```
**********************************************************
   15135 Thu Jan 24 10:03:15 2019
new/usr/src/cmd/setuname/setuname.c
10137 smatch fixes for setuname
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
  22 /*        All Rights Reserved   */


  25 /*
  26  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
  27  * Use is subject to license terms.
  28  */

  30 /*
  31  * Copyright (c) 2018, Joyent, Inc.
  32  */
  30 #pragma ident   "%Z%%M% %I%     %E% SMI"       /* SVr4.0 1.3 */

  34 /*
  35  *  setuname [-t] [-s name] [-n node]
  36  */

  38 /*
  39  *  Header files referenced:
  40  *      <stdio.h>       Standard I/O
  41  *      <unistd.h>      Standard UNIX definitions
  42  *      <string.h>      String handling
  43  *      <fmtmsg.h>      Standard message generation
  44  *      <ctype.h>       Character types
  45  *      <errno.h>       Error handling
  46  *      <signal.h>      Signal handling
  47  *      <sys/types.h>   Data types
  48  *      <sys/fcntl.h>   File control
  49  *      <sys/utsname.h> System Name
  50  *      <sys/sys3b.h>   sys3b() definitions
  51  *      <nlist.h>       Definitions for Sun symbol table entries
  52  */

  54 #include        <stdio.h>
  55 #include        <unistd.h>
  56 #include        <string.h>
  57 #include        <fmtmsg.h>
  58 #include        <ctype.h>
  59 #include        <errno.h>
  60 #include        <signal.h>
```



```
  61 #include        <sys/types.h>
  62 #include        <sys/uio.h>
  63 #include        <sys/fcntl.h>
  64 #include        <sys/psw.h>
  65 #include        <sys/utsname.h>

  67 #if u3b || u3b15 || u3b2
  68 #include        <sys/sys3b.h>
  69 #endif

  71 #if sun
  72 #include        <nlist.h>
  73 #include        <kvm.h>
  74 #endif

  76 /*
  77  * Externals referenced (and not defined in a header)
  78  *      optind          index to the next arg for getopt()
  79  *      opterr          FLAG, TRUE tells getopt() to write messages
  80  *      optarg          Ptr to an option's argument
  81  *      getopt()        Gets an option from the command line
  82  *      putenv()        Writes values into the environment
  83  *      exit()          Exit the process
  84  *      access()        Check accessibility of a file
  85  *      malloc()        Allocate a block of main memory
  86  *      free()          Free allocated space
  87  *      lseek()         Seek within a file
  88  *      open()          Open a file
  89  *      close()         Close an open file
  90  */
  91
  92 extern  int             optind;         /* argv[] index of next arg */
  93 extern  int             opterr;         /* TRUE if getopt() is to print msgs */
  94 extern  char           *optarg;         /* Argument to parsed option */
  95 extern  int             getopt();       /* Get an option from the command line *
  96 extern  int             putenv();       /* Put a value into the environment */
  97 extern  void            exit();         /* Exit the process */
  98 extern  int             access();       /* Check the accessibility of a file */
  99 extern  void           *malloc();       /* Get a chunk of main memory */
 100 extern  void            free();         /* Free alloc'd space */
 101 extern  long            lseek();        /* Seek within a file */
 102 extern  int             open();         /* Open a file */
 103 extern  int             close();        /* Close an open a file */
```

```
104 /*
105  *  L O C A L   D E F I N I T I O N S
106  */

108 /*
109  * Constants
110  */

112 #ifndef TRUE
113 #define TRUE            (1)
114 #endif

116 #ifndef FALSE
117 #define FALSE           (0)
118 #endif

120 #ifndef NULL
121 #define NULL            (0)
122 #endif

124 #define OPTSTRING       "tn:s:"

126 #define EX_OK           0
127 #define EX_ERROR        1

129 #define RC_FILENAME     "/etc/rc2.d/S18setuname"
130 #define RC_DIRNAME      "/etc/rc2.d"


133 /*
134  *  Messages
135  */

137 #define E_USAGE         "usage: setuname [-t] [-s name] [-n node]"
138 #define E_MISSING       "Either -s name or -n node must be specified"
139 #define E_UNAME         "Unable to get existing uname values"
140 #define E_INVNAME       "System-name invalid: %s"
141 #define E_LONGNAME      "System-name too long: %s"
142 #define E_INVNODE       "Network node-name invalid: %s"
143 #define E_LONGNODE      "Network node-name too long: %s"
144 #define E_NOPERMS       "No permissions, request denied"
145 #define E_NOSUCHDIR     "Directory doesn't exist: %s"
146 #define E_INTERNAL      "Internal error: %d"

148 /*
149  * Macros:
150  *      stdmsg(r,l,s,t)     Write a standard message.
151  *                              'r' is the recoverability flag
152  *                              'l' is the label
153  *                              's' is the severity
154  *                              't' is the text.
155  *      strend(p)           Return the address of the end of a string
156  *                          (This is supposed to be defined in <sys/inline.h>
157  *                          but that file has string-handing def'ns that
158  *                          conflict with <string.h>, so we can't use it!
159  *                          MR dn89-04701 requests this fix.
160  */
161
162 #define stdmsg(r,l,s,t) (void) fmtmsg(MM_PRINT|MM_UTIL|r,l,s,t,MM_NULLACT,MM_NUL
163 #define strend(p)       strrchr(p,'\0')

165 /*
166  * Local functions:
167  *      setuname        Changes the system name and the network node name
168  */
```



```
170 static int      setuname();             /* This does the "real" work */


173 /*
174  * Local data
175  *      lbl             Buffer for the standard message label
176  *      txt             Buffer for the standard message text
177  */
178
179 static  char            lbl[MM_MXLABELLN+1];    /* Space for std msg label */
180 static  char            msg[MM_MXTXTLN+1];      /* Space for std msg text  */
```

```
/*
 *  int main(argc, argv)
 *      int     argc
 *      char   *argv;
 */

int
main(argc, argv)
        int     argc;                   /* Argument count  */
        char   *argv[];                 /* Argument vector */
{
        /* Automatic data */
        char           *n_arg;                  /* Ptr to arg for -n */
        char           *s_arg;                  /* Ptr to arg for -s */
        int             t_seen;                 /* FLAG, -t option seen */
        char           *cmdname;                /* Ptr to the command's name */
        char           *p;                      /* Temp pointer */
        int             usageerr;               /* FLAG, TRUE if usage error */
        int             exitcode;               /* Value to exit with */
        int             c;                      /* Temp character */
        int             ok;                     /* Flag, everything okay? */

        /* Build the standard-message label */
        if (p = strrchr(argv[0], '/')) cmdname = p+1;
        else cmdname = argv[0];
        (void) strcat(strcpy(lbl, "UX:"), cmdname);

        /* Make only the text in standard messages appear (SVR4.0 only) */
        (void) putenv("MSGVERB=text");


        /* Initializations */
        n_arg = s_arg = (char *) NULL;
        t_seen = FALSE;


        /*
         * Parse command
         */

        usageerr = FALSE;
        opterr = FALSE;
        while (!usageerr && (c = getopt(argc, argv, OPTSTRING)) != EOF) switch(c

        case 'n':                       /* -n node */
            if (n_arg) usageerr = TRUE;
            else n_arg = optarg;
            break;

        case 's':                       /* -s name */
            if (s_arg) usageerr = TRUE;
            else s_arg = optarg;
            break;

        case 't':                       /* -t */
            if (t_seen) usageerr = TRUE;
            else t_seen = TRUE;
            break;

        default:                        /* Something that doesn't exist */
            usageerr = TRUE;
        }   /* switch() */

        /* If there was a usage error, report the error and exit */
        if ((argc >= (optind+1)) || usageerr) {
            stdmsg(MM_NRECOV, lbl, MM_ERROR, E_USAGE);
```



```
            exit(EX_ERROR);
        }

        /* Either -n <node> or -s <name> has to be specified */
        if (!(n_arg || s_arg)) {
            stdmsg(MM_NRECOV, lbl, MM_ERROR, E_MISSING);
            exit(EX_ERROR);
        }


        /*
         * Validate arguments:
         *  - The length of the system name must be less than SYS_NMLN-1
         *    characters,
         *  - The length of the network node-name must be less than
         *    SYS_NMLN-1 characters,
         *  - The system name must equal [a-zA-Z0-9-_]+,
         *  - The network node-name must equal [a-zA-Z0-9-_]+.
         */

        /* Check the length and the character-set of the system name */
        if (s_arg) {

            /* Check length of the system name */
            if (strlen(s_arg) > (size_t)(SYS_NMLN-1)) {
                (void) sprintf(msg, E_LONGNAME, s_arg);
                stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
                exit(EX_ERROR);
            }

            /* Check the character-set */
            ok = TRUE;
            for (p = s_arg ; ok && *p ; p++) {
                if (!isalnum(*p) && (*p != '-') && (*p != '_')) ok = FALSE;
            }
            if (!ok || (p == s_arg)) {
                (void) sprintf(msg, E_INVNAME, s_arg);
                stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
                exit(EX_ERROR);
            }
        }

        /* Check the length and the character-set of the network node-name */

        if (n_arg) {

            /* Check length of the network node-name */
            if (strlen(n_arg) > (size_t)(SYS_NMLN-1)) {
                (void) sprintf(msg, E_LONGNODE, n_arg);
                stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
                exit(EX_ERROR);
            }

            /* Check the character-set */
            ok = TRUE;
            for (p = n_arg ; ok && *p ; p++) {
                if (!isalnum(*p) && (*p != '-') && (*p != '_')) ok = FALSE;
            }
            if (!ok || (p == n_arg)) {
                (void) sprintf(msg, E_INVNODE, n_arg);
                stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
                exit(EX_ERROR);
            }
        }
```

```
313         /*
314          * Make sure we have access to needed resources:
315          *   -  Read/write access to kernel memory (/dev/kmem)
316          *   -  If -t is not specified, read/write access to /etc/rc2.d
317          *   -  If -t is not specified, read access to /etc/rc2.d/S18setuname
318          */
319
320         if (access("/dev/kmem", R_OK|W_OK) == 0) {
321             if (access(RC_DIRNAME, R_OK|W_OK) == 0) {
322                 if ((access(RC_FILENAME, R_OK) != 0) &&
323                     (access(RC_FILENAME, F_OK) == 0)) {
324                     stdmsg(MM_NRECOV, lbl, MM_ERROR, E_NOPERMS);
325                     exit(EX_ERROR);
326                 }
327             }
328             else {
329                 if (access(RC_DIRNAME, F_OK) == 0) {
330                     stdmsg(MM_NRECOV, lbl, MM_ERROR, E_NOPERMS);
331                     exit(EX_ERROR);
332                 }
333                 else {
334                     (void) sprintf(msg, E_NOSUCHDIR, RC_DIRNAME);
335                     stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
336                     exit(EX_ERROR);
337                 }
338             }
339         }
340         else {
341             stdmsg(MM_NRECOV, lbl, MM_ERROR, E_NOPERMS);
342             exit(EX_ERROR);
343         }


346         /* Attempt the setuname */
347         if (setuname(t_seen, s_arg, n_arg) == 0) exitcode = EX_OK;
348         else {
349             (void) sprintf(msg, E_INTERNAL, errno);
350             stdmsg(MM_NRECOV, lbl, MM_ERROR, msg);
351             exitcode = EX_ERROR;
352         }

354         /* Finished */
355         return (exitcode);
356 }  /* main() */
```



```
357 /*
358  * int setuname(temp, name, node)
359  *      int     temp
360  *      char   *name
361  *      char   *node
362  *
363  *      Set any or all of the following machine parameters, either
364  *      temporarily or permanently, depending on <temp>.
365  *          - System name
366  *          - Network Node-name
367  */

369 static int
370 setuname(temp, sysname, nodename)
371         int     temp;           /* Set in kernel only flag */
372         char   *sysname;        /* System name */
373         char   *nodename;       /* Network node-name */
374 {
375         /* Automatic Data */
376         struct utsname  utsname;        /* Space for the kernel's utsname inform
377 #if u3b || u3b15 || u3b2
378         struct s3bsym  *symbtbl;        /* The kernel's symbol table */
379 #endif
380 #if sun
381         struct nlist nl[] = {
382                 {"utsname", 0, 0, 0, 0, 0},
383                 {NULL}
384         };
385         kvm_t *kd;
386 #endif
387         uintptr_t       utsname_addr;   /* Addr of "utsname" in the kernel */
388         char           *sysnm = (char *)NULL;           /* System name to set (f
389         char           *nodenm = (char *)NULL;          /* Network node-name to
390         FILE           *fd;             /* Std I/O File Descriptor for /etc/rc2.
391         char           *p;              /* Temp pointer */
392         void          (*oldsighup)();   /* Function to call for SIGHUP */
393         void          (*oldsigint)();   /* Function to call for SIGINT */
394         int            rtncd;           /* Value to return to the caller */
395         unsigned long   symbtblsz;      /* The size of the kernel's symbol table
396         int             memfd;          /* File descriptor:  open kernel memory
397         int             i;              /* Temp counter */


400         /* Nothing's gone wrong yet (but we've only just begun!) */
401         rtncd = 0;


404         /*
405          * Get the virtual address of the symbol "utsname" in the kernel
406          * so we can get set the system name and/or the network node-name
407          * directly in the kernel's memory space.
408          */

410 #if u3b || u3b15 || u3b2
411         if ((sys3b(S3BSYM, (struct s3bsym *) &symbtblsz, sizeof(symbtblsz)) == 0
412             (symbtbl = (struct s3bsym *) malloc(symbtblsz))) {

414             (void) sys3b(S3BSYM, symbtbl, symbtblsz);
415             p = (char *) symbtbl;
416             for (i = symbtbl->count; i-- && (strcmp(p, "utsname") != 0) ; p = S3
417             if (i >= 0) utsname_addr = S3BSVAL(p);
418             else rtncd = -1;
419             free((void *) symbtbl);

421         } else rtncd = -1;
```

```
#elif sun
        /* Check out namelist and memory files. */
        if ((kd = kvm_open(NULL, NULL, NULL, O_RDWR, NULL)) == NULL)
                rtncd = -1;
        else if (kvm_nlist(kd, nl) != 0)
                rtncd = -1;
        else if (nl[0].n_value == 0)
                rtncd = -1;
        else
                utsname_addr = (uintptr_t)nl[0].n_value;
#else
        if (nlist("/unix", nl) != 0)
                rtncd = -1;
#endif
        if (rtncd != 0) return(rtncd);

        /*
         * Open the kernel's memory, get the existing "utsname" structure,
         * change the system name and/or the network node-name in that struct,
         * write it back out to kernel memory, then close kernel memory.
         */
#ifdef sun
        if (kvm_kread(kd, utsname_addr, &utsname, sizeof (utsname)) ==
            sizeof (utsname)) {
                if (sysname)
                        (void) strncpy(utsname.sysname, sysname,
                            sizeof (utsname.sysname));
                if (nodename)
                        (void) strncpy(utsname.nodename, nodename,
                            sizeof (utsname.nodename));
                (void) kvm_kwrite(kd, utsname_addr, &utsname, sizeof (utsname));
                (void) kvm_close(kd);
                kvm_close(kd);
        } else
                return (-1);
#else /* sun */
        if ((memfd = open("/dev/kmem", O_RDWR, 0)) > 0) {
            if ((lseek(memfd, (long) utsname_addr, SEEK_SET) != -1) &&
                (read(memfd, &utsname, sizeof(utsname)) == sizeof(utsname))) {
                if (sysname) (void) strncpy(utsname.sysname, sysname, sizeof(uts
                if (nodename) (void) strncpy(utsname.nodename, nodename, sizeof(
                (void) lseek(memfd, (long) utsname_addr, SEEK_SET);
                (void) write(memfd, &utsname, sizeof(utsname));
                (void) close(memfd);
            } else rtncd = -1;
        } else rtncd = -1;
        if (rtncd != 0) return(rtncd);
#endif /* sun */


        /*
         * If the "temp" flag is FALSE, we need to permanently set the
         * system name in the file  /etc/rc2.d/S18setuname
         */

        if (!temp) {
            /*
             * If a name was specified by the caller, use that, otherwise, use
             * whatever was in the "rc" file.
             */

                if (sysname) sysnm = sysname;
                if (nodename) nodenm = nodename;


            /*
```



```
             * Write the file /etc/rc2.d/S18setuname so that the system name is
             * set on boots and state changes.
             *
             * DISABLED SIGNALS: SIGHUP, SIGINT
             */

            /* Give us a reasonable chance to complete without interruptions */
                oldsighup = signal(SIGHUP, SIG_IGN);
                oldsigint = signal(SIGINT, SIG_IGN);

            /* Write the new setuname "rc" file */
                if (sysname != NULL) {
                        if ((fd = fopen(RC_FILENAME, "w")) != (FILE *) NULL) {
                                (void) fprintf(fd, "# %s\n", sysnm);
                                (void) fprintf(fd, "#\n");
                                (void) fprintf(fd, "# This script, generated by
                                (void) fprintf(fd, "# sets the system's system-n
                                (void) fprintf(fd, "#\n");
                                if (sysnm && *sysnm)
                                        (void) fprintf(fd, "setuname -t -s %s\n"
                                (void) fclose(fd);
                        } else return(rtncd = -1);
                }

                if(nodename != NULL) {
                        char curname[SYS_NMLN];
                        int curlen;
                        FILE *file;

                        if ((file = fopen("/etc/nodename", "r")) != NULL) {
                                curlen = fread(curname, sizeof(char), SYS_NMLN,
                                for (i = 0; i < curlen; i++) {
                                        if (curname[i] == '\n') {
                                                curname[i] = '\0';
                                                break;
                                        }
                                }
                                if (i == curlen) {
                                        curname[curlen] = '\0';
                                }
                                (void)fclose(file);
                        } else {
                                curname[0] = '\0';
                        }
                        if (strcmp(curname, nodenm) != 0) {
                                if ((file = fopen("/etc/nodename", "w")) == NULL
                                        (void) fprintf(stderr, "setuname: error
                                        exit(1);
                                }
                                if (fprintf(file, "%s\n", nodenm) < 0) {
                                        (void) fprintf(stderr, "setuname: error
                                        exit(1);
                                }
                                (void)fclose(file);
                        }
                }
            /* Restore signal handling */
                (void) signal(SIGHUP, oldsighup);
                (void) signal(SIGINT, oldsigint);
        }       /* if (!temp) */

        /* Fini */
        return(rtncd);
}
```

_____***unchanged_portion_omitted_***