*/
320         case M_STD:
321                 break;
322         }
323 }

325 void
326 check_mode(int letter, int mode)
327 {
328         if (rtc_mode == M_UNSET || rtc_mode == mode) {
329                 rtc_mode = mode;
330                 return;
331         }
332         (void) fprintf(stderr, "%s: option -%c conflicts with other options\n",
333             progname, letter);
334         exit(1);
335 }


338 int
339 main(int argc, char *argv[])
340 {
341         int c;
342         int cflg = 0;
343         char *zone_name = NULL;

345         progname = argv[0];

347         if (argc == 1) {
348                 errors_ok = 1;
349                 display_zone_string();
350                 exit(0);
351         }

353         while ((c = getopt(argc, argv, "suvwcz:d")) != EOF) {
288         while ((c = getopt(argc, argv, "cz:d")) != EOF) {
354                 switch (c) {
355                 case 'c':
356                         cflg++;
291                         correct_rtc_and_lag();
357                         continue;
358                 case 'z':
359                         zone_name = optarg;
294                         initialize_zone(optarg);
360                         continue;
361                 case 'd':
362                         debug = 1;
363                         continue;
364                 case 's':       /* standard: RTC runs local standard time */
365                         check_mode(c, M_STD);
366                         continue;
367                 case 'u':       /* utc: RTC runs UTC time */
368                         check_mode(c, M_UTC);
369                         continue;
370                 case 'v':       /* varies: RTC tracks local time */
371                         check_mode(c, M_VAR);
372                         continue;
373                 case 'w':       /* Does nothing */
374                         continue;
375                 case '?':
376                         verbose_usage();
377                         exit(0);
301                         return (0);
```

```
 378                 default:
 379                         usage();
 380                         exit(1);
 304                         return (1);
 381                 }
 382         }
 383         set_default();
 384         if (zone_name != NULL)
 385                 initialize_zone(zone_name);
 386         if (cflg > 0)
 387                 correct_rtc_and_lag();
 388         exit(0);
 389         /*LINTED*/
 307         return (0);
 390 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    3796 Sat Feb 10 09:35:49 2018
new/usr/src/man/man1m/rtc.1m
8980 BIOS clock is sometimes one hour fast
Reviewed by: Toomas Soome <tsoome@me.com>
Reviewed by: C Fraire <cfraire@me.com>
**********************************************************
```

```
.\"
.\" This file and its contents are supplied under the terms of the
.\" Common Development and Distribution License ("CDDL"), version 1.0.
.\" You may only use this file in accordance with the terms of version
.\" 1.0 of the CDDL.
.\"
.\" A full copy of the text of the CDDL should have accompanied this
.\" source.  A copy of the CDDL is also available via the Internet at
.\" http://www.illumos.org/license/CDDL.
.\"
.\"
.\" Copyright 2018 Gary Mills
'\" te
.\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
.\"
.Dd January 31, 2018
.Dt RTC 1M
.Os
.Sh NAME
.Nm rtc
.Nd provide all real-time clock and UTC-lag management
.Sh SYNOPSIS
.Nm
.Op Fl csuvw
.Op Fl z Ar zone-name
.Sh DESCRIPTION
The Real Time Clock (RTC) is the hardware device on x86 computers that maintains
the date and time.
The RTC is battery-powered, so that it keeps running when the computer is shut
down.
It can be set from the BIOS and also from the operating system running on the
computer.
The RTC has no setting for the time zone or for Daylight Saving Time (DST).
It relies on the operating system for these facilities and for automatic changes
between standard time and DST.
.Pp
On x86 systems, the
.Nm
command reconciles the difference in the way that time is established between
UNIX and Windows systems.
The internal clock on UNIX systems utilizes Universal Coordinated Time (UTC)
while Windows systems usually expect the RTC to run in local time, including DST
changes.
.Pp
Without arguments,
.Nm
displays the currently configured time zone string for the RTC.
The currently configured time zone string is based on what was last recorded by
.Nm Fl z Ar zone-name .
.Pp
The
.Nm
command is not normally run from a shell prompt; it is generally invoked by the
system.
Commands such as
.Xr date 1
and
.Xr rdate 1M ,
which are used to set the time on a system, invoke
```



```
.Nm Fl c
to ensure that daylight savings time (DST) is corrected for properly.
.Sh OPTIONS
.Bl -tag -width Ds
.It Fl c
This option checks for DST and makes corrections to the RTC if necessary.
It is normally run once a day by a
.Xr cron 1M
job.
.Pp
If there is no RTC time zone or
.Pa /etc/rtc_config
file, this option will do nothing.
.It Fl s
This option specifies that the RTC runs in local standard time all year round.
It is incompatible with Windows, but is convenient if only one operating system
is to be run on the computer.
The
.Xr cron 1M
command is not necessary, and should not be run.
.It Fl u
This option specifies that the RTC runs in UTC time.
As a side effect, it sets the time zone in
.Pa /etc/rtc_config
to UTC.
Windows can operate in UTC time, but requires a registry change to do so.
The
.Xr cron 1M
command is not necessary.
.It Fl v
This option specifies that the RTC tracks local time, including DST changes.
This is the default.
It accomodates Windows with no changes.
The
.Xr cron 1M
command is necessary to change the RTC when DST is in effect.
.It Fl w
This option does nothing.
It is present for compatibility with Solaris 11.
.It Fl z Ar zone-name
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH RTC 1M "Oct 3, 2003"
.SH NAME
rtc \- provide all real-time clock and GMT-lag management
.SH SYNOPSIS
.LP
.nf
\fB/usr/sbin/rtc\fR [\fB-c\fR] [\fB-z\fR \fIzone-name\fR]
.fi

.SH DESCRIPTION
.sp
.LP
On x86 systems, the \fBrtc\fR command reconciles the difference in the way that
time is established between UNIX and MS-DOS systems. UNIX systems utilize
Greenwich Mean Time (\fBGMT\fR), while \fBMS-DOS\fR systems utilize local time.
.sp
.LP
Without arguments, \fBrtc\fR displays the currently configured time zone
string. The currently configured time zone string is based on what was last
recorded by \fBrtc\fR\fB-z\fR \fIzone-name\fR.
.sp
.LP
The \fBrtc\fR command is not normally run from a shell prompt; it is generally
```

```
invoked by the system. Commands such as \fBdate\fR(1) and \fBrdate\fR(1M),
which are used to set the time on a system, invoke \fB/usr/sbin/rtc\fR \fB-c\fR
to ensure that daylight savings time (\fBDST\fR) is corrected for properly.
.SH OPTIONS
.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 16n
This option checks for \fBDST\fR and makes corrections if necessary. It is
normally run once a day by a \fBcron\fR job.
.sp
If there is no \fBRTC\fR time zone or \fB/etc/rtc_config\fR file, this option
will do nothing.
.RE

.sp
.ne 2
.na
\fB\fB\fR\fB-z\fR\fB \fR\fIzone-name\fR\fR
.ad
.RS 16n
This option, which is normally run by the system at software installation time,
is used to specify the time zone in which the RTC is to be maintained.
It updates the configuration file
.Pa /etc/rtc_config
with the name of the specified zone and the current UTC lag for that zone.
If there is an existing
.Pa /etc/rtc_config
file, this command will update it.
If not, this command will create it.
.El
.Sh FILES
.Bl -tag -width "/etc/rtc_config"
.It Pa /etc/rtc_config
The data file used to record the time zone and UTC lag.
This file is completely managed by
.Nm .
At boot time, the kernel reads the UTC lag from this file, and uses it to set
the system time.
.El
.Sh ARCHITECTURE
.Sy x86
.Sh SEE ALSO
.Xr date 1 ,
.Xr cron 1M ,
.Xr rdate 1M ,
.Xr attributes 5
is used to specify the time zone in which the \fBRTC\fR is to be maintained. It
updates the configuration file \fB/etc/rtc_config\fR with the name of the
specified zone and the current \fBGMT\fR lag for that zone. If there is an
existing \fBrtc_config\fR file, this command will update it. If not, this
command will create it.
.RE

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/rtc_config\fR\fR
.ad
.RS 19n
The data file used to record the time zone and \fBGMT\fR lag. This file is
completely managed by \fB/usr/sbin/rtc\fR, and it is read by the kernel.
.RE
```



```
.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Architecture    x86
.TE

.SH SEE ALSO
.sp
.LP
\fBdate\fR(1), \fBrdate\fR(1M), \fBattributes\fR(5)
```