```
**********************************************************
   24242 Sat Mar 15 11:39:23 2014
new/usr/src/cmd/bnu/uucico.c
4337 eliminate /etc/TIMEZONE
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (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 2014 Garrett D'Amore
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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



/*

 * uucp file transfer program:
 * to place a call to a remote machine, login, and
 * copy files between the two machines.

 */
/*
 * Added check to limit the total number of uucicos as defined
 * in the Limits file.
 *
 * Added -f flag to "force execution", ignoring the limit on the
 * number of uucicos. This will be used when invoking uucico from
 * Uutry.
 */

#include "uucp.h"
#include "log.h"

#ifndef V7
#include <sys/mkdev.h>
#endif /* V7 */

#ifdef TLI
#include        <sys/tiuser.h>
#endif /* TLI */

jmp_buf Sjbuf;
extern unsigned msgtime;
```



```
char    uuxqtarg[MAXBASENAME] = {'\0'};
int     uuxqtflag = 0;

extern int      (*Setup)(), (*Teardown)();      /* defined in interface.c */

#define USAGE   "Usage: %s [-x NUM] [-r [0|1]] -s SYSTEM -u USERID -d SPOOL -i I
extern void closedem();
void cleanup(), cleanTM();

extern int sysaccess(), guinfo(), eaccess(), countProcs(), interface(),
        savline(), omsg(), restline(), imsg(), callok(), gnxseq(),
        cmtseq(), conn(), startup(), cntrl();
extern void setuucp(), fixline(), gename(), ulkseq(), pfEndfile();

#ifdef  NOSTRANGERS
static void checkrmt();         /* See if we want to talk to remote. */
#endif /* NOSTRANGERS */

extern char *Mytype;

static char *pskip();

int
main(argc, argv, envp)
int argc;
char *argv[];
char **envp;
{

        extern void intrEXIT(), onintr(), timeout();
        extern void setservice();
#ifndef ATTSVR3
        void setTZ();
#endif /* ATTSVR3 */
        int ret, seq, exitcode;
        char file[NAMESIZE];
        char msg[BUFSIZ], *p, *q;
        char xflag[6];  /* -xN N is single digit */
        char *ttyn;
        char *iface;    /* interface name       */
        char    cb[128];
        time_t  ts, tconv;
        char lockname[MAXFULLNAME];
        struct limits limitval;
        int maxnumb;
        int force = 0;  /* set to force execution, ignoring uucico limit */
        char gradedir[2*NAMESIZE];

        /* Set locale environment variables local definitions */
        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"  /* Use this only if it wasn't */
#endif
        (void) textdomain(TEXT_DOMAIN);

        Ulimit = ulimit(1,0L);
        Uid = getuid();
        Euid = geteuid();       /* this should be UUCPUID */
        if (Uid == 0)
            setuid(UUCPUID);
        Env = envp;
        Role = SLAVE;
        strcpy(Logfile, LOGCICO);
        *Rmtname = NULLCHAR;
        Ifn = Ofn = -1;         /* must be set before signal handlers */
```

```
128         closedem();
129         time(&Nstat.t_qtime);
130         tconv = Nstat.t_start = Nstat.t_qtime;
131         strcpy(Progname, "uucico");
132         setservice(Progname);
133         ret = sysaccess(EACCESS_SYSTEMS);
134         ASSERT(ret == 0, Ct_OPEN, "Systems", ret);
135         ret = sysaccess(EACCESS_DEVICES);
136         ASSERT(ret == 0, Ct_OPEN, "Devices", ret);
137         ret = sysaccess(EACCESS_DIALERS);
138         ASSERT(ret == 0, Ct_OPEN, "Dialers", ret);
139         Pchar = 'C';
140         (void) signal(SIGILL, intrEXIT);
141         (void) signal(SIGTRAP, intrEXIT);
142         (void) signal(SIGIOT, intrEXIT);
143         (void) signal(SIGEMT, intrEXIT);
144         (void) signal(SIGFPE, intrEXIT);
145         (void) signal(SIGBUS, intrEXIT);
146         (void) signal(SIGSEGV, intrEXIT);
147         (void) signal(SIGSYS, intrEXIT);
148         if (signal(SIGPIPE, SIG_IGN) != SIG_IGN)        /* This for sockets */
149                 (void) signal(SIGPIPE, intrEXIT);
150         (void) signal(SIGINT, onintr);
151         (void) signal(SIGHUP, onintr);
152         (void) signal(SIGQUIT, onintr);
153         (void) signal(SIGTERM, onintr);
154 #ifdef SIGUSR1
155         (void) signal(SIGUSR1, SIG_IGN);
156 #endif
157 #ifdef SIGUSR2
158         (void) signal(SIGUSR2, SIG_IGN);
159 #endif
160 #ifdef BSD4_2
161         (void) sigsetmask(sigblock(0) & ~(1 << (SIGALRM - 1)));
162 #endif /*BSD4_2*/

164         pfInit();
165         scInit("xfer");
166         ret = guinfo(Euid, User);
167         ASSERT(ret == 0, "BAD UID ", "", ret);
168         strncpy(Uucp, User, NAMESIZE);

170         setuucp(User);

172         *xflag = NULLCHAR;
173         iface = "UNIX";

175         while ((ret = getopt(argc, argv, "fd:c:r:s:x:u:i:")) != EOF) {
176                 switch (ret) {
177                 case 'd':
178                         if ( eaccess(optarg, 01) != 0 ) {
179                                 (void) fprintf(stderr, gettext("%s: cannot"
180                                     " access spool directory %s\n"),
181                                       Progname, optarg);
182                                 exit(1);
183                         }
184                         Spool = optarg;
185                         break;
186                 case 'c':
187                         Mytype = optarg;
188                         break;
189                 case 'f':
190                         ++force;
191                         break;
192                 case 'r':
193                         if ( (Role = atoi(optarg)) != MASTER && Role != SLAVE )
```



```
194                                 (void) fprintf(stderr, gettext("%s: bad value"
195                                     " '%s' for -r argument\n" USAGE),
196                                       Progname, optarg, Progname);
197                                 exit(1);
198                         }
199                         break;
200                 case 's':
201                         strncpy(Rmtname, optarg, MAXFULLNAME-1);
202                         if (versys(Rmtname)) {
203                             (void) fprintf(stderr,
204                                 gettext("%s: %s not in Systems file\n"),
205                                 Progname, optarg);
206                             cleanup(101);
207                         }
208                         /* set args for possible xuuxqt call */
209                         strcpy(uuxqtarg, Rmtname);
210                         /* if versys put a longer name in, truncate it again */
211                         Rmtname[MAXBASENAME] = '\0';
212                         break;
213                 case 'x':
214                         Debug = atoi(optarg);
215                         if (Debug <= 0)
216                                 Debug = 1;
217                         if (Debug > 9)
218                                 Debug = 9;
219                         (void) sprintf(xflag, "-x%d", Debug);
220                         break;
221                 case 'u':
222                         DEBUG(4, "Loginuser %s specified\n", optarg);
223                         strncpy(Loginuser, optarg, NAMESIZE);
224                         Loginuser[NAMESIZE - 1] = NULLCHAR;
225                         break;
226                 case 'i':
227                         /*      interface type          */
228                         iface = optarg;
229                         break;
230                 default:
231                         (void) fprintf(stderr, gettext(USAGE), Progname);
232                         exit(1);
233                 }
234         }

236         if (Role == MASTER || *Loginuser == NULLCHAR) {
237             ret = guinfo(Uid, Loginuser);
238             ASSERT(ret == 0, "BAD LOGIN_UID ", "", ret);
239         }

241         /* limit the total number of uucicos */
242         if (force) {
243             DEBUG(4, "force flag set (ignoring uucico limit)\n%s", "");
244         } else if (scanlimit("uucico", &limitval) == FAIL) {
245             DEBUG(1, "No limits for uucico in %s\n", LIMITS);
246         } else {
247             maxnumb = limitval.totalmax;
248             if (maxnumb < 0) {
249                 DEBUG(4, "Non-positive limit for uucico in %s\n", LIMITS);
250                 DEBUG(1, "No limits for uucico\n%s", "");
251             } else {
252                 DEBUG(4, "Uucico limit %d -- ", maxnumb);
253                 (void) sprintf(lockname, "%s.", LOCKPRE);
254                 if (countProcs(lockname, (maxnumb-1)) == FALSE) {
255                         DEBUG(4, "exiting\n%s", "");
256                         cleanup(101);
257                 }
258                 DEBUG(4, "continuing\n%s", "");
259             }
```

```
260         }

262         pfStrtConn((Role == MASTER) ? 'M' : 'S');
263         if (Role == MASTER) {
264             if (*Rmtname == NULLCHAR) {
265                 DEBUG(5, "No -s specified\n%s" , "");
266                 cleanup(101);
267             }
268             /* get Myname - it depends on who I'm calling--Rmtname */
269             (void) mchFind(Rmtname);
270             myName(Myname);
271             if (EQUALSN(Rmtname, Myname, MAXBASENAME)) {
272                 DEBUG(5, "This system specified: -sMyname: %s, ", Myname);
273                 cleanup(101);
274             }
275             acInit("xfer");
276         }

278         ASSERT(chdir(Spool) == 0, Ct_CHDIR, Spool, errno);
279         strcpy(Wrkdir, Spool);

281         scReqsys((Role == MASTER) ? Myname : Rmtname); /* log requestor system *

283         if (Role == SLAVE) {

285 #ifndef ATTSVR3
286                 setTZ();
287 #endif /* ATTSVR3 */

289                 if (freopen(RMTDEBUG, "a", stderr) == 0) {
290                         errent(Ct_OPEN, RMTDEBUG, errno, __FILE__, __LINE__);
291                         freopen("/dev/null", "w", stderr);
292                 }
293                 if ( interface(iface) ) {
294                         (void)fprintf(stderr,
295                         "%s: invalid interface %s\n", Progname, iface);
296                         cleanup(101);
297                 }
298                 /*master setup will be called from processdev()*/
299                 if ( (*Setup)( Role, &Ifn, &Ofn ) ) {
300                         DEBUG(5, "SLAVE Setup failed%s", "");
301                         cleanup(101);
302                 }

304                 /*
305                  * initial handshake
306                  */
307                 (void) savline();
308                 fixline(Ifn, 0, D_ACU);
309                 /* get MyName - use logFind to check PERMISSIONS file */
310                 (void) logFind(Loginuser, "");
311                 myName(Myname);

313                 DEBUG(4,"cico.c: Myname - %s\n",Myname);
314                 DEBUG(4,"cico.c: Loginuser - %s\n",Loginuser);
315                 fflush(stderr);
316                 Nstat.t_scall = times(&Nstat.t_tga);
317                 (void) sprintf(msg, "here=%s", Myname);
318                 omsg('S', msg, Ofn);
319                 (void) signal(SIGALRM, timeout);
320                 (void) alarm(msgtime); /* give slow machines a second chance */
321                 if (setjmp(Sjbuf)) {

323                         /*
324                          * timed out
325                          */
```



```
326                         (void) restline();
327                         rmlock(CNULL);
328                         exit(0);
329                 }
330                 for (;;) {
331                         ret = imsg(msg, Ifn);
332                         if (ret != 0) {
333                                 (void) alarm(0);
334                                 (void) restline();
335                                 rmlock(CNULL);
336                                 exit(0);
337                         }
338                         if (msg[0] == 'S')
339                                 break;
340                 }
341                 Nstat.t_ecall = times(&Nstat.t_tga);
342                 (void) alarm(0);
343                 q = &msg[1];
344                 p = pskip(q);
345                 strncpy(Rmtname, q, MAXBASENAME);
346                 Rmtname[MAXBASENAME] = '\0';

348                 seq = 0;
349                 while (p && *p == '-') {
350                         q = pskip(p);
351                         switch(*(++p)) {
352                         case 'x':
353                                 Debug = atoi(++p);
354                                 if (Debug <= 0)
355                                         Debug = 1;
356                                 (void) sprintf(xflag, "-x%d", Debug);
357                                 break;
358                         case 'Q':
359                                 seq = atoi(++p);
360                                 if (seq < 0)
361                                         seq = 0;
362                                 break;
363 #ifdef MAXGRADE
364                         case 'v':       /* version -- -vname=val or -vname */
365                                 if (strncmp(++p, "grade=", 6) == 0 &&
366                                     isalnum(p[6]))
367                                         MaxGrade = p[6];
368                                 break;
369 #endif /* MAXGRADE */
370                         case 'R':
371                                 Restart++;
372                                 p++;
373                                 break;
374                         case 'U':
375                                 SizeCheck++;
376                                 RemUlimit = strtol(++p, (char **) NULL,0);
377                                 break;
378                         default:
379                                 break;
380                         }
381                         p = q;
382                 }
383                 DEBUG(4, "sys-%s\n", Rmtname);
384                 if (strpbrk(Rmtname, Shchar) != NULL) {
385                         DEBUG(4, "Bad remote system name '%s'\n", Rmtname);
386                         logent(Rmtname, "BAD REMOTE SYSTEM NAME");
387                         omsg('R', "Bad remote system name", Ofn);
388                         cleanup(101);
389                 }
390                 if (Restart)
391                     CDEBUG(1,"Checkpoint Restart enabled\n%s", "");
```

```
393 #ifdef NOSTRANGERS
394                 checkrmt();     /* Do we know the remote system. */
395 #else
396                 (void) versys(Rmtname); /* in case the real name is longer */
397 #endif /* NOSTRANGERS */
398
399                 (void) sprintf(lockname, "%ld", (long) getpid());
400                 if (umlock(LOCKPRE, lockname)) {
401                         omsg('R', "LCK", Ofn);
402                         cleanup(101);
403                 }

405                 /* validate login using PERMISSIONS file */
406                 if (logFind(Loginuser, Rmtname) == FAIL) {
407                         scWrite(); /* log security violation */
408                         Uerror = SS_BAD_LOG_MCH;
409                         logent(UERRORTEXT, "FAILED");
410                         systat(Rmtname, SS_BAD_LOG_MCH, UERRORTEXT,
411                             Retrytime);
412                         omsg('R', "LOGIN", Ofn);
413                         cleanup(101);
414                 }

416                 ret = callBack();
417                 DEBUG(4,"return from callcheck: %s",ret ? "TRUE" : "FALSE");
418                 if (ret==TRUE) {
419                         (void) signal(SIGINT, SIG_IGN);
420                         (void) signal(SIGHUP, SIG_IGN);
421                         omsg('R', "CB", Ofn);
422                         logent("CALLBACK", "REQUIRED");
423                         /*
424                          * set up for call back
425                          */
426                         chremdir(Rmtname);
427                         (void) sprintf(file, "%s/%c", Rmtname, D_QUEUE);
428                         chremdir(file);
429                         gename(CMDPRE, Rmtname, 'C', file);
430                         (void) close(creat(file, CFILEMODE));
431                         if (callok(Rmtname) == SS_CALLBACK_LOOP) {
432                             systat(Rmtname, SS_CALLBACK_LOOP, "CALL BACK - LOOP"
433                         } else {
434                             systat(Rmtname, SS_CALLBACK, "CALL BACK", Retrytime)
435                             xuucico(Rmtname);
436                         }
437                         cleanup(101);
438                 }

440                 if (callok(Rmtname) == SS_SEQBAD) {
441                         Uerror = SS_SEQBAD;
442                         logent(UERRORTEXT, "PREVIOUS");
443                         omsg('R', "BADSEQ", Ofn);
444                         cleanup(101);
445                 }

447                 if (gnxseq(Rmtname) == seq) {
448                         if (Restart) {
449                             if (SizeCheck)
450                                 (void) sprintf (msg, "OK -R -U0x%lx %s",
451                                         Ulimit, xflag);
452                             else
453                                 (void) sprintf (msg, "OK -R %s", xflag);
454                             omsg('R', msg, Ofn);
455                         } else
456                             omsg('R', "OK", Ofn);
457                         (void) cmtseq();
```



```
458                 } else {
459                         Uerror = SS_SEQBAD;
460                         systat(Rmtname, SS_SEQBAD, UERRORTEXT, Retrytime);
461                         logent(UERRORTEXT, "HANDSHAKE FAILED");
462                         ulkseq();
463                         omsg('R', "BADSEQ", Ofn);
464                         cleanup(101);
465                 }
466                 ttyn = ttyname(Ifn);
467                 if (ttyn != CNULL && *ttyn != NULLCHAR) {
468                         struct stat ttysbuf;
469                         if ( fstat(Ifn,&ttysbuf) == 0 )
470                                 Dev_mode = ttysbuf.st_mode;
471                         else
472                                 Dev_mode = R_DEVICEMODE;
473                         if ( EQUALSN(ttyn,"/dev/",5) )
474                             strcpy(Dc, ttyn+5);
475                         else
476                             strcpy(Dc, ttyn);
477                         chmod(ttyn, S_DEVICEMODE);
478                 } else
479                         strcpy(Dc, "notty");
480                 /* set args for possible xuuxqt call */
481                 strcpy(uuxqtarg, Rmtname);
482         }

484         strcpy(User, Uucp);
485 /*
486  *  Ensure reasonable ulimit (MINULIMIT)
487  */

489 #ifndef V7
490         {
491         long    minulimit;
492         minulimit = ulimit(1, (long) 0);
493         ASSERT(minulimit >= MINULIMIT, "ULIMIT TOO SMALL",
494             Loginuser, (int) minulimit);
495         }
496 #endif
497         if (Role == MASTER && callok(Rmtname) != 0) {
498                 logent("SYSTEM STATUS", "CAN NOT CALL");
499                 cleanup(101);
500         }

502         chremdir(Rmtname);

504         (void) strcpy(Wrkdir, RemSpool);
505         if (Role == MASTER) {

507                 /*
508                  * master part
509                  */
510                 (void) signal(SIGINT, SIG_IGN);
511                 (void) signal(SIGHUP, SIG_IGN);
512                 (void) signal(SIGQUIT, SIG_IGN);
513                 if (Ifn != -1 && Role == MASTER) {
514                         (void) (*Write)(Ofn, EOTMSG, strlen(EOTMSG));
515                         (void) close(Ofn);
516                         (void) close(Ifn);
517                         Ifn = Ofn = -1;
518                         rmlock(CNULL);
519                         sleep(3);
520                 }

522                 /*
523                  * Find the highest priority job grade that has
```

```
524                  * jobs to do. This is needed to form the lock name.
525                  */
527                 findgrade(RemSpool, JobGrade);
528                 DEBUG(4, "Job grade to process - %s\n", JobGrade);
530                 /*
531                  * Lock the job grade if there is one to process.
532                  */
534                 if (*JobGrade != NULLCHAR) {
535                         (void) sprintf(gradedir, "%s/%s", Rmtname, JobGrade);
536                         chremdir(gradedir);
538                         (void) sprintf(lockname, "%.*s.%s", SYSNSIZE, Rmtname, J
539                         (void) sprintf(msg, "call to %s - process job grade %s "
540                             Rmtname, JobGrade);
541                         if (umlock(LOCKPRE, lockname) != 0) {
542                                 logent(msg, "LOCKED");
543                                 CDEBUG(1, "Currently Talking With %s\n",
544                                     Rmtname);
545                                 cleanup(100);
546                         }
547                 } else {
548                         (void) sprintf(msg, "call to %s - no work", Rmtname);
549                 }
551                 Nstat.t_scall = times(&Nstat.t_tga);
552                 Ofn = Ifn = conn(Rmtname);
553                 Nstat.t_ecall = times(&Nstat.t_tga);
554                 if (Ofn < 0) {
555                         delock(LOCKPRE, lockname);
556                         logent(UERRORTEXT, "CONN FAILED");
557                         systat(Rmtname, Uerror, UERRORTEXT, Retrytime);
558                         cleanup(101);
559                 } else {
560                         logent(msg, "SUCCEEDED");
561                         ttyn = ttyname(Ifn);
562                         if (ttyn != CNULL && *ttyn != NULLCHAR) {
563                                 struct stat ttysbuf;
564                                 if ( fstat(Ifn,&ttysbuf) == 0 )
565                                         Dev_mode = ttysbuf.st_mode;
566                                 else
567                                         Dev_mode = R_DEVICEMODE;
568                                 chmod(ttyn, M_DEVICEMODE);
569                         }
570                 }
571
572                 if (setjmp(Sjbuf)) {
573                         delock(LOCKPRE, lockname);
574                         Uerror = SS_LOGIN_FAILED;
575                         logent(Rmtname, UERRORTEXT);
576                         systat(Rmtname, SS_LOGIN_FAILED,
577                             UERRORTEXT, Retrytime);
578                         DEBUG(4, "%s - failed\n", UERRORTEXT);
579                         cleanup(101);
580                 }
581                 (void) signal(SIGALRM, timeout);
582                 /* give slow guys lots of time to thrash */
583                 (void) alarm(2 * msgtime);
584                 for (;;) {
585                         ret = imsg(msg, Ifn);
586                         if (ret != 0) {
587                                 continue; /* try again */
588                         }
589                         if (msg[0] == 'S')
```



```
590                                 break;
591                 }
592                 (void) alarm(0);
593                 if(EQUALSN("here=", &msg[1], 5)){
594                         /* This may be a problem, we check up to MAXBASENAME
595                          * characters now. The old comment was:
596                          * this is a problem.  We'd like to compare with an
597                          * untruncated Rmtname but we fear incompatability.
598                          * So we'll look at most 6 chars (at most).
599                          */
600                         (void) pskip(&msg[6]);
601                         if (!EQUALSN(&msg[6], Rmtname, MAXBASENAME)) {
602                                 delock(LOCKPRE, lockname);
603                                 Uerror = SS_WRONG_MCH;
604                                 logent(&msg[6], UERRORTEXT);
605                                 systat(Rmtname, SS_WRONG_MCH, UERRORTEXT,
606                                     Retrytime);
607                                 DEBUG(4, "%s - failed\n", UERRORTEXT);
608                                 cleanup(101);
609                         }
610                 }
611                 CDEBUG(1,"Login Successful: System=%s\n",&msg[6]);
612                 seq = gnxseq(Rmtname);
613                 (void) sprintf(msg, "%s -Q%d -R -U0x%lx %s",
614                         Myname, seq, Ulimit, xflag);
615 #ifdef MAXGRADE
616                 if (MaxGrade != NULLCHAR) {
617                         p = strchr(msg, NULLCHAR);
618                         sprintf(p, " -vgrade=%c", MaxGrade);
619                 }
620 #endif /* MAXGRADE */
621                 omsg('S', msg, Ofn);
622                 (void) alarm(msgtime);  /* give slow guys some thrash time */
623                 for (;;) {
624                         ret = imsg(msg, Ifn);
625                         DEBUG(4, "msg-%s\n", msg);
626                         if (ret != 0) {
627                                 (void) alarm(0);
628                                 delock(LOCKPRE, lockname);
629                                 ulkseq();
630                                 cleanup(101);
631                         }
632                         if (msg[0] == 'R')
633                                 break;
634                 }
635                 (void) alarm(0);
637                 /*  check for rejects from remote */
638                 Uerror = 0;
639                 if (EQUALS(&msg[1], "LCK"))
640                         Uerror = SS_RLOCKED;
641                 else if (EQUALS(&msg[1], "LOGIN"))
642                         Uerror = SS_RLOGIN;
643                 else if (EQUALS(&msg[1], "CB"))
644                         Uerror = (callBack() ? SS_CALLBACK_LOOP : SS_CALLBACK);
645                 else if (EQUALS(&msg[1], "You are unknown to me"))
646                         Uerror = SS_RUNKNOWN;
647                 else if (EQUALS(&msg[1], "BADSEQ"))
648                         Uerror = SS_SEQBAD;
649                 else if (!EQUALSN(&msg[1], "OK", 2))
650                         Uerror = SS_UNKNOWN_RESPONSE;
651                 if (Uerror)  {
652                         delock(LOCKPRE, lockname);
653                         systat(Rmtname, Uerror, UERRORTEXT, Retrytime);
654                         logent(UERRORTEXT, "HANDSHAKE FAILED");
655                         CDEBUG(1, "HANDSHAKE FAILED: %s\n", UERRORTEXT);
```

```
656                         ulkseq();
657                         cleanup(101);
658                 }
659                 (void) cmtseq();

661                 /*
662                  * See if we have any additional parameters on the OK
663                  */

665                 if (strlen(&msg[3])) {
666                         p = pskip(&msg[3]);
667                         while (p && *p == '-') {
668                                 q = pskip(p);
669                                 switch(*(++p)) {
670                                 case 'R':
671                                         Restart++;
672                                         p++;
673                                         break;
674                                 case 'U':
675                                         SizeCheck++;
676                                         RemUlimit = strtol(++p, (char **) NULL,
677                                         break;
678                                 case 'x':
679                                         if (!Debug) {
680                                                 Debug = atoi(++p);
681                                                 if (Debug <= 0)
682                                                         Debug = 1;
683                                         }
684                                         break;
685                                 default:
686                                         break;
687                                 }
688                                 p = q;
689                         }
690                 }

692         }
693         DEBUG(4, " Rmtname %s, ", Rmtname);
694         DEBUG(4, " Restart %s, ", (Restart ? "YES" : "NO"));
695         DEBUG(4, "Role %s,  ", Role ? "MASTER" : "SLAVE");
696         DEBUG(4, "Ifn - %d, ", Ifn);
697         DEBUG(4, "Loginuser - %s\n", Loginuser);

699         /* alarm/setjmp added here due to experience with uucico
700          * hanging for hours in imsg().
701          */
702         if (setjmp(Sjbuf)) {
703                 delock(LOCKPRE, lockname);
704                 logent("startup", "TIMEOUT");
705                 DEBUG(4, "%s - timeout\n", "startup");
706                 cleanup(101);
707         }
708         (void) alarm(MAXSTART);
709         ret = startup();
710         (void) alarm(0);

712         if (ret != SUCCESS) {
713                 delock(LOCKPRE, lockname);
714                 logent("startup", "FAILED");
715                 Uerror = SS_STARTUP;
716                 CDEBUG(1, "%s\n", UERRORTEXT);
717                 systat(Rmtname, Uerror, UERRORTEXT, Retrytime);
718                 exitcode = 101;
719         } else {
720                 pfConnected(Rmtname, Dc);
721                 acConnected(Rmtname, Dc);
```



```
722                 logent("startup", "OK");
723                 systat(Rmtname, SS_INPROGRESS, UTEXT(SS_INPROGRESS),Retrytime);
724                 Nstat.t_sftp = times(&Nstat.t_tga);

726                 exitcode = cntrl();
727                 Nstat.t_eftp = times(&Nstat.t_tga);
728                 DEBUG(4, "cntrl - %d\n", exitcode);
729                 (void) signal(SIGINT, SIG_IGN);
730                 (void) signal(SIGHUP, SIG_IGN);
731                 (void) signal(SIGALRM, timeout);

733                 if (exitcode == 0) {
734                         (void) time(&ts);
735                         (void) sprintf(cb, "conversation complete %s %ld",
736                                 Dc, ts - tconv);
737                         logent(cb, "OK");
738                         systat(Rmtname, SS_OK, UTEXT(SS_OK), Retrytime);

740                 } else {
741                         logent("conversation complete", "FAILED");
742                         systat(Rmtname, SS_CONVERSATION,
743                             UTEXT(SS_CONVERSATION), Retrytime);
744                 }
745                 (void) alarm(msgtime);  /* give slow guys some thrash time */
746                 omsg('O', "OOOOO", Ofn);
747                 CDEBUG(4, "send OO %d,", ret);
748                 if (!setjmp(Sjbuf)) {
749                         for (;;) {
750                                 omsg('O', "OOOOO", Ofn);
751                                 ret = imsg(msg, Ifn);
752                                 if (ret != 0)
753                                         break;
754                                 if (msg[0] == 'O')
755                                         break;
756                         }
757                 }
758                 (void) alarm(0);
759         }
760         cleanup(exitcode);
761         /*NOTREACHED*/
762         return (0);
763 }
_____unchanged_portion_omitted_

902 #ifndef ATTSVR3

904 /*
905  *      setTZ()
906  *
907  *      if login "shell" is uucico (i.e., Role == SLAVE), must set
908  *      timezone env variable TZ.  otherwise will default to EST.
909  */

911 #define LINELEN 81

913 void
914 setTZ()
915 {
916         static char     buf[LINELEN], *bp;
917         extern char     *fgets();
918         FILE            *tzfp;
919         extern FILE     *fopen();
920         int             i;
921         extern int      fclose(), strncmp();

923         if ( (tzfp = fopen("/etc/default/init","r")) == (FILE *)NULL )
```

```
920         if ( (tzfp = fopen("/etc/TIMEZONE","r")) == (FILE *)NULL )
924                 return;
925         while ( (bp = fgets(buf,LINELEN,tzfp)) != (char *)NULL ) {
926                 while ( isspace(*bp) )
927                         ++bp;
928                 if ( strncmp(bp, "TZ=", 3) == 0 ) {
929                         for ( i = strlen(bp) - 1; i > 0 && isspace(*(bp+i)); --i
930                                 *(bp+i) = '\0';
931                         putenv(bp);
932                         (void)fclose(tzfp);
933                         return;
934                 }
935         }
936         (void)fclose(tzfp);
937         return;
938 }
```

_____unchanged_portion_omitted_

```
**********************************************************
    1235 Sat Mar 15 11:39:23 2014
new/usr/src/cmd/init/init.dfl
4337 eliminate /etc/TIMEZONE
**********************************************************
```

```
   1 #
   2 # Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
   3 # Use is subject to license terms.
   4 #
   5 # CDDL HEADER START
   6 #
   7 # The contents of this file are subject to the terms of the
   8 # Common Development and Distribution License, Version 1.0 only
   9 # (the "License").  You may not use this file except in compliance
  10 # with the License.
  11 #
  12 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  13 # or http://www.opensolaris.org/os/licensing.
  14 # See the License for the specific language governing permissions
  15 # and limitations under the License.
  16 #
  17 # When distributing Covered Code, include this CDDL HEADER in each
  18 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  19 # If applicable, add the following below this CDDL HEADER, with the
  20 # fields enclosed by brackets "[]" replaced with your own identifying
  21 # information: Portions Copyright [yyyy] [name of copyright owner]
  22 #
  23 # CDDL HEADER END
  24 #
  25 # This file is /etc/default/init.
  26 # This file looks like a shell script, but it is not.
  25 #ident  "%Z%%M% %I%     %E% SMI"
  27 #
  27 # This file is /etc/default/init.  /etc/TIMEZONE is a symlink to this file.
  28 # This file looks like a shell script, but it is not.  To maintain
  29 # compatibility with old versions of /etc/TIMEZONE, some shell constructs
  30 # (i.e., export commands) are allowed in this file, but are ignored.
  31 #
  28 # Lines of this file should be of the form VAR=value, where VAR is one of
  29 # TZ, LANG, CMASK, or any of the LC_* environment variables.  value may
  30 # be enclosed in double quotes (") or single quotes (').
  31 #
  32 TZ=PST8PDT
  33 CMASK=022
```

```
**********************************************************
   48464 Sat Mar 15 11:39:23 2014
new/usr/src/cmd/listen/listen.c
4337 eliminate /etc/TIMEZONE
**********************************************************
```

```
/*
 * 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 2014 Garrett D'Amore
 */
/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

/*
 * Network Listener Process
 *
 *              command line:
 *
 *              listen [ -m minor_prefix ] netspec
 *
 */

/* system include files */

#include <fcntl.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <memory.h>
#include <sys/utsname.h>
#include <sys/tiuser.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mkdev.h>
#include <values.h>
#include <ctype.h>
```



```
#include <pwd.h>
#include <grp.h>
#include <sys/ipc.h>
#include <sys/poll.h>
#include <sys/stropts.h>
#include <sac.h>
#include <utmpx.h>

/* listener include files */

#include "lsparam.h"            /* listener parameters          */
#include "lsfiles.h"            /* listener files info          */
#include "lserror.h"            /* listener error codes         */
#include "lsnlsmsg.h"           /* NLPS listener protocol       */
#include "lssmbmsg.h"           /* MS_NET identifier            */
#include "lsdbf.h"              /* data base file stuff         */
#include "listen.h"

/* defines      */

#define NAMESIZE        (NAMEBUFSZ-1)

#define SPLhi()         Splflag = 1
#define SPLlo()         Splflag = 0

#define GEN     1
#define LOGIN   0

/* global variables     */

int     NLPS_proc = 0;  /* set if process is a listener child           */
pid_t   Pid;            /* listener's process ID                        */
char    *Progname;      /* listener's basename (from argv[0])           */
static  char Provbuf[PATHSIZE];
char    *Provider = Provbuf;    /* name of transport provider           */
char    *Netspec = NETSPEC;
char    *Minor_prefix;          /* prefix for minor device names        */
int     Dbf_entries;            /* number of private addresses in dbf file*/
int     Valid_addrs;            /* number of addresses bound            */
struct  pollfd *Pollfds;        /* for polling fds                      */
dbf_t   *Dbfhead;               /* Beginning of in-memory database      */
dbf_t   *Newdbf;                /* Beginning of in-memory database (reread) */
char    *Server_cmd_lines;      /* database space                       */
char    *New_cmd_lines;         /* database space (reread)              */
long    Ndesc;                  /* Number of per-process file descriptors */
int     Readdb;                 /* set to TRUE by SAC_READDB message    */
struct  netconfig *Netconf;     /* netconfig structure for this network */

struct  call_list       Free_call;
struct  call_list       *Free_call_p = &Free_call; /* call free list    */
struct  call_list       *Priv_call;     /* call save pending list       */

/* FILE DESCRIPTOR MANAGEMENT:
 *
 * The listener uses 6 (sometimes 7) file descriptors:
 *      fd 0:   Originally opened to /dev/null, used to accept incoming calls.
 *      fd 1:   In the parent, a connection to _sacpipe.  Closed in the child
 *              and dup'ed to 0.
 *      fd 2:   In the parent, a connection to _pmpipe.  Dup'ed in the child
 *              to 0.
 *      fd 3:   Originally opened to /dev/null, this file descriptor is
 *              reserved to open the STREAMS pipe when passing the connection
 *              to a standing server.
 *      fd 4:   Opened to the pid file.  We have to keep it open to keep the
 *              lock active.
 *      fd 5:   Opened to the log file.
```

```
 126  *      fd 6:   Opened to the debug file ONLY when compiled with DEBUGMODE.
 127  *
 128  * The remaining file descriptors are available for binding private addresses.
 129  */

 131 #ifndef DEBUGMODE
 132 #define USEDFDS 6
 133 #else
 134 #define USEDFDS 7
 135 FILE    *Debugfp;               /* for the debugging file       */
 136 #endif

 138 int     Acceptfd;               /* to accept connections (fd 0) */
 139 int     Sacpipefd;              /* pipe TO sac process (fd 1)   */
 140 int     Pmpipefd;               /* pipe FROM sac process (fd 2) */
 141 int     Passfd;                 /* pipe used to pass FD (fd 3)  */
 142 int     Pidfd;                  /* locked pid file (fd 4)       */
 143 FILE    *Logfp;                 /* for logging listener activity*/

 145 struct  pmmsg   Pmmsg;          /* to respond to SAC            */
 146 int     State = PM_STARTING;    /* current SAC state            */
 147 char    Mytag[15];

 149 char    Lastmsg[BUFSIZ];        /* contains last msg logged (by stampbuf) */
 150 int     Logmax = LOGMAX;        /* number of entriet to allow in logfile  */

 152 int     Splflag;                /* logfile critical region flag           */

 154 static char *badnspmsg = "Bad netspec on command line ( Pathname too long )";
 155 static char *badstart  = "Listener failed to start properly";
 156 static char *nologfile = "Unable to open listener log file during initialization
 157 static char *usage     = "Usage: listen [ -m minor_prefix ] network_device";
 158 static char *nopmtag   = "Fatal error: Unable to get PMTAG from environment";
 159 static char tzenv[BUFSIZ];

 161 #define TZFILE  "/etc/default/init"
 160 #define TIMEZONE        "/etc/TIMEZONE"
 162 #define TZSTR   "TZ="

 164 void    check_sac_mesg();       /* routine to process messages from sac */
 165 void    rpc_register();         /* routine to register rpc services */
 166 void    rpc_unregister();       /* routine to unregister rpc services */
 167 extern  struct  netconfig       *getnetconfigent();
 168 extern  char    *t_alloc();
 169 extern  void    logexit();
 170 extern  int     t_errno;
 171 extern  int     errno;

 173 #ifndef TRUE
 174 #define TRUE    1
 175 #define FALSE   0
 176 #endif

 178 static void mod_prvaddr(void);
 179 static void pitchcall(struct call_list *pending, struct t_discon *discon);
 180 static void clr_call(struct t_call *call);
 181 static void trycon(struct call_list *phead, int fd);
 182 static void send_dis(struct call_list *phead, int fd);
 183 static void doevent(struct call_list *phead, int fd);
 184 static void listen(void);
 185 static void rst_signals(void);
 186 static void catch_signals(void);
 187 static void net_open(void);
 188 static void init_files(void);
 189 static void pid_open(void);
```



```
 191 int
 192 main(int argc, char **argv)
 193 {
 194         struct stat buf;
 195         int ret;
 196         char scratch[BUFSIZ];
 197         char log[BUFSIZ];
 198         char olog[BUFSIZ];
 199         char *scratch_p = scratch;
 200         char *mytag_p;
 201         FILE *fp;
 202         extern char *getenv();
 203         char *parse();
 204         int     c;
 205         extern  char *optarg;
 206         extern  int optind;
 207         int i;
 208         char    *Mytag_p = Mytag;

 210         /* Get my port monitor tag out of the environment               */
 211         if ((mytag_p = getenv("PMTAG")) == NULL) {
 212                 /* no place to write */
 213                 exit(1);
 214         }
 215         strcpy(Mytag, mytag_p);

 217         /* open log file */
 218         sprintf(log, "%s/%s/%s", ALTDIR, Mytag_p, LOGNAME);
 219         sprintf(olog, "%s/%s/%s", ALTDIR, Mytag_p, OLOGNAME);
 220         if (stat(log, &buf) == 0) {
 221                 /* file exists, try and save it but if we can't don't worry */
 222                 unlink(olog);
 223                 rename(log, olog);
 224         }
 225         if ((i = open(log, O_WRONLY|O_CREAT|O_APPEND, 0444)) < 0)
 226                 logexit(1, nologfile);
 227         /* as stated above, the log file should be file descriptor 5 */
 228         if ((ret = fcntl(i, F_DUPFD, 5)) != 5)
 229                 logexit(1, nologfile);
 230         Logfp = fdopen(ret, "a+");

 232         /* Get my port monitor tag out of the environment               */
 233         if ((mytag_p = getenv("PMTAG")) == NULL) {
 234                 logexit(1, nopmtag);
 235         }
 236         strcpy(Mytag, mytag_p);

 238         (void) umask(022);
 239         Readdb = FALSE;

 241         if (geteuid() != (uid_t) 0) {
 242                 logmessage("Must be root to start listener");
 243                 logexit(1, badstart);
 244         }

 246         while ((c = getopt(argc, argv, "m:")) != EOF)
 247                 switch (c) {
 248                 case 'm':
 249                         Minor_prefix = optarg;
 250                         break;
 251                 default:
 252                         logexit(1, usage);
 253                         break;
 254                 }

 256         if ((Netspec = argv[optind]) == NULL) {
```

```
257                 logexit(1, usage);
258         }
259         if ((Netconf = getnetconfigent(Netspec)) == NULL) {
260                 sprintf(scratch, "no netconfig entry for <%s>", Netspec);
261                 logmessage(scratch);
262                 logexit(1, badstart);
263         }
264         if (!Minor_prefix)
265                 Minor_prefix = argv[optind];

267         if ((int) strlen(Netspec) > PATHSIZE)  {
268                 logmessage(badnspmsg);
269                 logexit(1, badstart);
270         }

272         /*
273          * SAC will start the listener in the correct directory, so we
274          * don't need to chdir there, as we did in older versions
275          */

277         strcpy(Provbuf, "/dev/");
278         strcat(Provbuf, Netspec);

280         (void) umask(0);

282         init_files();           /* open Accept, Sac, Pm, Pass files     */
283         pid_open();             /* create pid file                      */

285 #ifdef  DEBUGMODE
286         sprintf(scratch, "%s/%s/%s", ALTDIR, Mytag, DBGNAME);
287         Debugfp = fopen(scratch, "w");
288 #endif


291 #ifdef  DEBUGMODE
292         if ((!Logfp) || (!Debugfp))
293 #else
294         if (!Logfp)
295 #endif
296                 logexit(1, badstart);

298 /*
299  * In case we started with no environment, find out what timezone we're
300  * in.  This will get passed to children, so only need to do once.
301  */

303         if (getenv("TZ") == NULL) {
304                 fp = fopen(TZFILE, "r");
303                 fp = fopen(TIMEZONE, "r");
305                 if (fp) {
306                         while (fgets(tzenv, BUFSIZ, fp)) {
307                                 if (tzenv[strlen(tzenv) - 1] == '\n')
308                                         tzenv[strlen(tzenv) - 1] = '\0';
309                                 if (!strncmp(TZSTR, tzenv, strlen(TZSTR))) {
310                                         putenv(parse(tzenv));
311                                         break;
312                                 }
313                         }
314                         fclose(fp);
315                 }
316                 else {
317                         sprintf(scratch, "couldn't open %s, default to GMT",
318                             TZFILE);
316                         sprintf(scratch, "couldn't open %s, default to GMT", TIM
319                         logmessage(scratch);
320                 }
```



```
321         }

323         logmessage("@(#)listen:listen.c 1.19.9.1");

325 #ifdef  DEBUGMODE
326         logmessage("Listener process with DEBUG capability");
327 #endif

329         sprintf(scratch, "Listener port monitor tag: %s", Mytag_p);
330         logmessage(scratch);
331         DEBUG((9, "Minor prefix: %s  Netspec %s", Minor_prefix, Netspec));

333         /* fill in Pmmesg fields that always stay the same */

335         Pmmsg.pm_maxclass = MAXCLASS;
336         strcpy(Pmmsg.pm_tag, Mytag_p);
337         Pmmsg.pm_size = 0;

339         /* Find out what state to start in.  If not in env, exit */
340         if ((scratch_p = getenv("ISTATE")) == NULL)
341                 logexit(1, "ERROR: ISTATE variable not set in environment");
342
343         if (!strcmp(scratch_p, "enabled")) {
344                 State = PM_ENABLED;
345                 logmessage("Starting state: ENABLED");
346         }
347         else {
348                 State = PM_DISABLED;
349                 logmessage("Starting state: DISABLED");
350         }

352         /* try to get my "basename"             */
353         Progname = strrchr(argv[0], '/');
354         if (Progname && Progname[1])
355                 ++Progname;
356         else
357                 Progname = argv[0];

359         catch_signals();

361         /*
362          * Allocate memory for private address and file descriptor table
363          * Here we are assuming that no matter how many private addresses
364          * exist in the system if the system limit is 20 then we will only
365          * get 20 file descriptors
366          */

368         Ndesc = ulimit(4,0L);           /* get num of file des on system */

370         read_dbf(DB_INIT);
371         net_open();                     /* init, open, bind names       */

373         for (i = 3; i < Ndesc; i++)  {  /* leave stdout, stderr open    */
374                 fcntl(i, F_SETFD, 1);   /* set close on exec flag*/
375         }

377         logmessage("Initialization Complete");

379         listen();
380         return (0);
381 }
______unchanged_portion_omitted_


1555 /*
1556  * parse:       Parse TZ= string like init does for consistency
```

```
 *              Work on string in place since result will
 *              either be the same or shorter.
 */

char *
parse(s)
char *s;
{
        char *p;
        char *tp;
        char scratch[BUFSIZ];
        int delim;

        tp = p = s + strlen("TZ=");     /* skip TZ= in parsing */
        if ((*p == '"') || (*p == '\'')) {
                /* it is quoted */
                delim = *p++;
                for (;;) {
                        if (*p == '\0') {
                                /* etc/default/init ill-formed, go without TZ */
                                sprintf(scratch, "%s ill-formed", TZFILE);
                                /* etc/TIMEZONE ill-formed, go without TZ */
                                sprintf(scratch, "%s ill-formed", TIMEZONE);
                                logmessage(scratch);
                                strcpy(s, "TZ=");
                                return(s);
                        }
                        if (*p == delim) {
                                *tp = '\0';
                                return(s);
                        }
                        else {
                                *tp++ = *p++;
                        }
                }
        }
        else { /* look for comment or trailing whitespace */
                for ( ; *p && !isspace(*p) && *p != '#'; ++p)
                        ;
                /* if a comment or trailing whitespace, trash it */
                if (*p) {
                        *p = '\0';
                }
                return(s);
        }
}
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   56576 Sat Mar 15 11:39:23 2014
new/usr/src/cmd/login/login.c
4337 eliminate /etc/TIMEZONE
**********************************************************
______unchanged_portion_omitted_

2152 /*
2153  * establish_user_environment   - Set up the new users enviornment
2154  */

2156 static void
2157 establish_user_environment(char **renvp)
2158 {
2159         int i, j, k, l_index, length, idx = 0;
2160         char *endptr;
2161         char **lenvp;
2162         char **pam_env;

2164         lenvp = environ;
2165         while (*lenvp++)
2166                 ;

2168         /* count the number of PAM environment variables set by modules */
2169         if ((pam_env = pam_getenvlist(pamh)) != 0) {
2170                 for (idx = 0; pam_env[idx] != 0; idx++)
2171                         ;
2172         }

2174         envinit = (char **)calloc(lenvp - environ + 10 + MAXARGS + idx,
2175             sizeof (char *));
2176         if (envinit == NULL) {
2177                 (void) printf("Calloc failed - out of swap space.\n");
2178                 login_exit(8);
2179         }

2181         /*
2182          * add PAM environment variables first so they
2183          * can be overwritten at login's discretion.
2184          * check for illegal environment variables.
2185          */
2186         idx = 0;        basicenv = 0;
2187         if (pam_env != 0) {
2188                 while (pam_env[idx] != 0) {
2189                         if (legalenvvar(pam_env[idx])) {
2190                                 envinit[basicenv] = pam_env[idx];
2191                                 basicenv++;
2192                         }
2193                         idx++;
2194                 }
2195         }
2196         (void) memcpy(&envinit[basicenv], newenv, sizeof (newenv));

2198         /* Set up environment */
2199         if (rflag) {
2200                 ENVSTRNCAT(term, terminal);
2201         } else if (hflag) {
2202                 if (strlen(terminal)) {
2203                         ENVSTRNCAT(term, terminal);
2204                 }
2205         } else {
2206                 char *tp = getenv("TERM");

2208                 if ((tp != NULL) && (*tp != '\0'))
2209                         ENVSTRNCAT(term, tp);
2210         }
```



```
2212         ENVSTRNCAT(logname, pwd->pw_name);

2214         /*
2215          * There are three places to get timezone info.  init.c sets
2216          * TZ if the file /etc/default/init contains a value for TZ.
2216          * TZ if the file /etc/TIMEZONE contains a value for TZ.
2217          * login.c looks in the file /etc/default/login for a
2218          * variable called TIMEZONE being set.  If TIMEZONE has a
2219          *  value, TZ is set to that value; no environment variable
2220          * TIMEZONE is set, only TZ.  If neither of these methods
2221          * work to set TZ, then the library routines  will default
2222          * to using the file /usr/lib/locale/TZ/localtime.
2223          *
2224          * There is a priority set up here.  If /etc/default/init has
2224          * There is a priority set up here.  If /etc/TIMEZONE has
2225          * a value for TZ, that value remains top priority.  If the
2226          * file /etc/default/login has TIMEZONE set, that has second
2227          * highest priority not overriding the value of TZ in
2228          * /etc/default/init.  The reason for this priority is that the
2229          * file /etc/default/init is supposed to be sourced by
2228          * /etc/TIMEZONE.  The reason for this priority is that the
2229          * file /etc/TIMEZONE is supposed to be sourced by
2230          * /etc/profile.  We are doing the "sourcing" prematurely in
2231          * init.c.  Additionally, a login C shell doesn't source the
2232          * file /etc/profile thus not sourcing /etc/default/init thus not
2232          * file /etc/profile thus not sourcing /etc/TIMEZONE thus not
2233          * allowing an adminstrator to globally set TZ for all users
2234          */
2235         if (Def_tz != NULL)     /* Is there a TZ from defaults/login? */
2236                 tmp_tz = Def_tz;

2238         if ((Def_tz = getenv("TZ")) != NULL) {
2239                 ENVSTRNCAT(timez, Def_tz);
2240         } else if (tmp_tz != NULL) {
2241                 Def_tz = tmp_tz;
2242                 ENVSTRNCAT(timez, Def_tz);
2243         }

2245         if (Def_hertz == NULL)
2246                 (void) sprintf(hertz + strlen(hertz), "%lu", HZ);
2247         else
2248                 ENVSTRNCAT(hertz, Def_hertz);

2250         if (Def_path == NULL)
2251                 (void) strlcat(path, DEF_PATH, sizeof (path));
2252         else
2253                 ENVSTRNCAT(path, Def_path);

2255         ENVSTRNCAT(home, pwd->pw_dir);

2257         /*
2258          * Find the end of the basic environment
2259          */
2260         for (basicenv = 0; envinit[basicenv] != NULL; basicenv++)
2261                 ;

2263         /*
2264          * If TZ has a value, add it.
2265          */
2266         if (strcmp(timez, "TZ=") != 0)
2267                 envinit[basicenv++] = timez;

2269         if (*pwd->pw_shell == '\0') {
2270                 /*
2271                  * If possible, use the primary default shell,
```

```
2272                     * otherwise, use the secondary one.
2273                     */
2274                    if (access(SHELL, X_OK) == 0)
2275                            pwd->pw_shell = SHELL;
2276                    else
2277                            pwd->pw_shell = SHELL2;
2278            } else if (Altshell != NULL && strcmp(Altshell, "YES") == 0) {
2279                    envinit[basicenv++] = shell;
2280                    ENVSTRNCAT(shell, pwd->pw_shell);
2281            }

2283 #ifndef NO_MAIL
2284            envinit[basicenv++] = mail;
2285            (void) strlcat(mail, pwd->pw_name, sizeof (mail));
2286 #endif

2288            /*
2289             * Pick up locale environment variables, if any.
2290             */
2291            lenvp = renvp;
2292            while (*lenvp != NULL) {
2293                    j = 0;
2294                    while (localeenv[j] != 0) {
2295                            /*
2296                             * locale_envmatch() returns 1 if
2297                             * *lenvp is localenev[j] and valid.
2298                             */
2299                            if (locale_envmatch(localeenv[j], *lenvp) == 1) {
2300                                    envinit[basicenv++] = *lenvp;
2301                                    break;
2302                            }
2303                            j++;
2304                    }
2305                    lenvp++;
2306            }

2308            /*
2309             * If '-p' flag, then try to pass on allowable environment
2310             * variables.  Note that by processing this first, what is
2311             * passed on the final "login:" line may over-ride the invocation
2312             * values.  XXX is this correct?
2313             */
2314            if (pflag) {
2315                    for (lenvp = renvp; *lenvp; lenvp++) {
2316                            if (!legalenvvar(*lenvp)) {
2317                                    continue;
2318                            }
2319                            /*
2320                             * If this isn't 'xxx=yyy', skip it.  XXX
2321                             */
2322                            if ((endptr = strchr(*lenvp, '=')) == NULL) {
2323                                    continue;
2324                            }
2325                            length = endptr + 1 - *lenvp;
2326                            for (j = 0; j < basicenv; j++) {
2327                                    if (strncmp(envinit[j], *lenvp, length) == 0) {
2328                                            /*
2329                                             * Replace previously established value
2330                                             */
2331                                            envinit[j] = *lenvp;
2332                                            break;
2333                                    }
2334                            }
2335                            if (j == basicenv) {
2336                                    /*
2337                                     * It's a new definition, so add it at the end.
```



```
2338                                     */
2339                                    envinit[basicenv++] = *lenvp;
2340                            }
2341                    }
2342            }

2344            /*
2345             * Add in all the environment variables picked up from the
2346             * argument list to "login" or from the user response to the
2347             * "login" request, if any.
2348             */

2350            if (envp == NULL)
2351                    goto switch_env;        /* done */

2353            for (j = 0, k = 0, l_index = 0;
2354                *envp != NULL && j < (MAXARGS-1);
2355                j++, envp++) {

2357                    /*
2358                     * Scan each string provided.  If it doesn't have the
2359                     * format xxx=yyy, then add the string "Ln=" to the beginning.
2360                     */
2361                    if ((endptr = strchr(*envp, '=')) == NULL) {
2362                            /*
2363                             * This much to be malloc'd:
2364                             *   strlen(*envp) + 1 char for 'L' +
2365                             *   MAXARGSWIDTH + 1 char for '=' + 1 for null char;
2366                             *
2367                             * total = strlen(*envp) + MAXARGSWIDTH + 3
2368                             */
2369                            int total = strlen(*envp) + MAXARGSWIDTH + 3;
2370                            envinit[basicenv+k] = malloc(total);
2371                            if (envinit[basicenv+k] == NULL) {
2372                                    (void) printf("%s: malloc failed\n", PROG_NAME);
2373                                    login_exit(1);
2374                            }
2375                            (void) snprintf(envinit[basicenv+k], total, "L%d=%s",
2376                                l_index, *envp);

2378                            k++;
2379                            l_index++;
2380                    } else  {
2381                            if (!legalenvvar(*envp)) { /* this env var permited? */
2382                                    continue;
2383                            } else {

2385                                    /*
2386                                     * Check to see whether this string replaces
2387                                     * any previously defined string
2388                                     */
2389                                    for (i = 0, length = endptr + 1 - *envp;
2390                                        i < basicenv + k; i++) {
2391                                            if (strncmp(*envp, envinit[i], length)
2392                                                == 0) {
2393                                                    envinit[i] = *envp;
2394                                                    break;
2395                                            }
2396                                    }

2398                                    /*
2399                                     * If it doesn't, place it at the end of
2400                                     * environment array.
2401                                     */
2402                                    if (i == basicenv+k) {
2403                                            envinit[basicenv+k] = *envp;
```

```
2404                                            k++;
2405                                    }
2406                            }
2407                    }
2408            }               /* for (j = 0 ... ) */

2410 switch_env:
2411            /*
2412             * Switch to the new environment.
2413             */
2414            environ = envinit;
2415 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   42631 Sat Mar 15 11:39:23 2014
new/usr/src/cmd/tsol/misc/txzonemgr.sh
4337 eliminate /etc/TIMEZONE
**********************************************************
```

```
#!/bin/ksh
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (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 (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2014 Garrett D'Amore
#
#

# This script provides a simple GUI for managing labeled zones.
# It provides contextual menus which provide appropriate choices.
# It must be run in the global zone as root.

# These arguments are accepted, and will result in non-interactive
# (text-only) mode:
#
#       txzonemgr [-c | -d[f]]
#
#       -c      create default zones
#       -d      destroy all zones; prompts for confirmation unless
#               the -f flag is also specified
#       -f      force
#

# DISP - use GUI (otherwise use non-interactive mode)
DISP=1
# CREATEDEF - make default zones (non-interactive)
CREATEDEF=0
# DESTROYZONES - tear down all zones (non-interactive)
DESTROYZONES=0
# FORCE - force
FORCE=0

NSCD_PER_LABEL=0
NSCD_INDICATOR=/var/tsol/doors/nscd_per_label
if [ -f $NSCD_INDICATOR ] ; then
        NSCD_PER_LABEL=1
fi

myname=$(basename $0)

TXTMP=/tmp/txzonemgr
TNRHTP=/etc/security/tsol/tnrhtp
TNRHDB=/etc/security/tsol/tnrhdb
```



```
TNZONECFG=/etc/security/tsol/tnzonecfg
PUBZONE=public
INTZONE=internal

PATH=/usr/bin:/usr/sbin:/usr/lib export PATH
title="Labeled Zone Manager 2.1"

msg_defzones=$(gettext "Create default zones using default settings?")
msg_confirmkill=$(gettext "OK to destroy all zones?")
msg_continue=$(gettext "(exit to resume $(basename $0) when ready)")
msg_getlabel=$(gettext "Select a label for the")
msg_getremote=$(gettext "Select a remote host or network from the list below:")
msg_getnet=$(gettext "Select a network configuration for the")
msg_getzone=$(gettext "Select a zone from the list below:
(select global for zone creation and shared settings)")
msg_getcmd=$(gettext "Select a command from the list below:")
msg_inuse=$(gettext "That label is already assigned\nto the")
msg_getmin=$(gettext "Select the minimum network label for the")
msg_getmax=$(gettext "Select the maximum network label for the")
msg_badip=$(gettext " is not a valid IP address")


process_options()
{
        typeset opt optlist

        optlist='cdf'

        while getopts ":$optlist" opt
        do
                case $opt in
                c)      CREATEDEF=1
                        DISP=0
                        ;;
                d)      DESTROYZONES=1
                        DISP=0
                        ;;
                f)      FORCE=1
                        ;;
                *)      gettext "invalid option -$OPTARG\n"
                        usage
                        return 2
                        ;;
                esac
        done

        if [ $CREATEDEF -eq 1 -a $DESTROYZONES -eq 1 ] ; then
                gettext "cannot combine options -c and -d\n"
                usage
                return 2
        fi
        if [ $CREATEDEF -eq 1 -a $FORCE -eq 1 ] ; then
                gettext "option -f not allowed with -c\n"
                usage
                return 2
        fi
        if [ $FORCE -eq 1 -a $CREATEDEF -eq 0 -a $DESTROYZONES -eq 0 ] ; then
                gettext "option -f specified without any other options\n"
                usage
                return 2
        fi

        shift $((OPTIND - 1))
        if [ "x$1" != "x" ] ; then
                usage
                return 2
```

```
128         fi

130         return 0
131 }
```

unchanged_portion_omitted_

```
415 initialize() {
416         zonepath=$(zoneadm -z $zonename list -p|cut -d : -f4)
417         ZONE_ETC_DIR=$zonepath/root/etc
418         SYSIDCFG=${ZONE_ETC_DIR}/sysidcfg

420         if [ -f /var/ldap/ldap_client_file ] ; then
421                 ldapaddress=$(ldapclient list | \
422                     grep "^NS_LDAP_SERVERS" | cut -d " " -f2)
423                 print "name_service=LDAP {" > ${SYSIDCFG}
424                 domain=$(domainname)
425                 print "domain_name=$domain" >> ${SYSIDCFG}
426                 profName=$(ldapclient list | \
427                     grep "^NS_LDAP_PROFILE" | cut -d " " -f2)
428                 proxyPwd=$(ldapclient list | \
429                     grep "^NS_LDAP_BINDPASSWD" | cut -d " " -f2)
430                 proxyDN=$(ldapclient list | \
431                     grep "^NS_LDAP_BINDDN" | cut -d " " -f 2)
432                 if [ "$proxyDN" ] ; then
433                         print "proxy_dn=\"$proxyDN\"" >> ${SYSIDCFG}
434                         print "proxy_password=\"$proxyPwd\"" >> ${SYSIDCFG}
435                 fi
436                 print "profile=$profName" >> ${SYSIDCFG}
437                 print "profile_server=$ldapaddress }" >> ${SYSIDCFG}
438                 cp /etc/nsswitch.conf $ZONE_ETC_DIR/nsswitch.ldap
439         else
440                 print "name_service=NONE" > ${SYSIDCFG}
441                 fi
442         print "security_policy=NONE" >> ${SYSIDCFG}
443         locale=$(locale|grep LANG | cut -d "=" -f2)
444         if [[ -z $locale ]] ; then
445                 locale="C"
446         fi
447         print "system_locale=$locale" >> ${SYSIDCFG}
448         timezone=$(grep "^TZ" /etc/default/init|cut -d "=" -f2)
447         timezone=$(grep "^TZ" /etc/TIMEZONE|cut -d "=" -f2)
449         print "timezone=$timezone" >> ${SYSIDCFG}
450         print "terminal=vt100" >> ${SYSIDCFG}
451         rootpwd=$(grep "^root:" /etc/shadow|cut -d : -f2)
452
453 #       There are two problems with setting the root password:
454 #               The zone's shadow file may be read-only
455 #               The password contains unparsable characters
456 #       so the following line is commented out until this is resolved.

458         #print "root_password=$rootpwd" >> ${SYSIDCFG}
459         print "nfs4_domain=dynamic" >> ${SYSIDCFG}
460         print "network_interface=PRIMARY {" >> ${SYSIDCFG}

462         net=$(zonecfg -z $zonename info net)
463         ipType=$(zonecfg -z $zonename info ip-type|cut -d" " -f2)
464         if [ $ipType = exclusive ] ; then
465                 hostname=$(zenity --entry \
466                     --title="$title" \
467                     --width=330 \
468                     --text="${zonename}0: Enter Hostname or dhcp: ")
469                 [ $? != 0 ] && return

471                 if [ $hostname = dhcp ] ; then
472                         print "dhcp" >> ${SYSIDCFG}
```



```
473                 else
474                         print "hostname=$hostname" >> ${SYSIDCFG}
475                         ipaddr=$(getent hosts $hostname|cut -f1)
476                         if [ $? != 0 ] ; then
477                                 ipaddr=$(zenity --entry \
478                                     --title="$title" \
479                                     --text="$nic: Enter IP address: " \
480                                     --entry-text a.b.c.d)
481                                 [ $? != 0 ] && return
482
483                                 validateIPaddr
484                                 if [[ -z $ipaddr ]] ; then
485                                         return
486                                 fi
487                         fi
488                         print "ip_address=$ipaddr" >> ${SYSIDCFG}
489                         getNetmask
490                         print "netmask=$nm" >> ${SYSIDCFG}
491                         print "default_route=none" >> ${SYSIDCFG}
492                         template=${zonename}_cipso
493                         cidr=32
494                         updateTnrhdb
495                 fi
496         elif [[ -n $net ]] ; then
497                 hostname=$(hostname)
498                 hostname=$(zenity --entry \
499                     --title="$title" \
500                     --width=330 \
501                     --text="Enter Hostname: " \
502                     --entry-text $hostname)
503                 [ $? != 0 ] && return
504
505                 print "hostname=$hostname" >> ${SYSIDCFG}
506                 ipaddr=$(getent hosts $hostname|cut -f1)
507                 if [ $? = 0 ] ; then
508                         print "ip_address=$ipaddr" >> ${SYSIDCFG}
509                 fi
510         else
511                 getAllZoneNICs
512                 for i in ${aznics[*]} ; do
513                         ipaddr=$(ifconfig $i|grep inet|cut -d " " -f2)
514                 done
515                 print "hostname=$(hostname)" >> ${SYSIDCFG}
516                 print "ip_address=$ipaddr" >> ${SYSIDCFG}
517         fi
518
519         print "protocol_ipv6=no }" >> ${SYSIDCFG}
520         cp /etc/default/nfs ${ZONE_ETC_DIR}/default/nfs
521         touch ${ZONE_ETC_DIR}/.NFS4inst_state.domain
522 }
```

unchanged_portion_omitted_

```
**********************************************************
    5664 Sat Mar 15 11:39:23 2014
new/usr/src/man/man4/Makefile
4337 eliminate /etc/TIMEZONE
**********************************************************
#
# 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 2011, Richard Lowe
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
#

include         $(SRC)/Makefile.master

MANSECT=        4

_MANFILES=      Intro.4                         \
                NISLDAPmapping.4                \
                TIMEZONE.4                      \
                a.out.4                         \
                admin.4                         \
                alias.4                         \
                aliases.4                       \
                au.4                            \
                audit.log.4                     \
                audit_class.4                   \
                audit_control.4                 \
                audit_event.4                   \
                audit_user.4                    \
                auth_attr.4                     \
                autofs.4                        \
                bart_manifest.4                 \
                bart_rules.4                    \
                bootparams.4                    \
                cardbus.4                       \
                compver.4                       \
                contents.4                      \
                contract.4                      \
                copyright.4                     \
                core.4                          \
                crypt.conf.4                    \
                crypto_certs.4                  \
                d_passwd.4                      \
                dacf.conf.4                     \
                dat.conf.4                      \
                default_fs.4                    \
                defaultdomain.4                 \
                defaultrouter.4                 \
                depend.4                        \
                device_allocate.4               \
                device_contract.4               \
                device_maps.4                   \
                devices.4                       \
                dfstab.4                        \
                dhcp_inittab.4                  \
                dhcp_network.4                  \
                dhcpsvc.conf.4                  \
```



```
                dhcptab.4                       \
                dialups.4                       \
                dir_ufs.4                       \
                driver.conf.4                   \
                ds.log.4                        \
                ethers.4                        \
                exec_attr.4                     \
                fdi.4                           \
                format.dat.4                    \
                fspec.4                         \
                fstypes.4                       \
                ftp.4                           \
                ftpaccess.4                     \
                ftpconversions.4                \
                ftpgroups.4                     \
                ftphosts.4                      \
                ftpservers.4                    \
                ftpusers.4                      \
                fx_dptbl.4                      \
                gateways.4                      \
                group.4                         \
                gsscred.conf.4                  \
                hba.conf.4                      \
                holidays.4                      \
                hosts.4                         \
                hosts.equiv.4                   \
                hosts_access.4                  \
                hosts_options.4                 \
                ib.4                            \
                ike.config.4                    \
                ike.preshared.4                 \
                inet_type.4                     \
                inetd.conf.4                    \
                init.4                          \
                init.d.4                        \
                inittab.4                       \
                ipaddrsel.conf.4                \
                ipnodes.4                       \
                issue.4                         \
                kadm5.acl.4                     \
                kdc.conf.4                      \
                keytables.4                     \
                krb5.conf.4                     \
                ldapfilter.conf.4               \
                ldapsearchprefs.conf.4          \
                ldaptemplates.conf.4            \
                logadm.conf.4                   \
                logindevperm.4                  \
                loginlog.4                      \
                magic.4                         \
                md.tab.4                        \
                mddb.cf.4                       \
                mech.4                          \
                meddb.4                         \
                mnttab.4                        \
                mod_ipp.4                       \
                mpapi.conf.4                    \
                nca.if.4                        \
                ncad_addr.4                     \
                ncakmod.conf.4                  \
                ncalogd.conf.4                  \
                ncaport.conf.4                  \
                ndmp.4                          \
                ndpd.conf.4                     \
                netconfig.4                     \
                netgroup.4                      \
```

```
127                 netid.4                         \
128                 netmasks.4                      \
129                 netrc.4                         \
130                 networks.4                      \
131                 nfs.4                           \
132                 nfslog.conf.4                   \
133                 nfssec.conf.4                   \
134                 nodename.4                      \
135                 nologin.4                       \
136                 note.4                          \
137                 notrouter.4                     \
138                 nscd.conf.4                     \
139                 nsmbrc.4                        \
140                 nss.4                           \
141                 nsswitch.conf.4                 \
142                 packingrules.4                  \
143                 pam.conf.4                      \
144                 passwd.4                        \
145                 path_to_inst.4                  \
146                 pci.4                           \
147                 phones.4                        \
148                 pkginfo.4                       \
149                 pkgmap.4                        \
150                 policy.conf.4                   \
151                 power.conf.4                    \
152                 printers.4                      \
153                 printers.conf.4                 \
154                 priv_names.4                    \
155                 proc.4                          \
156                 process.4                       \
157                 prof_attr.4                     \
158                 profile.4                       \
159                 project.4                       \
160                 protocols.4                     \
161                 prototype.4                     \
162                 pseudo.4                        \
163                 publickey.4                     \
164                 queuedefs.4                     \
165                 rcmscript.4                     \
166                 rdc.cf.4                        \
167                 remote.4                        \
168                 resolv.conf.4                   \
169                 rmtab.4                         \
170                 rpc.4                           \
171                 rt_dptbl.4                      \
172                 sasl_appname.conf.4             \
173                 scsi.4                          \
174                 securenets.4                    \
175                 sel_config.4                    \
176                 sendmail.4                      \
177                 service_bundle.4                \
178                 service_provider.conf.4         \
179                 services.4                      \
180                 shadow.4                        \
181                 sharetab.4                      \
182                 shells.4                        \
183                 slp.conf.4                      \
184                 slpd.reg.4                      \
185                 smb.4                           \
186                 smbautohome.4                   \
187                 smhba.conf.4                    \
188                 sndr.4                          \
189                 sock2path.4                     \
190                 space.4                         \
191                 ssh_config.4                    \
192                 sshd_config.4                   \
```



```
193                 sulog.4                         \
194                 syslog.conf.4                   \
195                 system.4                        \
196                 term.4                          \
197                 terminfo.4                      \
198                 timezone.4                      \
199                 tnf_kernel_probes.4             \
200                 ts_dptbl.4                      \
201                 ttydefs.4                       \
202                 ttysrch.4                       \
203                 ufsdump.4                       \
204                 updaters.4                      \
205                 user_attr.4                     \
206                 utmp.4                          \
207                 utmpx.4                         \
208                 vfstab.4                        \
209                 volume-config.4                 \
210                 volume-request.4                \
211                 wanboot.conf.4                  \
212                 warn.conf.4                     \
213                 xferlog.4                       \
214                 ypfiles.4                       \
215                 yppasswdd.4                     \
216                 ypserv.4                        \
217                 zoneinfo.4

219 sparc_MANFILES= sbus.4

221 i386_MANFILES=  sysbus.4

223 _MANLINKS=      addresses.4                     \
224                 devid_cache.4                   \
225                 devname_cache.4                 \
226                 dir.4                           \
227                 dumpdates.4                     \
228                 fbtab.4                         \
229                 forward.4                       \
230                 fs.4                            \
231                 hosts.allow.4                   \
232                 hosts.deny.4                    \
233                 intro.4                         \
234                 md.cf.4                         \
235                 mdi_ib_cache.4                  \
236                 mdi_scsi_vhci_cache.4           \
237                 pci_unitaddr_persistent.4       \
238                 pcie.4                          \
239                 qop.4                           \
240                 rhosts.4                        \
241                 sendmail.cf.4                   \
242                 snapshot_cache.4                \
243                 submit.cf.4                     \
244                 volume-defaults.4               \
245                 wtmp.4                          \
246                 wtmpx.4

248 i386_MANLINKS=  isa.4

250 MANFILES=       $(_MANFILES) $($(MACH)_MANFILES)
251 MANLINKS=       $(_MANLINKS) $($(MACH)_MANLINKS)

253 intro.4                        := LINKSRC = Intro.4

255 addresses.4                    := LINKSRC = aliases.4
256 forward.4                     := LINKSRC = aliases.4

258 fs.4                           := LINKSRC = default_fs.4
```

```
devid_cache.4                   := LINKSRC = devices.4
devname_cache.4                 := LINKSRC = devices.4
mdi_ib_cache.4                  := LINKSRC = devices.4
mdi_scsi_vhci_cache.4           := LINKSRC = devices.4
pci_unitaddr_persistent.4       := LINKSRC = devices.4
snapshot_cache.4                := LINKSRC = devices.4

dir.4                           := LINKSRC = dir_ufs.4

rhosts.4                        := LINKSRC = hosts.equiv.4

hosts.allow.4                   := LINKSRC = hosts_access.4
hosts.deny.4                    := LINKSRC = hosts_access.4

fbtab.4                         := LINKSRC = logindevperm.4

md.cf.4                         := LINKSRC = md.tab.4

qop.4                           := LINKSRC = mech.4

pcie.4                          := LINKSRC = pci.4

sendmail.cf.4                   := LINKSRC = sendmail.4
submit.cf.4                     := LINKSRC = sendmail.4

isa.4                           := LINKSRC = sysbus.4

dumpdates.4                     := LINKSRC = ufsdump.4

wtmp.4                          := LINKSRC = utmp.4

wtmpx.4                         := LINKSRC = utmpx.4

volume-defaults.4               := LINKSRC = volume-request.4

.KEEP_STATE:

include         $(SRC)/man/Makefile.man

install:        $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
    2500 Sat Mar 15 11:39:23 2014
new/usr/src/man/man4/init.4
4337 eliminate /etc/TIMEZONE
**********************************************************
'\" te
.\" Copyright 2014 Garrett D'Amore
.\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 1989 AT&T
.\" 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 init 4 "Mar 15, 2014"
.TH TIMEZONE 4 "Jun 26, 2003"
.SH NAME
init \- set default system time zone and locale
TIMEZONE \- set default system time zone and locale
.SH SYNOPSIS
.LP
.nf
\fB/etc/TIMEZONE\fR
\fB/etc/default/init\fR
.fi

.SH DESCRIPTION
.sp
.LP
This file sets the time zone environment variable \fBTZ\fR, and the
locale-related environment variables \fBLANG\fR, \fBLC_COLLATE\fR,
\fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, \fBLC_MONETARY\fR, \fBLC_NUMERIC\fR, and
\fBLC_TIME\fR.
.sp
.LP
\fB/etc/TIMEZONE\fR is a symbolic link to \fB/etc/default/init\fR.  This
link exists for compatibility with legacy software, is obsolete, and may
be removed in a future release.
\fB/etc/TIMEZONE\fR is a symbolic link to \fB/etc/default/init\fR.
.sp
.LP
The number of environment variables that can be set from
\fB/etc/default/init\fR is limited to 20.
.sp
.LP
The format of the file is:
.sp
.in +2
.nf
\fIVAR\fR\fB=\fR\fIvalue\fR
.fi
.in -2
.sp

.sp
.LP
where \fIVAR\fR is a timezone environment variable and \fIvalue\fR is the value
assigned to the variable. \fIvalue\fR can be enclosed in double quotes (") or
single quotes (\&'). The double or single quotes cannot be part of the value.
.SH SEE ALSO
.sp
.LP
\fBinit\fR(1M), \fBrtc\fR(1M), \fBctime\fR(3C), \fBenviron\fR(5)
.SH NOTES
.sp
.LP
When changing the \fBTZ\fR setting on x86 systems, you must make a
corresponding change to the \fB/etc/rtc_config\fR file to account for the new
```



```
timezone setting. This can be accomplished by executing the following commands,
followed by a reboot, to make the changes take effect:
.sp
.in +2
.nf
# rtc \fB-z\fR \fIzone-name\fR
# rtc \fB-c\fR

.fi
.in -2
.sp

.sp
.LP
where \fIzone-name\fR is the same name as the \fBTZ\fR variable setting.
.sp
.LP
See \fBrtc\fR(1M) for information on the \fBrtc\fR command.
```

```
**********************************************************
    4854 Sat Mar 15 11:39:24 2014
new/usr/src/pkg/manifests/SUNWcs.man4.inc
4337 eliminate /etc/TIMEZONE
**********************************************************
#
# 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 2011, Richard Lowe
# Copyright 2012 Nexenta Systems, Inc. All rights reserved.
#

file path=usr/share/man/man4/Intro.4
file path=usr/share/man/man4/TIMEZONE.4
file path=usr/share/man/man4/audit.log.4
file path=usr/share/man/man4/audit_class.4
file path=usr/share/man/man4/audit_control.4
file path=usr/share/man/man4/audit_event.4
file path=usr/share/man/man4/audit_user.4
file path=usr/share/man/man4/auth_attr.4
file path=usr/share/man/man4/contract.4
file path=usr/share/man/man4/core.4
file path=usr/share/man/man4/crypt.conf.4
file path=usr/share/man/man4/crypto_certs.4
file path=usr/share/man/man4/d_passwd.4
file path=usr/share/man/man4/dacf.conf.4
file path=usr/share/man/man4/default_fs.4
file path=usr/share/man/man4/defaultrouter.4
file path=usr/share/man/man4/device_allocate.4
file path=usr/share/man/man4/device_contract.4
file path=usr/share/man/man4/device_maps.4
file path=usr/share/man/man4/devices.4
file path=usr/share/man/man4/dfstab.4
file path=usr/share/man/man4/dhcp_inittab.4
file path=usr/share/man/man4/dialups.4
file path=usr/share/man/man4/ethers.4
file path=usr/share/man/man4/exec_attr.4
file path=usr/share/man/man4/format.dat.4
file path=usr/share/man/man4/fspec.4
file path=usr/share/man/man4/fstypes.4
file path=usr/share/man/man4/fx_dptbl.4
file path=usr/share/man/man4/group.4
file path=usr/share/man/man4/hosts.4
file path=usr/share/man/man4/ike.config.4
file path=usr/share/man/man4/ike.preshared.4
file path=usr/share/man/man4/inet_type.4
file path=usr/share/man/man4/inetd.conf.4
file path=usr/share/man/man4/init.4
file path=usr/share/man/man4/init.d.4
file path=usr/share/man/man4/inittab.4
file path=usr/share/man/man4/ipaddrsel.conf.4
file path=usr/share/man/man4/ipnodes.4
file path=usr/share/man/man4/issue.4
file path=usr/share/man/man4/logadm.conf.4
file path=usr/share/man/man4/logindevperm.4
file path=usr/share/man/man4/loginlog.4
file path=usr/share/man/man4/magic.4
```



```
file path=usr/share/man/man4/mnttab.4
file path=usr/share/man/man4/ndpd.conf.4
file path=usr/share/man/man4/netconfig.4
file path=usr/share/man/man4/netgroup.4
file path=usr/share/man/man4/netid.4
file path=usr/share/man/man4/netmasks.4
file path=usr/share/man/man4/networks.4
file path=usr/share/man/man4/nodename.4
file path=usr/share/man/man4/nologin.4
file path=usr/share/man/man4/notrouter.4
file path=usr/share/man/man4/nscd.conf.4
file path=usr/share/man/man4/nsswitch.conf.4
file path=usr/share/man/man4/packingrules.4
file path=usr/share/man/man4/pam.conf.4
file path=usr/share/man/man4/passwd.4
file path=usr/share/man/man4/phones.4
file path=usr/share/man/man4/policy.conf.4
file path=usr/share/man/man4/prof_attr.4
file path=usr/share/man/man4/profile.4
file path=usr/share/man/man4/project.4
file path=usr/share/man/man4/protocols.4
file path=usr/share/man/man4/queuedefs.4
file path=usr/share/man/man4/rcmscript.4
file path=usr/share/man/man4/remote.4
file path=usr/share/man/man4/rpc.4
file path=usr/share/man/man4/rt_dptbl.4
file path=usr/share/man/man4/service_bundle.4
file path=usr/share/man/man4/service_provider.conf.4
file path=usr/share/man/man4/services.4
file path=usr/share/man/man4/shadow.4
file path=usr/share/man/man4/sharetab.4
file path=usr/share/man/man4/shells.4
file path=usr/share/man/man4/sulog.4
file path=usr/share/man/man4/syslog.conf.4
file path=usr/share/man/man4/term.4
file path=usr/share/man/man4/terminfo.4
file path=usr/share/man/man4/timezone.4
file path=usr/share/man/man4/ttydefs.4
file path=usr/share/man/man4/ttysrch.4
file path=usr/share/man/man4/ufsdump.4
file path=usr/share/man/man4/user_attr.4
file path=usr/share/man/man4/utmp.4
file path=usr/share/man/man4/utmpx.4
file path=usr/share/man/man4/vfstab.4
file path=usr/share/man/man4/wanboot.conf.4
file path=usr/share/man/man4/zoneinfo.4
link path=usr/share/man/man4/devid_cache.4 target=devices.4
link path=usr/share/man/man4/devname_cache.4 target=devices.4
link path=usr/share/man/man4/dumpdates.4 target=ufsdump.4
link path=usr/share/man/man4/fbtab.4 target=logindevperm.4
link path=usr/share/man/man4/fs.4 target=default_fs.4
link path=usr/share/man/man4/intro.4 target=Intro.4
link path=usr/share/man/man4/mdi_ib_cache.4 target=devices.4
link path=usr/share/man/man4/mdi_scsi_vhci_cache.4 target=devices.4
link path=usr/share/man/man4/pci_unitaddr_persistent.4 target=devices.4
link path=usr/share/man/man4/snapshot_cache.4 target=devices.4
link path=usr/share/man/man4/wtmp.4 target=utmp.4
link path=usr/share/man/man4/wtmpx.4 target=utmpx.4
```