```
**********************************************************
    9191 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/grpck/grpck.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2007 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"

#include <sys/param.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <pwd.h>
#include <errno.h>
#include <locale.h>
#include <limits.h>

#define BADLINE "Too many/few fields"
#define TOOLONG "Line too long"
#define NONAME  "No group name"
#define BADNAME "Bad character(s) in group name"
#define BADGID  "Invalid GID"
#define NULLNAME "Null login name"
#define NOTFOUND "Logname not found in password file"
#define DUPNAME "Duplicate logname entry"
#define DUPNAME2 "Duplicate logname entry (gid first occurs in passwd entry)"
#define NOMEM   "Out of memory"
#define NGROUPS "Maximum groups exceeded for logname "
#define BLANKLINE "Blank line detected. Please remove line"
#define LONGNAME  "Group name too long"

#ifdef  LOGNAME_MAX_ILLUMOS
```

```
#define _LOGNAME_MAX    LOGNAME_MAX_ILLUMOS
#else /* LOGNAME_MAX_ILLUMOS */
#define _LOGNAME_MAX    LOGNAME_MAX
#endif /* LOGNAME_MAX_ILLUMOS */

int eflag, badchar, baddigit, badlognam, colons, len;
static int longnam = 0;
int code;

#define MYBUFSIZE (LINE_MAX)    /* max line length including newline and null */
#define NUM_COLONS      3

char *buf;
char *nptr;
char *cptr;
FILE *fptr;
gid_t gid;
void error(char *msg);

struct group {
        struct group *nxt;
        int cnt;
        gid_t grp;
};
______unchanged_portion_omitted_

int
main(int argc, char *argv[])
{
        struct passwd *pwp;
        struct node *root = NULL;
        struct node *t;
        struct group *gp;
        int ngroups_max;
        int ngroups = 0;
        int listlen;
        int i;
        int lineno = 0;
        char *buf_off, *tmpbuf;
        int delim[NUM_COLONS + 1], buf_len, bufsize;

        (void) setlocale(LC_ALL, "");

#if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"
#endif
        (void) textdomain(TEXT_DOMAIN);

        code = 0;
        ngroups_max = sysconf(_SC_NGROUPS_MAX);

        if (argc == 1)
                argv[1] = "/etc/group";
        else if (argc != 2) {
                fprintf(stderr, gettext("usage: %s filename\n"), *argv);
                exit(1);
        }

        if ((fptr = fopen(argv[1], "r")) == NULL) {
                fprintf(stderr, gettext("cannot open file %s: %s\n"), argv[1],
                    strerror(errno));
                exit(1);
        }

#ifdef ORIG_SVR4
        while ((pwp = getpwent()) != NULL) {
                t = (struct node *)emalloc(sizeof (*t) + strlen(pwp->pw_name));
                t->next = root;
                root = t;
                strcpy(t->user, pwp->pw_name);
                t->ngroups = 1;
                if (!ngroups_max)
                        t->groups = NULL;
                else {
                        t->groups = (struct group *)
                            emalloc(sizeof (struct group));
                        t->groups->grp = pwp->pw_gid;
                        t->groups->cnt = 1;
                        t->groups->nxt = NULL;
                }
        }
#endif

        bufsize = MYBUFSIZE;
        if ((buf = malloc(bufsize)) == NULL) {
                (void) fprintf(stderr, gettext(NOMEM));
                exit(1);
        }
        while (!feof(fptr) && !ferror(fptr)) {
                buf_len = 0;
                buf_off = buf;
                while (fgets(buf_off, (bufsize - buf_len), fptr) != NULL) {
                        buf_len += strlen(buf_off);
                        if (buf[buf_len - 1] == '\n' || feof(fptr))
                                break;
                        tmpbuf = realloc(buf, (bufsize + MYBUFSIZE));
                        if (tmpbuf == NULL) {
                                (void) fprintf(stderr, gettext(NOMEM));
                                exit(1);
                        }
                        bufsize += MYBUFSIZE;
                        buf = tmpbuf;
                        buf_off = buf + buf_len;
                }
                if (buf_len == 0)
                        continue;

                /* Report error to be consistent with libc */
                if ((buf_len + 1) > LINE_MAX)
                        error(TOOLONG);

                lineno++;
                if (buf[0] == '\n')    /* blank lines are ignored */
                {
                        code = 1;               /* exit with error code = 1 */
                        eflag = 0;      /* force print of "blank" line */
                        fprintf(stderr, "\n%s %d\n", gettext(BLANKLINE),
                            lineno);
                        continue;
                }

                if (buf[buf_len - 1] == '\n') {
                        if ((tmpbuf = strdup(buf)) == NULL) {
                                (void) fprintf(stderr, gettext(NOMEM));
                                exit(1);
                        }
                        tmpbuf[buf_len - 1] = ',';
                } else {
                        if ((tmpbuf = malloc(buf_len + 2)) == NULL) {
                                (void) fprintf(stderr, gettext(NOMEM));
                                exit(1);
                        }
```

```
                        (void) strcpy(tmpbuf, buf);
                        tmpbuf[buf_len++] = ',';
                        tmpbuf[buf_len] = '\0';
                }

                colons = 0;
                eflag = 0;
                badchar = 0;
                baddigit = 0;
                badlognam = 0;
                gid = 0;

                ngroups++;      /* Increment number of groups found */
                /* Check that entry is not a nameservice redirection */

                if (buf[0] == '+' || buf[0] == '-')  {
                        /*
                         * Should set flag here to allow special case checking
                         * in the rest of the code,
                         * but for now, we'll just ignore this entry.
                         */
                        free(tmpbuf);
                        continue;
                }

                /*      Check number of fields  */

                for (i = 0; buf[i] != NULL; i++) {
                        if (buf[i] == ':') {
                                delim[colons] = i;
                                if (++colons > NUM_COLONS)
                                        break;
                        }
                }
                if (colons != NUM_COLONS) {
                        error(BADLINE);
                        free(tmpbuf);
                        continue;
                }

                /* check to see that group name is at least 1 character */
                /* and that all characters are lowrcase or digits.      */

                if (buf[0] == ':')
                        error(NONAME);
                else {
                        for (i = 0; buf[i] != ':'; i++) {
                                if (i >= _LOGNAME_MAX)
                                if (i >= LOGNAME_MAX)
                                        longnam++;
                                if (!(islower(buf[i]) || isdigit(buf[i])))
                                        badchar++;
                        }
                        if (longnam > 0)
                                error(LONGNAME);
                        if (badchar > 0)
                                error(BADNAME);
                }

                /*      check that GID is numeric and <= 31 bits        */

                len = (delim[2] - delim[1]) - 1;

                if (len > 10 || len < 1)
                        error(BADGID);
                else {
                        for (i = (delim[1]+1); i < delim[2]; i++) {
                                if (! (isdigit(buf[i])))
                                        baddigit++;
                                else if (baddigit == 0)
                                        gid = gid * 10 + (gid_t)(buf[i] - '0');
                                /* converts ascii GID to decimal */
                        }
                        if (baddigit > 0)
                                error(BADGID);
                        else if (gid > (gid_t)MAXUID)
                                error(BADGID);
                }

                /*  check that logname appears in the passwd file  */

                nptr = &tmpbuf[delim[2]];
                nptr++;

                listlen = strlen(nptr) - 1;

                while ((cptr = strchr(nptr, ',')) != NULL) {
                        *cptr = NULL;
                        if (*nptr == NULL) {
                                if (listlen)
                                        error(NULLNAME);
                                nptr++;
                                continue;
                        }

                        for (t = root; t != NULL; t = t->next) {
                                if (strcmp(t->user, nptr) == 0)
                                        break;
                        }
                        if (t == NULL) {
#ifndef ORIG_SVR4
                                /*
                                 * User entry not found, so check if in
                                 *  password file
                                 */
                                struct passwd *pwp;

                                if ((pwp = getpwnam(nptr)) == NULL) {
#endif
                                        badlognam++;
                                        error(NOTFOUND);
                                        goto getnext;
#ifndef ORIG_SVR4
                                }

                                /* Usrname found, so add entry to user-list */
                                t = (struct node *)
                                    emalloc(sizeof (*t) + strlen(nptr));
                                t->next = root;
                                root = t;
                                strcpy(t->user, nptr);
                                t->ngroups = 1;
                                if (!ngroups_max)
                                        t->groups = NULL;
                                else {
                                        t->groups = (struct group *)
                                            emalloc(sizeof (struct group));
                                        t->groups->grp = pwp->pw_gid;
                                        t->groups->cnt = 1;
                                        t->groups->nxt = NULL;
                                }
                        }
```

```
#endif
                        if (!ngroups_max)
                                goto getnext;

                        t->ngroups++;

                        /*
                         * check for duplicate logname in group
                         */

                        for (gp = t->groups; gp != NULL; gp = gp->nxt) {
                                if (gid == gp->grp) {
                                        if (gp->cnt++ == 1) {
                                                badlognam++;
                                                if (gp->nxt == NULL)
                                                        error(DUPNAME2);
                                                else
                                                        error(DUPNAME);
                                        }
                                        goto getnext;
                                }
                        }

                        gp = (struct group *)emalloc(sizeof (struct group));
                        gp->grp = gid;
                        gp->cnt = 1;
                        gp->nxt = t->groups;
                        t->groups = gp;
getnext:
                        nptr = ++cptr;
                }
                free(tmpbuf);
        }

        if (ngroups == 0) {
                fprintf(stderr, gettext("Group file '%s' is empty\n"), argv[1]);
                code = 1;
        }

        if (ngroups_max) {
                for (t = root; t != NULL; t = t->next) {
                        if (t->ngroups > ngroups_max) {
                                fprintf(stderr, "\n\n%s%s (%d)\n",
                                    NGROUPS, t->user, t->ngroups);
                                code = 1;
                        }
                }
        }
        return (code);
}
```

______***unchanged_portion_omitted_***

```
***********************************************************
  115298 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/init/init.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
***********************************************************
```

```
/*
 * 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) 2013 Gary Mills
 *
 * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 */

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

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

/*
 * init(1M) is the general process spawning program.  Its primary job is to
 * start and restart svc.startd for smf(5).  For backwards-compatibility it also
 * spawns and respawns processes according to /etc/inittab and the current
 * run-level.  It reads /etc/default/inittab for general configuration.
 *
 * To change run-levels the system administrator runs init from the command
 * line with a level name.  init signals svc.startd via libscf and directs the
 * zone's init (pid 1 in the global zone) what to do by sending it a signal;
 * these signal numbers are commonly refered to in the code as 'states'.  Valid
 * run-levels are [sS0123456].  Additionally, init can be given directives
 * [qQabc], which indicate actions to be taken pertaining to /etc/inittab.
 *
 * When init processes inittab entries, it finds processes that are to be
 * spawned at various run-levels.  inittab contains the set of the levels for
 * which each inittab entry is valid.
 *
 * State File and Restartability
 *   Premature exit by init(1M) is handled as a special case by the kernel:
 *   init(1M) will be immediately re-executed, retaining its original PID.  (PID
```

```
 *   1 in the global zone.)  To track the processes it has previously spawned,
 *   as well as other mutable state, init(1M) regularly updates a state file
 *   such that its subsequent invocations have knowledge of its various
 *   dependent processes and duties.
 *
 * Process Contracts
 *   We start svc.startd(1M) in a contract and transfer inherited contracts when
 *   restarting it.  Everything else is started using the legacy contract
 *   template, and the created contracts are abandoned when they become empty.
 *
 * utmpx Entry Handling
 *   Because init(1M) no longer governs the startup process, its knowledge of
 *   when utmpx becomes writable is indirect.  However, spawned processes
 *   expect to be constructed with valid utmpx entries.  As a result, attempts
 *   to write normal entries will be retried until successful.
 *
 * Maintenance Mode
 *   In certain failure scenarios, init(1M) will enter a maintenance mode, in
 *   which it invokes sulogin(1M) to allow the operator an opportunity to
 *   repair the system.  Normally, this operation is performed as a
 *   fork(2)-exec(2)-waitpid(3C) sequence with the parent waiting for repair or
 *   diagnosis to be completed.  In the cases that fork(2) requests themselves
 *   fail, init(1M) will directly execute sulogin(1M), and allow the kernel to
 *   restart init(1M) on exit from the operator session.
 *
 *   One scenario where init(1M) enters its maintenance mode is when
 *   svc.startd(1M) begins to fail rapidly, defined as when the average time
 *   between recent failures drops below a given threshold.
 */

#include <sys/contract/process.h>
#include <sys/ctfs.h>
#include <sys/stat.h>
#include <sys/statvfs.h>
#include <sys/stropts.h>
#include <sys/systeminfo.h>
#include <sys/time.h>
#include <sys/termios.h>
#include <sys/tty.h>
#include <sys/types.h>
#include <sys/utsname.h>

#include <bsm/adt_event.h>
#include <bsm/libbsm.h>
#include <security/pam_appl.h>

#include <assert.h>
#include <ctype.h>
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <libcontract.h>
#include <libcontract_priv.h>
#include <libintl.h>
#include <libscf.h>
#include <libscf_priv.h>
#include <poll.h>
#include <procfs.h>
#include <signal.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdio_ext.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <syslog.h>
```

```c
#include <time.h>
#include <ulimit.h>
#include <unistd.h>
#include <utmpx.h>
#include <wait.h>
#include <zone.h>
#include <ucontext.h>

#undef  sleep

#define fioctl(p, sptr, cmd)    ioctl(fileno(p), sptr, cmd)
#define min(a, b)               (((a) < (b)) ? (a) : (b))

#define TRUE    1
#define FALSE   0
#define FAILURE -1

#define UT_USER_SZ      32      /* Size of a utmpx ut_user field */
#define UT_LINE_SZ      32      /* Size of a utmpx ut_line field */

/*
 * SLEEPTIME    The number of seconds "init" sleeps between wakeups if
 *              nothing else requires this "init" wakeup.
 */
#define SLEEPTIME       (5 * 60)

/*
 * MAXCMDL      The maximum length of a command string in inittab.
 */
#define MAXCMDL 512

/*
 * EXEC         The length of the prefix string added to all comamnds
 *              found in inittab.
 */
#define EXEC    (sizeof ("exec ") - 1)

/*
 * TWARN        The amount of time between warning signal, SIGTERM,
 *              and the fatal kill signal, SIGKILL.
 */
#define TWARN   5

#define id_eq(x, y)     ((x[0] == y[0] && x[1] == y[1] && x[2] == y[2] &&\
                        x[3] == y[3]) ? TRUE : FALSE)

/*
 * The kernel's default umask is 022 these days; since some processes inherit
 * their umask from init, init will set it from CMASK in /etc/default/init.
 * init gets the default umask from the kernel, it sets it to 022 whenever
 * it wants to create a file and reverts to CMASK afterwards.
 */

static int cmask;

/*
 * The following definitions, concluding with the 'lvls' array, provide a
 * common mapping between level-name (like 'S'), signal number (state),
 * run-level mask, and specific properties associated with a run-level.
 * This array should be accessed using the routines lvlname_to_state(),
 * lvlname_to_mask(), state_to_mask(), and state_to_flags().
 */

/*
 * Correspondence of signals to init actions.
 */
```

```c
#define LVLQ            SIGHUP
#define LVL0            SIGINT
#define LVL1            SIGQUIT
#define LVL2            SIGILL
#define LVL3            SIGTRAP
#define LVL4            SIGIOT
#define LVL5            SIGEMT
#define LVL6            SIGFPE
#define SINGLE_USER     SIGBUS
#define LVLa            SIGSEGV
#define LVLb            SIGSYS
#define LVLc            SIGPIPE

/*
 * Bit Mask for each level.  Used to determine legal levels.
 */
#define MASK0   0x0001
#define MASK1   0x0002
#define MASK2   0x0004
#define MASK3   0x0008
#define MASK4   0x0010
#define MASK5   0x0020
#define MASK6   0x0040
#define MASKSU  0x0080
#define MASKa   0x0100
#define MASKb   0x0200
#define MASKc   0x0400

#define MASK_NUMERIC (MASK0 | MASK1 | MASK2 | MASK3 | MASK4 | MASK5 | MASK6)
#define MASK_abc (MASKa | MASKb | MASKc)

/*
 * Flags to indicate properties of various states.
 */
#define LSEL_RUNLEVEL   0x0001  /* runlevels you can transition to */

typedef struct lvl {
        int     lvl_state;
        int     lvl_mask;
        char    lvl_name;
        int     lvl_flags;
} lvl_t;
```

______***unchanged_portion_omitted_***

```c
/*
 * getcmd() parses lines from inittab.  Each time it finds a command line
 * it will return TRUE as well as fill the passed CMD_LINE structure and
 * the shell command string.  When the end of inittab is reached, FALSE
 * is returned inittab is automatically opened if it is not currently open
 * and is closed when the end of the file is reached.
 */
static FILE *fp_inittab = NULL;

static int
getcmd(struct CMD_LINE *cmd, char *shcmd)
{
        char    *ptr;
        int     c, lastc, state;
        char    *ptr1;
        int     answer, i, proceed;
        struct  stat    sbuf;
        static char *actions[] = {
                "off", "respawn", "ondemand", "once", "wait", "boot",
                "bootwait", "powerfail", "powerwait", "initdefault",
                "sysinit",
        };
        static short act_masks[] = {
                M_OFF, M_RESPAWN, M_ONDEMAND, M_ONCE, M_WAIT, M_BOOT,
                M_BOOTWAIT, M_PF, M_PWAIT, M_INITDEFAULT, M_SYSINIT,
        };
        /*
         * Only these actions will be allowed for entries which
         * are specified for single-user mode.
         */
        short su_acts = M_INITDEFAULT | M_PF | M_PWAIT | M_WAIT;

        if (fp_inittab == NULL) {
                /*
                 * Before attempting to open inittab we stat it to make
                 * sure it currently exists and is not empty.  We try
                 * several times because someone may have temporarily
                 * unlinked or truncated the file.
                 */
                for (i = 0; i < 3; i++) {
                        if (stat(INITTAB, &sbuf) == -1) {
                                if (i == 2) {
                                        console(B_TRUE,
                                            "Cannot stat %s, errno: %d\n",
                                            INITTAB, errno);
                                        return (FAILURE);
                                } else {
                                        timer(3);
                                }
                        } else if (sbuf.st_size < 10) {
                                if (i == 2) {
                                        console(B_TRUE,
                                            "%s truncated or corrupted\n",
                                            INITTAB);
                                        return (FAILURE);
                                } else {
                                        timer(3);
                                }
                        } else {
                                break;
                        }
                }

                /*
                 * If unable to open inittab, print error message and
                 * return FAILURE to caller.
                 */
                if ((fp_inittab = fopen(INITTAB, "r")) == NULL) {
                        console(B_TRUE, "Cannot open %s errno: %d\n", INITTAB,
                            errno);
                        return (FAILURE);
                }
        }

        /*
         * Keep getting commands from inittab until you find a
         * good one or run out of file.
         */
        for (answer = FALSE; answer == FALSE; ) {
                /*
                 * Zero out the cmd itself before trying next line.
                 */
                bzero(cmd, sizeof (struct CMD_LINE));

                /*
                 * Read in lines of inittab, parsing at colons, until a line is
                 * read in which doesn't end with a backslash.  Do not start if
                 * the first character read is an EOF.  Note that this means
```

```
                 * that lines which don't end in a newline are still processed,
                 * since the "for" will terminate normally once started,
                 * regardless of whether line terminates with a newline or EOF.
                 */
                state = FAILURE;
                if ((c = fgetc(fp_inittab)) == EOF) {
                        answer = FALSE;
                        (void) fclose(fp_inittab);
                        fp_inittab = NULL;
                        break;
                }

                for (proceed = TRUE, ptr = shcmd, state = ID, lastc = '\0';
                    proceed && c != EOF;
                    lastc = c, c = fgetc(fp_inittab)) {
                        /* If we're not in the FAILURE state and haven't */
                        /* yet reached the shell command field, process  */
                        /* the line, otherwise just look for a real end  */
                        /* of line.                                      */
                        if (state != FAILURE && state != COMMAND) {
                        /*
                         * Squeeze out spaces and tabs.
                         */
                        if (c == ' ' || c == '\t')
                                continue;

                        /*
                         * Ignore characters in a comment, except for the \n.
                         */
                        if (state == COMMENT) {
                                if (c == '\n') {
                                        lastc = ' ';
                                        break;
                                } else {
                                        continue;
                                }
                        }

                        /*
                         * Detect comments (lines whose first non-whitespace
                         * character is '#') by checking that we're at the
                         * beginning of a line, have seen a '#', and haven't
                         * yet accumulated any characters.
                         */
                        if (state == ID && c == '#' && ptr == shcmd) {
                                state = COMMENT;
                                continue;
                        }

                        /*
                         * If the character is a ':', then check the
                         * previous field for correctness and advance
                         * to the next field.
                         */
                        if (c == ':') {
                                switch (state) {

                                case ID :
                                /*
                                 * Check to see that there are only
                                 * 1 to 4 characters for the id.
                                 */
                                if ((i = ptr - shcmd) < 1 || i > 4) {
                                        state = FAILURE;
                                } else {
                                        bcopy(shcmd, &cmd->c_id[0], i);
                                        ptr = shcmd;
                                        state = LEVELS;
                                }
                                break;

                                case LEVELS :
                                /*
                                 * Build a mask for all the levels for
                                 * which this command will be legal.
                                 */
                                for (cmd->c_levels = 0, ptr1 = shcmd;
                                    ptr1 < ptr; ptr1++) {
                                        int mask;
                                        if (lvlname_to_mask(*ptr1,
                                            &mask) == -1) {
                                                state = FAILURE;
                                                break;
                                        }
                                        cmd->c_levels |= mask;
                                }
                                if (state != FAILURE) {
                                        state = ACTION;
                                        ptr = shcmd;    /* Reset the buffer */
                                }
                                break;

                                case ACTION :
                                /*
                                 * Null terminate the string in shcmd buffer and
                                 * then try to match against legal actions.  If
                                 * the field is of length 0, then the default of
                                 * "RESPAWN" is used if the id is numeric,
                                 * otherwise the default is "OFF".
                                 */
                                if (ptr == shcmd) {
                                        if (isdigit(cmd->c_id[0]) &&
                                            (cmd->c_id[1] == '\0' ||
                                            isdigit(cmd->c_id[1])) &&
                                            (cmd->c_id[2] == '\0' ||
                                            isdigit(cmd->c_id[2])) &&
                                            (cmd->c_id[3] == '\0' ||
                                            isdigit(cmd->c_id[3])))
                                                cmd->c_action = M_RESPAWN;
                                        else
                                                cmd->c_action = M_OFF;
                                } else {
                                        for (cmd->c_action = 0, i = 0,
                                            *ptr = '\0';
                                            i <
                                            sizeof (actions)/sizeof (char *);
                                    for (cmd->c_action = 0, i = 0, *ptr = '\0';
                                    i < sizeof (actions)/sizeof (char *);
                                            i++) {
                                        if (strcmp(shcmd, actions[i]) == 0) {
                                                if ((cmd->c_levels & MASKSU) &&
                                                    !(act_masks[i] & su_acts))
                                                        cmd->c_action = 0;
                                                else
                                                        cmd->c_action =
                                                            act_masks[i];
                                                cmd->c_action = act_masks[i];
                                                break;
                                        }
                                        }
                                }
```

```
                                /*
                                 * If the action didn't match any legal action,
                                 * set state to FAILURE.
                                 */
                                if (cmd->c_action == 0) {
                                        state = FAILURE;
                                } else {
                                        state = COMMAND;
                                        (void) strcpy(shcmd, "exec ");
                                }
                                ptr = shcmd + EXEC;
                                break;
                                }
                                continue;
                        }
                }

                /* If the character is a '\n', then this is the end of a */
                /* line.  If the '\n' wasn't preceded by a backslash, */
                /* it is also the end of an inittab command.  If it was */
                /* preceded by a backslash then the next line is a */
                /* continuation.  Note that the continuation '\n' falls */
                /* through and is treated like other characters and is */
                /* stored in the shell command line. */
                if (c == '\n' && lastc != '\\') {
                        proceed = FALSE;
                        *ptr = '\0';
                        break;
                }

                /* For all other characters just stuff them into the */
                /* command as long as there aren't too many of them. */
                /* Make sure there is room for a terminating '\0' also. */
                if (ptr >= shcmd + MAXCMDL - 1)
                        state = FAILURE;
                else
                        *ptr++ = (char)c;

                /* If the character we just stored was a quoted */
                /* backslash, then change "c" to '\0', so that this     */
                /* backslash will not cause a subsequent '\n' to appear */
                /* quoted.  In otherwords '\' '\' '\n' is the real end */
                /* of a command, while '\' '\n' is a continuation. */
                if (c == '\\' && lastc == '\\')
                        c = '\0';
                }

                /*
                 * Make sure all the fields are properly specified
                 * for a good command line.
                 */
                if (state == COMMAND) {
                        answer = TRUE;
                        cmd->c_command = shcmd;

                        /*
                         * If no default level was supplied, insert
                         * all numerical levels.
                         */
                        if (cmd->c_levels == 0)
                                cmd->c_levels = MASK_NUMERIC;

                        /*
                         * If no action has been supplied, declare this
                         * entry to be OFF.
                         */
                        if (cmd->c_action == 0)
                                cmd->c_action = M_OFF;

                        /*
                         * If no shell command has been supplied, make sure
                         * there is a null string in the command field.
                         */
                        if (ptr == shcmd + EXEC)
                                *shcmd = '\0';
                } else
                        answer = FALSE;

                /*
                 * If we have reached the end of inittab, then close it
                 * and quit trying to find a good command line.
                 */
                if (c == EOF) {
                        (void) fclose(fp_inittab);
                        fp_inittab = NULL;
                        break;
                }
        }
        return (answer);
}
______unchanged_portion_omitted_

/*
 * boot_init(): Do initialization things that should be done at boot.
 */
void
boot_init()
{
        int i;
        struct PROC_TABLE *process, *oprocess;
        struct CMD_LINE cmd;
        char    line[MAXCMDL];
        char    svc_aux[SVC_AUX_SIZE];
        char    init_svc_fmri[SVC_FMRI_SIZE];
        char *old_path;
        int maxfiles;

        /* Use INIT_PATH for sysinit cmds */
        old_path = glob_envp[0];
        glob_envp[0] = malloc((unsigned)(strlen(INIT_PATH)+2));
        (void) strcpy(glob_envp[0], INIT_PATH);

        /*
         * Scan inittab(4) and process the special svc.startd entry, initdefault
         * and sysinit entries.
         */
        while (getcmd(&cmd, &line[0]) == TRUE) {
                if (startd_tmpl >= 0 && id_eq(cmd.c_id, "smf")) {
                        process_startd_line(&cmd, line);
                        (void) snprintf(startd_svc_aux, SVC_AUX_SIZE,
                            INITTAB_ENTRY_ID_STR_FORMAT, cmd.c_id);
                } else if (cmd.c_action == M_INITDEFAULT) {
                        /*
                         * initdefault is no longer meaningful, as the SMF
                         * milestone controls what (legacy) run level we
                         * boot to.
                         */
                        console(B_TRUE,
                            "Ignoring legacy \"initdefault\" entry.\n");
                } else if (cmd.c_action == M_SYSINIT) {
                        /*
                         * Execute the "sysinit" entry and wait for it to
```

```
                         * complete.  No bookkeeping is performed on these
                         * entries because we avoid writing to the file system
                         * until after there has been an chance to check it.
                         */
                        if (process = findpslot(&cmd)) {
                                (void) sighold(SIGCLD);
                                (void) snprintf(svc_aux, SVC_AUX_SIZE,
                                    INITTAB_ENTRY_ID_STR_FORMAT, cmd.c_id);
                                (void) snprintf(init_svc_fmri, SVC_FMRI_SIZE,
                                    SVC_INIT_PREFIX INITTAB_ENTRY_ID_STR_FORMAT,
                                    cmd.c_id);
                                if (legacy_tmpl >= 0) {
                                        (void) ct_pr_tmpl_set_svc_fmri(
                                            legacy_tmpl, init_svc_fmri);
                                        (void) ct_pr_tmpl_set_svc_aux(
                                            legacy_tmpl, svc_aux);
                                }

                                for (oprocess = process;
                                    (process = efork(M_OFF, oprocess,
                                    (NAMED|NOCLEANUP))) == NO_ROOM;
                                    /* CSTYLED */)
                                        ;
                                (void) sigrelse(SIGCLD);

                                if (process == NULLPROC) {
                                        maxfiles = ulimit(UL_GDESLIM, 0);

                                        for (i = 0; i < maxfiles; i++)
                                                (void) fcntl(i, F_SETFD,
                                                    FD_CLOEXEC);
                                        (void) execle(SH, "INITSH", "-c",
                                            cmd.c_command,
                                            (char *)0, glob_envp);
                                        console(B_TRUE,
"Command\n\"%s\"\n failed to execute.  errno = %d (exec of shell failed)\n",
                                            cmd.c_command, errno);
                                        exit(1);
                                } else
                                        while (waitproc(process) == FAILURE)
                                                ;
                                } else while (waitproc(process) == FAILURE);
                                process->p_flags = 0;
                                st_write();
                        }
                }
        }

        /* Restore the path. */
        free(glob_envp[0]);
        glob_envp[0] = old_path;

        /*
         * This will enable st_write() to complain about init_state_file.
         */
        booting = 0;

        /*
         * If the /etc/ioctl.syscon didn't exist or had invalid contents write
         * out a correct version.
         */
        if (write_ioctl)
                write_ioctl_syscon();

        /*
         * Start svc.startd(1M), which does most of the work.
         */
        if (startd_cline[0] != ’\0’ && startd_tmpl >= 0) {
                /* Start svc.startd. */
                if (startd_run(startd_cline, startd_tmpl, 0) == -1)
                        cur_state = SINGLE_USER;
        } else {
                console(B_TRUE, "Absent svc.startd entry or bad "
                    "contract template.  Not starting svc.startd.\n");
                enter_maintenance();
        }
}
______unchanged_portion_omitted_

/*
 * prog_name() searches for the word or unix path name and
 * returns a pointer to the last element of the pathname.
 */
static char *
prog_name(char *string)
{
        char    *ptr, *ptr2;
        static char word[UT_USER_SZ + 1];
        /* XXX - utmp - fix name length */
        static char word[_POSIX_LOGIN_NAME_MAX];

        /*
         * Search for the first word skipping leading spaces and tabs.
         */
        while (*string == ’ ’ || *string == ’\t’)
                string++;

        /*
         * If the first non-space non-tab character is not one allowed in
         * a word, return a pointer to a null string, otherwise parse the
         * pathname.
         */
        if (*string != ’.’ && *string != ’/’ && *string != ’_’ &&
            (*string < ’a’ || *string > ’z’) &&
            (*string < ’A’ || * string > ’Z’) &&
            (*string < ’0’ || *string > ’9’))
                return ("");

        /*
         * Parse the pathname looking forward for ’/’, ’ ’, ’\t’, ’\n’ or
         * ’\0’.  Each time a ’/’ is found, move "ptr" to one past the
         * ’/’, thus when a ’ ’, ’\t’, ’\n’, or ’\0’ is found, "ptr" will
         * point to the last element of the pathname.
         */
        for (ptr = string; *string != ’ ’ && *string != ’\t’ &&
            *string != ’\n’ && *string != ’\0’; string++) {
                if (*string == ’/’)
                        ptr = string+1;
        }

        /*
         * Copy out up to the size of the "ut_user" array into "word",
         * null terminate it and return a pointer to it.
         */
        for (ptr2 = &word[0]; ptr2 < &word[UT_USER_SZ] &&
        /* XXX - utmp - fix name length */
        for (ptr2 = &word[0]; ptr2 < &word[_POSIX_LOGIN_NAME_MAX - 1] &&
            ptr < string; /* CSTYLED */)
                *ptr2++ = *ptr++;

        *ptr2 = ’\0’;
        return (&word[0]);
```

```
}
______unchanged_portion_omitted_


/*
 * /etc/inittab has more entries and we have run out of room in the proc_table
 * array. Double the size of proc_table to accomodate the extra entries.
 */
static void
increase_proc_table_size()
{
        sigset_t block, unblock;
        void *ptr;
        size_t delta = num_proc * sizeof (struct PROC_TABLE);


        /*
         * Block signals for realloc.
         */
        (void) sigfillset(&block);
        (void) sigprocmask(SIG_BLOCK, &block, &unblock);


        /*
         * On failure we just return because callers of this function check
         * for failure.
         */
        do
                ptr = realloc(g_state, g_state_sz + delta);
        while (ptr == NULL && errno == EAGAIN)
                ;
        while (ptr == NULL && errno == EAGAIN);

        if (ptr != NULL) {
                /* ensure that the new part is initialized to zero */
                bzero((caddr_t)ptr + g_state_sz, delta);

                g_state = ptr;
                g_state_sz += delta;
                num_proc <<= 1;
        }


        /* unblock our signals before returning */
        (void) sigprocmask(SIG_SETMASK, &unblock, NULL);
}
______unchanged_portion_omitted_

/*
 * Initialize our state.
 *
 * If the system just booted, then init_state_file, which is located on an
 * everpresent tmpfs filesystem, should not exist.
 *
 * If we were restarted, then init_state_file should exist, in
 * which case we’ll read it in, sanity check it, and use it.
 *
 * Note: You can’t call console() until proc_table is ready.
 */
void
st_init()
{
        struct stat stb;
        int ret, st_fd, insane = 0;
        size_t to_be_read;
        char *ptr;
        booting = 1;

        do {
                /*
                 * If we can exclusively create the file, then we’re the
                 * initial invocation of init(1M).
                 */
                st_fd = open(init_state_file, O_RDWR | O_CREAT | O_EXCL,
                    S_IRUSR | S_IWUSR);
        } while (st_fd == -1 && errno == EINTR);
        if (st_fd != -1)
                goto new_state;

        booting = 0;

        do {
                st_fd = open(init_state_file, O_RDWR, S_IRUSR | S_IWUSR);
        } while (st_fd == -1 && errno == EINTR);
        if (st_fd == -1)
                goto new_state;

        /* Get the size of the file. */
        do
                ret = fstat(st_fd, &stb);
        while (ret == -1 && errno == EINTR)
                ;
        while (ret == -1 && errno == EINTR);
        if (ret == -1)
                goto new_state;

        do
                g_state = malloc(stb.st_size);
        while (g_state == NULL && errno == EAGAIN)
                ;
        while (g_state == NULL && errno == EAGAIN);
        if (g_state == NULL)
                goto new_state;

        to_be_read = stb.st_size;
        ptr = (char *)g_state;
        while (to_be_read > 0) {
                ssize_t read_ret;

                read_ret = read(st_fd, ptr, to_be_read);
                if (read_ret < 0) {
                        if (errno == EINTR)
                                continue;

                        goto new_state;
                }

                to_be_read -= read_ret;
                ptr += read_ret;
        }

        (void) close(st_fd);

        g_state_sz = stb.st_size;

        if (st_sane()) {
                console(B_TRUE, "Restarting.\n");
                return;
        }
```

```
        insane = 1;

new_state:
        if (st_fd >= 0)
                (void) close(st_fd);
        else
                (void) unlink(init_state_file);

        if (g_state != NULL)
                free(g_state);

        /* Something went wrong, so allocate new state. */
        g_state_sz = sizeof (struct init_state) +
            ((init_num_proc - 1) * sizeof (struct PROC_TABLE));
        do
                g_state = calloc(1, g_state_sz);
        while (g_state == NULL && errno == EAGAIN)
                ;
        while (g_state == NULL && errno == EAGAIN);
        if (g_state == NULL) {
                /* Fatal error! */
                exit(errno);
        }

        g_state->ist_runlevel = -1;
        num_proc = init_num_proc;

        if (!booting) {
                console(B_TRUE, "Restarting.\n");

                /* Overwrite the bad state file. */
                st_write();

                if (!insane) {
                        console(B_TRUE,
                            "Error accessing persistent state file ‘%s’.  "
                            "Ignored.\n", init_state_file);
                } else {
                        console(B_TRUE,
                            "Persistent state file ‘%s’ is invalid and was "
                            "ignored.\n", init_state_file);
                }
        }
}
______unchanged_portion_omitted_

/*
 * Create a contract with these parameters.
 */
static int
contract_make_template(uint_t info, uint_t critical, uint_t fatal,
    uint64_t cookie)
{
        int fd, err;

        char *ioctl_tset_emsg =
            "Couldn’t set \"%s\" contract template parameter: %s.\n";

        do
                fd = open64(CTFS_ROOT "/process/template", O_RDWR);
        while (fd < 0 && errno == EINTR)
                ;
        while (fd < 0 && errno == EINTR);
        if (fd < 0) {
                console(B_TRUE, "Couldn’t create process template: %s.\n",
                    strerror(errno));
                return (-1);
        }

        if (err = ct_pr_tmpl_set_param(fd, CT_PR_INHERIT | CT_PR_REGENT))
                console(B_TRUE, "Contract set template inherit, regent "
                    "failed: %s.\n", strerror(err));

        /*
         * These errors result in a misconfigured template, which is better
         * than no template at all, so warn but don't abort.
         */
        if (err = ct_tmpl_set_informative(fd, info))
                console(B_TRUE, ioctl_tset_emsg, "informative", strerror(err));

        if (err = ct_tmpl_set_critical(fd, critical))
                console(B_TRUE, ioctl_tset_emsg, "critical", strerror(err));

        if (err = ct_pr_tmpl_set_fatal(fd, fatal))
                console(B_TRUE, ioctl_tset_emsg, "fatal", strerror(err));

        if (err = ct_tmpl_set_cookie(fd, cookie))
                console(B_TRUE, ioctl_tset_emsg, "cookie", strerror(err));

        (void) fcntl(fd, F_SETFD, FD_CLOEXEC);

        return (fd);
}

/*
 * Create the templates and open an event file descriptor.  We use dup2(2) to
 * get these descriptors away from the stdin/stdout/stderr group.
 */
static void
contracts_init()
{
        int err, fd;

        /*
         * Create & configure a legacy template.  We only want empty events so
         * we know when to abandon them.
         */
        legacy_tmpl = contract_make_template(0, CT_PR_EV_EMPTY, CT_PR_EV_HWERR,
            ORDINARY_COOKIE);
        if (legacy_tmpl >= 0) {
                err = ct_tmpl_activate(legacy_tmpl);
                if (err != 0) {
                        (void) close(legacy_tmpl);
                        legacy_tmpl = -1;
                        console(B_TRUE,
                            "Couldn't activate legacy template (%s); "
                            "legacy services will be in init's contract.\n",
                            strerror(err));
                }
        } else
                console(B_TRUE,
                    "Legacy services will be in init's contract.\n");

        if (dup2(legacy_tmpl, 255) == -1) {
                console(B_TRUE, "Could not duplicate legacy template: %s.\n",
                    strerror(errno));
        } else {
                (void) close(legacy_tmpl);
                legacy_tmpl = 255;
        }

        (void) fcntl(legacy_tmpl, F_SETFD, FD_CLOEXEC);
```

```

        startd_tmpl = contract_make_template(0, CT_PR_EV_EMPTY,
            CT_PR_EV_HWERR | CT_PR_EV_SIGNAL | CT_PR_EV_CORE, STARTD_COOKIE);

        if (dup2(startd_tmpl, 254) == -1) {
                console(B_TRUE, "Could not duplicate startd template: %s.\n",
                    strerror(errno));
        } else {
                (void) close(startd_tmpl);
                startd_tmpl = 254;
        }

        (void) fcntl(startd_tmpl, F_SETFD, FD_CLOEXEC);

        if (legacy_tmpl < 0 && startd_tmpl < 0) {
                /* The creation errors have already been reported. */
                console(B_TRUE,
                    "Ignoring contract events.  Core smf(5) services will not "
                    "be restarted.\n");
                return;
        }

        /*
         * Open an event endpoint.
         */
        do
                fd = open64(CTFS_ROOT "/process/pbundle", O_RDONLY);
        while (fd < 0 && errno == EINTR)
                ;
        while (fd < 0 && errno == EINTR);
        if (fd < 0) {
                console(B_TRUE,
                    "Couldn't open process pbundle: %s.  Core smf(5) services "
                    "will not be restarted.\n", strerror(errno));
                return;
        }

        if (dup2(fd, 253) == -1) {
                console(B_TRUE, "Could not duplicate process bundle: %s.\n",
                    strerror(errno));
        } else {
                (void) close(fd);
                fd = 253;
        }

        (void) fcntl(fd, F_SETFD, FD_CLOEXEC);

        /* Reset in case we've been restarted. */
        (void) ct_event_reset(fd);

        poll_fds[0].fd = fd;
        poll_fds[0].events = POLLIN;
        poll_nfds = 1;
}

static int
contract_getfile(ctid_t id, const char *name, int oflag)
{
        int fd;

        do
                fd = contract_open(id, "process", name, oflag);
        while (fd < 0 && errno == EINTR)
                ;
        while (fd < 0 && errno == EINTR);
```

```
	if (fd < 0)
		console(B_TRUE, "Couldn't open %s for contract %ld: %s.\n",
		    name, id, strerror(errno));

	return (fd);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   13232 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/last/last.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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 (c) 2013 Gary Mills
 *
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

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

/*
 * last
 */
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <strings.h>
#include <signal.h>
#include <sys/stat.h>
#include <pwd.h>
#include <fcntl.h>
#include <utmpx.h>
#include <locale.h>
#include <ctype.h>
```

```
/*
 * Use the full lengths from utmpx for NMAX, LMAX and HMAX .
 * NMAX, LMAX and HMAX are set to these values for now. They
 * should be much higher because of the max allowed limit in
 * utmpx.h
 */
#define NMAX    (sizeof (((struct utmpx *)0)->ut_user))
#define LMAX    (sizeof (((struct utmpx *)0)->ut_line))
#define NMAX    8
#define LMAX    12
#define HMAX    (sizeof (((struct utmpx *)0)->ut_host))

/* Print minimum field widths. */
#define LOGIN_WIDTH     8
#define LINE_WIDTH      12

#define SECDAY  (24*60*60)
#define CHUNK_SIZE 256

#define lineq(a, b)     (strncmp(a, b, LMAX) == 0)
#define nameq(a, b)     (strncmp(a, b, NMAX) == 0)
#define hosteq(a, b)    (strncmp(a, b, HMAX) == 0)
#define linehostnameq(a, b, c, d) \
            (lineq(a, b)&&hosteq(a+LMAX+1, c)&&nameq(a+LMAX+HMAX+2, d))

#define USAGE   "usage: last [-n number] [-f filename] [-a ] [name | tty] ...\n"

/* Beware: These are set in main() to exclude the executable name.  */
static char     **argv;
static int      argc;
static char     **names;
static int      names_num;

static struct   utmpx buf[128];

/*
 * ttnames and logouts are allocated in the blocks of
 * CHUNK_SIZE lines whenever needed. The count of the
 * current size is maintained in the variable "lines"
 * The variable bootxtime is used to hold the time of
 * the last BOOT_TIME
 * All elements of the logouts are initialised to bootxtime
 * everytime the buffer is reallocated.
 */

static char     **ttnames;
static time_t   *logouts;
static time_t   bootxtime;
static int      lines;
static char     timef[128];
static char     hostf[HMAX + 1];

static char *strspl(char *, char *);
static void onintr(int);
static void reallocate_buffer();
static void memory_alloc(int);
static int want(struct utmpx *, char **, char **);
static void record_time(time_t *, int *, int, struct utmpx *);

int
main(int ac, char **av)
{
        int i, j;
        int aflag = 0;
        int fpos;       /* current position in time format buffer */
        int chrcnt;     /* # of chars formatted by current sprintf */
        int bl, wtmp;
        char *ct;
        char *ut_host;
        char *ut_user;
        struct utmpx *bp;
        time_t otime;
        struct stat stb;
        int print = 0;
        char *crmsg = (char *)0;
        long outrec = 0;
        long maxrec = 0x7fffffffL;
        char *wtmpfile = "/var/adm/wtmpx";
        size_t hostf_len;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)               /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"          /* Use this only if it weren't. */
#endif
        (void) textdomain(TEXT_DOMAIN);

        (void) time(&buf[0].ut_xtime);
        ac--, av++;
        argc = ac;
        argv = av;
        names = malloc(argc * sizeof (char *));
        if (names == NULL) {
                perror("last");
                exit(2);
        }
        names_num = 0;
        for (i = 0; i < argc; i++) {
                if (argv[i][0] == '-') {

                        /* -[0-9]*   sets max # records to print */
                        if (isdigit(argv[i][1])) {
                                maxrec = atoi(argv[i]+1);
                                continue;
                        }

                        for (j = 1; argv[i][j] != '\0'; ++j) {
                                switch (argv[i][j]) {

                                /* -f name sets filename of wtmp file */
                                case 'f':
                                        if (argv[i][j+1] != '\0') {
                                                wtmpfile = &argv[i][j+1];
                                        } else if (i+1 < argc) {
                                                wtmpfile = argv[++i];
                                        } else {
                                                (void) fprintf(stderr,
                                                    gettext("last: argument to "
                                                    "-f is missing\n"));
                                                (void) fprintf(stderr,
                                                    gettext(USAGE));
                                                exit(1);
                                        }
                                        goto next_word;

                                /* -n number sets max # records to print */
                                case 'n': {
                                        char *arg;

                                        if (argv[i][j+1] != '\0') {
                                                arg = &argv[i][j+1];
                                        } else if (i+1 < argc) {
```

```
                                                arg = argv[++i];
                                        } else {
                                                (void) fprintf(stderr,
                                                    gettext("last: argument to "
                                                    "-n is missing\n"));
                                                (void) fprintf(stderr,
                                                    gettext(USAGE));
                                                exit(1);
                                        }

                                        if (!isdigit(*arg)) {
                                                (void) fprintf(stderr,
                                                    gettext("last: argument to "
                                                    "-n is not a number\n"));
                                                (void) fprintf(stderr,
                                                    gettext(USAGE));
                                                exit(1);
                                        }
                                        maxrec = atoi(arg);
                                        goto next_word;
                                }

                                /* -a displays hostname last on the line */
                                case 'a':
                                        aflag++;
                                        break;

                                default:
                                        (void) fprintf(stderr, gettext(USAGE));
                                        exit(1);
                                }
                        }

next_word:
                        continue;
                }

                if (strlen(argv[i]) > 2 || strcmp(argv[i], "~") == 0 ||
                    getpwnam(argv[i]) != NULL) {
                        /* Not a tty number. */
                        names[names_num] = argv[i];
                        ++names_num;
                } else {
                        /* tty number.  Prepend "tty". */
                        names[names_num] = strspl("tty", argv[i]);
                        ++names_num;
                }
        }

        wtmp = open(wtmpfile, 0);
        if (wtmp < 0) {
                perror(wtmpfile);
                exit(1);
        }
        (void) fstat(wtmp, &stb);
        bl = (stb.st_size + sizeof (buf)-1) / sizeof (buf);
        if (signal(SIGINT, SIG_IGN) != SIG_IGN) {
                (void) signal(SIGINT, onintr);
                (void) signal(SIGQUIT, onintr);
        }
        lines = CHUNK_SIZE;
        ttnames = calloc(lines, sizeof (char *));
        logouts = calloc(lines, sizeof (time_t));
        if (ttnames == NULL || logouts == NULL) {
                (void) fprintf(stderr, gettext("Out of memory \n "));
                exit(2);
        }
                for (bl--; bl >= 0; bl--) {
                (void) lseek(wtmp, (off_t)(bl * sizeof (buf)), 0);
                bp = &buf[read(wtmp, buf, sizeof (buf)) / sizeof (buf[0]) - 1];
                for (; bp >= buf; bp--) {
                        if (want(bp, &ut_host, &ut_user)) {
                                for (i = 0; i <= lines; i++) {
                                if (i == lines)
                                        reallocate_buffer();
                                if (ttnames[i] == NULL) {
                                        memory_alloc(i);
                                        /*
                                         * LMAX+HMAX+NMAX+3 bytes have been
                                         * allocated for ttnames[i].
                                         * If bp->ut_line is longer than LMAX,
                                         * ut_host is longer than HMAX,
                                         * and ut_user is longer than NMAX,
                                         * truncate it to fit ttnames[i].
                                         */
                                        (void) strlcpy(ttnames[i], bp->ut_line,
                                            LMAX+1);
                                        (void) strlcpy(ttnames[i]+LMAX+1,
                                            ut_host, HMAX+1);
                                        (void) strlcpy(ttnames[i]+LMAX+HMAX+2,
                                            ut_user, NMAX+1);
                                                record_time(&otime, &print,
                                                    i, bp);
                                                break;
                                        } else if (linehostnameq(ttnames[i],
                                            bp->ut_line, ut_host, ut_user)) {
                                                record_time(&otime,
                                                    &print, i, bp);
                                                break;
                                        }
                                }
                        }
                        if (print) {
                                if (strncmp(bp->ut_line, "ftp", 3) == 0)
                                        bp->ut_line[3] = '\0';
                                if (strncmp(bp->ut_line, "uucp", 4) == 0)
                                        bp->ut_line[4] = '\0';

                                ct = ctime(&bp->ut_xtime);
                                (void) printf(gettext("%-*.*s  %-*.*s "),
                                    LOGIN_WIDTH, NMAX, bp->ut_name,
                                    LINE_WIDTH, LMAX, bp->ut_line);
                                    NMAX, NMAX, bp->ut_name,
                                    LMAX, LMAX, bp->ut_line);
                                hostf_len = strlen(bp->ut_host);
                                (void) snprintf(hostf, sizeof (hostf),
                                    "%-*.*s", hostf_len, hostf_len,
                                    bp->ut_host);
                                fpos = snprintf(timef, sizeof (timef),
                                    "%10.10s %5.5s ",
                                    ct, 11 + ct);
                                if (!lineq(bp->ut_line, "system boot") &&
                                    !lineq(bp->ut_line, "system down")) {
                                        if (otime == 0 &&
                                            bp->ut_type == USER_PROCESS) {

        if (fpos < sizeof (timef)) {
                /* timef still has room */
                (void) snprintf(timef + fpos, sizeof (timef) - fpos,
                    gettext("  still logged in"));
        }
```

```
                                        } else {
                                        time_t delta;
                                        if (otime < 0) {
                                                otime = -otime;
                                                /*
                                                 * TRANSLATION_NOTE
                                                 * See other notes on "down"
                                                 * and "- %5.5s".
                                                 * "-" means "until".  This
                                                 * is displayed after the
                                                 * starting time as in:
                                                 *      16:20 - down
                                                 * You probably don't want to
                                                 * translate this.  Should you
                                                 * decide to translate this,
                                                 * translate "- %5.5s" too.
                                                 */

        if (fpos < sizeof (timef)) {
                /* timef still has room */
                chrcnt = snprintf(timef + fpos, sizeof (timef) - fpos,
                    gettext("- %s"), crmsg);
                fpos += chrcnt;
        }

                                        } else {

        if (fpos < sizeof (timef)) {
                /* timef still has room */
                chrcnt = snprintf(timef + fpos, sizeof (timef) - fpos,
                    gettext("- %5.5s"), ctime(&otime) + 11);
                fpos += chrcnt;
        }

                                        }
                                        delta = otime - bp->ut_xtime;
                                        if (delta < SECDAY) {

        if (fpos < sizeof (timef)) {
                /* timef still has room */
                (void) snprintf(timef + fpos, sizeof (timef) - fpos,
                    gettext("  (%5.5s)"), asctime(gmtime(&delta)) + 11);
        }

                                        } else {

        if (fpos < sizeof (timef)) {
                /* timef still has room */
                (void) snprintf(timef + fpos, sizeof (timef) - fpos,
                    gettext(" (%ld+%5.5s)"), delta / SECDAY,
                    asctime(gmtime(&delta)) + 11);
        }

                                        }
                                }
                                }
                                if (aflag)
                                        (void) printf("%-35.35s %-.*s\n",
                                            timef, strlen(hostf), hostf);
                                else
                                        (void) printf("%-16.16s %-.35s\n",
                                            hostf, timef);
                                (void) fflush(stdout);
                                if (++outrec >= maxrec)
                                        exit(0);
                        }
                        /*
                         * when the system is down or crashed.
                         */
                        if (bp->ut_type == BOOT_TIME) {
                                for (i = 0; i < lines; i++)
                                        logouts[i] = -bp->ut_xtime;
                                bootxtime = -bp->ut_xtime;
                                /*
                                 * TRANSLATION_NOTE
                                 * Translation of this "down " will replace
                                 * the %s in "- %s".  "down" is used instead
                                 * of the real time session was ended, probably
                                 * because the session ended by a sudden crash.
                                 */
                                crmsg = gettext("down ");
                        }
                        print = 0;      /* reset the print flag */
                }
        }
        ct = ctime(&buf[0].ut_xtime);
        (void) printf(gettext("\nwtmp begins %10.10s %5.5s \n"), ct, ct + 11);

        /* free() called to prevent lint warning about names */
        free(names);

        return (0);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   21508 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/newtask/newtask.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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 (c) 2013 Gary Mills
 *
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

#include <sys/types.h>
#include <sys/task.h>

#include <alloca.h>
#include <libproc.h>
#include <libintl.h>
#include <libgen.h>
#include <limits.h>
#include <project.h>
#include <pwd.h>
#include <secdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/varargs.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <priv_utils.h>

#include "utils.h"

#define OPTIONS_STRING  "Fc:lp:v"
#define NENV            8
#define ENVSIZE         255
#define PATH            "PATH=/usr/bin"
#define SUPATH          "PATH=/usr/sbin:/usr/bin"
#define SHELL           "/usr/bin/sh"
#define SHELL2          "/sbin/sh"
#define TIMEZONEFILE    "/etc/default/init"
```

```
#define LOGINFILE       "/etc/default/login"
#define GLOBAL_ERR_SZ   1024
#define GRAB_RETRY_MAX  100

static const char *pname;
extern char **environ;
static char *supath = SUPATH;
static char *path = PATH;
static char global_error[GLOBAL_ERR_SZ];
static int verbose = 0;

static priv_set_t *nset;

/* Private definitions for libproject */
extern projid_t setproject_proc(const char *, const char *, int, pid_t,
    struct ps_prochandle *, struct project *);
extern priv_set_t *setproject_initpriv(void);

static void usage(void);

static void preserve_error(const char *format, ...);

static int update_running_proc(int, char *, char *);
static int set_ids(struct ps_prochandle *, struct project *,
    struct passwd *);
static struct passwd *match_user(uid_t, char *, int);
static void setproject_err(char *, char *, int, struct project *);

static void
usage(void)
{
        (void) fprintf(stderr, gettext("usage: \n\t%s [-v] [-p project] "
            "[-c pid | [-Fl] [command [args ...]]]\n"), pname);
        exit(2);
}
______unchanged_portion_omitted_

/*
 * Given the input arguments, return the passwd structure that matches best.
 * Also, since we use getpwnam() and friends, subsequent calls to this
 * function will re-use the memory previously returned.
 */
static struct passwd *
match_user(uid_t uid, char *projname, int is_my_uid)
{
        char prbuf[PROJECT_BUFSZ], username[LOGNAME_MAX_ILLUMOS+1];
        char prbuf[PROJECT_BUFSZ], username[LOGNAME_MAX+1];
        struct project prj;
        char *tmp_name;
        struct passwd *pw = NULL;

        /*
         * In order to allow users with the same UID but distinguishable
         * user names to be in different projects we play a guessing
         * game of which username is most appropriate. If we're checking
         * for the uid of the calling process, the login name is a
         * good starting point.
         */
        if (is_my_uid) {
                if ((tmp_name = getlogin()) == NULL ||
                    (pw = getpwnam(tmp_name)) == NULL || (pw->pw_uid != uid) ||
                    (pw->pw_name == NULL))
                        pw = NULL;
        }

        /*
```

```
	 * If the login name doesn't work,  we try the first match for
	 * the current uid in the password file.
	 */
	if (pw == NULL) {
		if (((pw = getpwuid(uid)) == NULL) || pw->pw_name == NULL) {
			preserve_error(gettext("cannot find username "
			    "for uid %d"), uid);
			return (NULL);
		}
	}

	/*
	 * If projname wasn't supplied, we've done our best, so just return
	 * what we've got now. Alternatively, if newtask's invoker has
	 * superuser privileges, return the pw structure we've got now, with
	 * no further checking from inproj(). Superuser should be able to
	 * join any project, and the subsequent call to setproject() will
	 * allow this.
	 */
	if (projname == NULL || getuid() == (uid_t)0)
		return (pw);

	(void) strncpy(username, pw->pw_name, sizeof (username) - 1);
	username[sizeof (username) - 1] = '\0';
	(void) strcpy(username, pw->pw_name);

	if (inproj(username, projname, prbuf, PROJECT_BUFSZ) == 0) {
		char **u;
		tmp_name = NULL;

		/*
		 * If the previous guesses didn't work, walk through all
		 * project members and test for UID-equivalence.
		 */

		if (getprojbyname(projname, &prj, prbuf,
		    PROJECT_BUFSZ) == NULL) {
			preserve_error(gettext("unknown project \"%s\""),
			    projname);
			return (NULL);
		}

		for (u = prj.pj_users; *u; u++) {
			if ((pw = getpwnam(*u)) == NULL)
				continue;

			if (pw->pw_uid == uid) {
				tmp_name = pw->pw_name;
				break;
			}
		}

		if (tmp_name == NULL) {
			preserve_error(gettext("user \"%s\" is not a member of "
			    "project \"%s\""), username, projname);
			return (NULL);
		}
	}

	return (pw);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
    2641 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/oamuser/inc/users.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 */

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


#ifndef _USERS_H
#define _USERS_H


#include <pwd.h>
#include <grp.h>
#include <project.h>

#define GROUP           "/etc/group"

/* max number of projects that can be specified when adding a user */
#define NPROJECTS_MAX   1024

/* validation returns */
#define NOTUNIQUE       0       /* not unique */
#define RESERVED        1       /* reserved */
#define UNIQUE          2       /* is unique */
#define TOOBIG          3       /* number too big */
#define INVALID         4
#define LONGNAME        5       /* string too long */

/*
 * Note: constraints checking for warning (release 2.6),
 * and these may be enforced in the future releases.
 */
#define WARN_NAME_TOO_LONG      0x1
#define WARN_BAD_GROUP_NAME     0x2
#define WARN_BAD_LOGNAME_CHAR   0x4
#define WARN_BAD_LOGNAME_FIRST  0x8
#define WARN_NO_LOWERCHAR       0x10
```

```
#define WARN_BAD_PROJ_NAME      0x20
#define WARN_LOGGED_IN          0x40

/* Exit codes from passmgmt */
#define PEX_SUCCESS     0
#define PEX_NO_PERM     1
#define PEX_SYNTAX      2
#define PEX_BADARG      3
#define PEX_BADUID      4
#define PEX_HOSED_FILES 5
#define PEX_FAILED      6
#define PEX_MISSING     7
#define PEX_BUSY        8
#define PEX_BADNAME     9

#define REL_PATH(x)     (x && *x != '/')

/*
 * interfaces available from the library
 */
extern int valid_login(char *, struct passwd **, int *);
extern int valid_gname(char *, struct group **, int *);
extern int valid_group(char *, struct group **, int *);
extern int valid_project(char *, struct project *, void *buf, size_t, int *);
extern int valid_projname(char *, struct project *, void *buf, size_t, int *);
extern void warningmsg(int, char *);
extern void putgrent(struct group *, FILE *);

/* passmgmt */
#define PASSMGMT        "/usr/lib/passmgmt";
#endif  /* _USERS_H */
```

```
************************************************************
    2115 Wed Apr  3 09:33:10 2013
new/usr/src/cmd/oamuser/lib/vlogin.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
************************************************************
```

```
/*
 * 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 (c) 2013 Gary Mills
 *
 * Copyright (c) 1997, by Sun Microsystems, Inc.
 * All rights reserved.
 */

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


#pragma ident   "%Z%%M% %I%     %E% SMI"        /* SVr4.0 1.3 */

/*LINTLIBRARY*/

#include        <sys/types.h>
#include        <stdio.h>
#include        <ctype.h>
#include        <userdefs.h>
#include        <users.h>
#include        <limits.h>

/*
 * validate string given as login name.
 */
int
valid_login(char *login, struct passwd **pptr, int *warning)
{
        struct passwd *t_pptr;
        char *ptr = login;
        int bad1char, badc, clower, len;
        char c;

        len = 0; clower = 0; badc = 0; bad1char = 0;
        *warning = 0;
        if (!login || !*login)
                return (INVALID);

        c = *ptr;
```

```
        if (!isalpha(c))
                bad1char++;
        for (; c != NULL; ptr++, c = *ptr) {
                len++;
                if (!isprint(c) || (c == ':') || (c == '\n'))
                        return (INVALID);
                if (!isalnum(c) && c != '_' && c != '-' && c != '.')
                        badc++;
                if (islower(c))
                        clower++;
        }

        if (len > LOGNAME_MAX_ILLUMOS)
                return (LONGNAME);

        /*
         * XXX length checking causes some operational/compatibility problem.
         * This has to be revisited in the future as ARC/standards issue.
         */
        if (len > LOGNAME_MAX)
                *warning = *warning | WARN_NAME_TOO_LONG;
        if (clower == 0)
                *warning = *warning | WARN_NO_LOWERCHAR;
        if (badc != 0)
                *warning = *warning | WARN_BAD_LOGNAME_CHAR;
        if (bad1char != 0)
                *warning = *warning | WARN_BAD_LOGNAME_FIRST;

        if ((t_pptr = getpwnam(login)) != NULL) {
                if (pptr) *pptr = t_pptr;
                return (NOTUNIQUE);
        }
        return (UNIQUE);
}
```

______unchanged_portion_omitted_

```
**********************************************************
    4796 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/oamuser/user/messages.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */


/*
 * Copyright (c) 2013 Gary Mills
 *
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"        /* SVr4.0 1.6 */

char *errmsgs[] = {
        "WARNING: uid %ld is reserved.\n",
        "WARNING: more than NGROUPS_MAX(%d) groups specified.\n",
        "ERROR: invalid syntax.\n"
            "usage:  useradd [-u uid [-o] | -g group | -G group[[,group]...] |"
            "-d dir | -b base_dir |\n"
            "\t\t-s shell | -c comment | -m [-k skel_dir] | -f inactive |\n"
            "\t\t-e expire | -A authorization [, authorization ...] |\n"
            "\t\t-P profile [, profile ...] | -R role [, role ...] |\n"
            "\t\t-K key=value | -p project [, project ...]] login\n"
            "\tuseradd -D [-g group | -b base_dir | -f inactive | -e expire\n"
            "\t\t-A authorization [, authorization ...] |\n"
            "\t\t-P profile [, profile ...] | -R role [, role ...] |\n"
            "\t\t-K key=value ... -p project] | [-s shell] | [-k skel_dir]\n",
        "ERROR: Invalid syntax.\nusage:  userdel [-r] login\n",
        "ERROR: Invalid syntax.\n"
            "usage:  usermod -u uid [-o] | -g group | -G group[[,group]...] |\n"
            "\t\t-d dir [-m] | -s shell | -c comment |\n"
            "\t\t-l new_logname | -f inactive | -e expire |\n"
            "\t\t-A authorization [, authorization ...] | -K key=value ... |\n"
            "\t\t-P profile [, profile ...] | -R role [, role ...] login\n",
        "ERROR: Unexpected failure.  Defaults unchanged.\n",
        "ERROR: Unable to remove files from home directory.\n",
        "ERROR: Unable to remove home directory.\n",
        "ERROR: Cannot update system files - login cannot be %s.\n",
        "ERROR: uid %ld is already in use.  Choose another.\n",
        "ERROR: %s is already in use.  Choose another.\n",
        "ERROR: %s does not exist.\n",
        "ERROR: %s is not a valid %s.  Choose another.\n",
        "ERROR: %s is in use.  Cannot %s it.\n",
        "WARNING: %s has no permissions to use %s.\n",
        "ERROR: There is not sufficient space to move %s home directory to %s"
            "\n",
        "ERROR: %s %ld is too big.  Choose another.\n",
        "ERROR: group %s does not exist.  Choose another.\n",
        "ERROR: Unable to %s: %s.\n",
        "ERROR: %s is not a full path name.  Choose another.\n",
        "ERROR: %s is the primary group name.  Choose another.\n",
        "ERROR: Inconsistent password files.  See pwconv(1M).\n",
        "ERROR: %s is not a local user.\n",
        "ERROR: Permission denied.\n",
        "WARNING: Group entry exceeds 2048 char: /etc/group entry truncated.\n",
        "ERROR: invalid syntax.\n"
            "usage:  roleadd [-u uid [-o] | -g group | -G group[[,group]...] |"
            "-d dir |\n"
            "\t\t-s shell | -c comment | -m [-k skel_dir] | -f inactive |\n"
            "\t\t-e expire | -A authorization [, authorization ...] |\n"
            "\t\t-P profile [, profile ...] | -K key=value ] login\n"
            "\troleadd -D [-g group | -b base_dir | -f inactive | -e expire\n"
            "\t\t-A authorization [, authorization ...] |\n"
            "\t\t-P profile [, profile ...]]\n",
        "ERROR: Invalid syntax.\nusage:  roledel [-r] login\n",
        "ERROR: Invalid syntax.\n"
            "usage:  rolemod -u uid [-o] | -g group | -G group[[,group]...] |\n"
            "\t\t-d dir [-m] | -s shell | -c comment |\n"
            "\t\t-l new_logname | -f inactive | -e expire |\n"
            "\t\t-A authorization [, authorization ...] | -K key=value |\n"
            "\t\t-P profile [, profile ...] login\n",
        "ERROR: project %s does not exist.  Choose another.\n",
        "WARNING: more than NPROJECTS_MAX(%d) projects specified.\n",
        "WARNING: Project entry exceeds %d char: /etc/project entry truncated."
            "\n",
        "ERROR: Invalid key.\n",
        "ERROR: Missing value specification.\n",
        "ERROR: Multiple definitions of key ‘‘%s’’.\n",
        "ERROR: Roles must be modified with ‘‘rolemod’’.\n",
        "ERROR: Users must be modified with ‘‘usermod’’.\n",
        "WARNING: gid %ld is reserved.\n",
        "ERROR: Failed to read /etc/group file due to invalid entry or"
            " read error.\n",
        "ERROR: %s is too long.  Choose another.\n",
};
______unchanged_portion_omitted_
```

```
**********************************************************
    4075 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/oamuser/user/messages.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */


/*
 * Copyright (c) 2013 Gary Mills
 *
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _MESSAGES_H
#define _MESSAGES_H

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

extern void errmsg(int, ...);

/* WARNING: uid %d is reserved. */
#define M_RESERVED              0

/* WARNING: more than NGROUPS_MAX(%d) groups specified. */
#define M_MAXGROUPS     1

/* ERROR: invalid syntax.\nusage:  useradd ... */
#define M_AUSAGE                2

/* ERROR: Invalid syntax.\nusage:  userdel [-r] login\n" */
#define M_DUSAGE                3

/* ERROR: Invalid syntax.\nusage:  usermod ... */
#define M_MUSAGE                4


/* ERROR: Unexpected failure.  Defaults unchanged. */
#define M_FAILED        5

/* ERROR: Unable to remove files from home directory. */
#define M_RMFILES       6
```

```
/* ERROR: Unable to remove home directory. */
#define M_RMHOME                7

/* ERROR: Cannot update system files - login cannot be %s. */
#define M_UPDATE                8

/* ERROR: uid %d is already in use.  Choose another. */
#define M_UID_USED      9

/* ERROR: %s is already in use.  Choose another. */
#define M_USED  10

/* ERROR: %s does not exist. */
#define M_EXIST 11

/* ERROR: %s is not a valid %s.  Choose another. */
#define M_INVALID               12

/* ERROR: %s is in use.  Cannot %s it. */
#define M_BUSY  13

/* WARNING: %s has no permissions to use %s. */
#define M_NO_PERM       14

/* ERROR: There is not sufficient space to move %s home directory to %s */
#define M_NOSPACE               15

/* ERROR: %s %d is too big.  Choose another. */
#define M_TOOBIG        16

/* ERROR: group %s does not exist.  Choose another. */
#define M_GRP_NOTUSED   17

/* ERROR: Unable to %s: %s */
#define M_OOPS  18

/* ERROR: %s is not a full path name.  Choose another. */
#define M_RELPATH       19

/* ERROR: %s is the primary group name.  Choose another. */
#define M_SAME_GRP      20

/* ERROR: Inconsistent password files.  See pwconv(1M). */
#define M_HOSED_FILES   21

/* ERROR: %s is not a local user. */
#define M_NONLOCAL      22

/* ERROR: Permission denied. */
#define M_PERM_DENIED   23

/* WARNING: Group entry exceeds 2048 char: /etc/group entry truncated. */
#define M_GROUP_ENTRY_OVF  24

/* ERROR: invalid syntax.\nusage:  roleadd ... */
#define M_ARUSAGE               25

/* ERROR: Invalid syntax.\nusage:  roledel [-r] login\n" */
#define M_DRUSAGE               26

/* ERROR: Invalid syntax.\nusage:  rolemod -u ... */
#define M_MRUSAGE               27

/* ERROR: project %s does not exist.  Choose another. */
#define M_PROJ_NOTUSED 28
```

```
/* WARNING: more than NPROJECTS_MAX(%d) projects specified. */
#define M_MAXPROJECTS   29

/* WARNING: Project entry exceeds 512 char: /etc/project entry truncated. */
#define M_PROJ_ENTRY_OVF  30

/* ERROR: Invalid key. */
#define M_INVALID_KEY   31

/* ERROR: Missing value specification. */
#define M_INVALID_VALUE 32

/* ERROR: Multiple definitions of key ‘‘%s’’. */
#define M_REDEFINED_KEY 33

/* ERROR: Roles must be modified with rolemod */
#define M_ISROLE        34

/* ERROR: Users must be modified with usermod */
#define M_ISUSER        35

/* WARNING: gid %d is reserved. */
#define M_RESERVED_GID          36

/* ERROR: Failed to read /etc/group file due to invalid entry or read error. */
#define M_READ_ERROR    37

/* ERROR: %s is too long.  Choose another. */
#define M_TOO_LONG      38

#endif /* _MESSAGES_H */
```

```
**********************************************************
   17433 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/oamuser/user/useradd.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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


#include        <sys/types.h>
#include        <sys/stat.h>
#include        <sys/param.h>
#include        <stdio.h>
#include        <stdlib.h>
#include        <ctype.h>
#include        <limits.h>
#include        <string.h>
#include        <userdefs.h>
#include        <errno.h>
#include        <project.h>
#include        <unistd.h>
#include        <user_attr.h>
#include        "users.h"
#include        "messages.h"
#include        "userdisp.h"
#include        "funcs.h"

/*
 *  useradd [-u uid [-o] | -g group | -G group [[, group]...] | -d dir [-m]
 *              | -s shell | -c comment | -k skel_dir | -b base_dir] ]
 *              [ -A authorization [, authorization ...]]
 *              [ -P profile [, profile ...]]
 *              [ -K key=value ]
 *              [ -R role [, role ...]] [-p project [, project ...]] login
 *  useradd -D [ -g group ] [ -b base_dir | -f inactive | -e expire |
 *              -s shell | -k skel_dir ]
 *              [ -A authorization [, authorization ...]]
 *              [ -P profile [, profile ...]] [ -K key=value ]
```

```
 *              [ -R role [, role ...]] [-p project [, project ...]] login
 *
 *      This command adds new user logins to the system.  Arguments are:
 *
 *      uid - an integer
 *      group - an existing group's integer ID or char string name
 *      dir - home directory
 *      shell - a program to be used as a shell
 *      comment - any text string
 *      skel_dir - a skeleton directory
 *      base_dir - a directory
 *      login - a string of printable chars except colon(:)
 *      authorization - One or more comma separated authorizations defined
 *                      in auth_attr(4).
 *      profile - One or more comma separated execution profiles defined
 *                in prof_attr(4)
 *      role - One or more comma-separated role names defined in user_attr(4)
 *      project - One or more comma-separated project names or numbers
 *
 */

extern struct userdefs *getusrdef();
extern void dispusrdef();

static void cleanup();

extern uid_t findnextuid(void);
extern int check_perm(), valid_expire();
extern int putusrdef(), valid_uid();
extern int call_passmgmt(), edit_group(), create_home();
extern int edit_project();
extern int **valid_lgroup();
extern projid_t **valid_lproject();
extern void update_def(struct userdefs *);
extern void import_def(struct userdefs *);

static uid_t uid;                       /* new uid */
static char *logname;                   /* login name to add */
static struct userdefs *usrdefs;        /* defaults for useradd */

char *cmdname;

static char homedir[ PATH_MAX + 1 ];    /* home directory */
static char gidstring[32];              /* group id string representation */
static gid_t gid;                       /* gid of new login */
static char uidstring[32];              /* user id string representation */
static char *uidstr = NULL;             /* uid from command line */
static char *base_dir = NULL;           /* base_dir from command line */
static char *group = NULL;              /* group from command line */
static char *grps = NULL;               /* multi groups from command line */
static char *dir = NULL;                /* home dir from command line */
static char *shell = NULL;              /* shell from command line */
static char *comment = NULL;            /* comment from command line */
static char *skel_dir = NULL;           /* skel dir from command line */
static long inact;                      /* inactive days */
static char *inactstr = NULL;           /* inactive from command line */
static char inactstring[10];            /* inactivity string representation */
static char *expirestr = NULL;          /* expiration date from command line */
static char *projects = NULL;           /* project id's from command line */

static char *usertype = NULL;   /* type of user, either role or normal */

typedef enum {
        BASEDIR = 0,
        SKELDIR,
        SHELL
```

```c
} path_opt_t;


static void valid_input(path_opt_t, const char *);

int
main(argc, argv)
int argc;
char *argv[];
{
        int ch, ret, mflag = 0, oflag = 0, Dflag = 0, **gidlist;
        projid_t **projlist;
        char *ptr;                      /* loc in a str, may be set by strtol */
        struct group *g_ptr;
        struct project p_ptr;
        char mybuf[PROJECT_BUFSZ];
        struct stat statbuf;            /* status buffer for stat */
        int warning;
        int busy = 0;
        char **nargv;                   /* arguments for execvp of passmgmt */
        int argindex;                   /* argument index into nargv */

        cmdname = argv[0];

        if (geteuid() != 0) {
                errmsg(M_PERM_DENIED);
                exit(EX_NO_PERM);
        }

        opterr = 0;                     /* no print errors from getopt */
        usertype = getusertype(argv[0]);

        change_key(USERATTR_TYPE_KW, usertype);

        while ((ch = getopt(argc, argv,
                    "b:c:Dd:e:f:G:g:k:mop:s:u:A:P:R:K:")) != EOF)
                switch (ch) {
                case 'b':
                        base_dir = optarg;
                        break;

                case 'c':
                        comment = optarg;
                        break;

                case 'D':
                        Dflag++;
                        break;

                case 'd':
                        dir = optarg;
                        break;

                case 'e':
                        expirestr = optarg;
                        break;

                case 'f':
                        inactstr = optarg;
                        break;

                case 'G':
                        grps = optarg;
                        break;

                case 'g':
```

```c
                        group = optarg;
                        break;

                case 'k':
                        skel_dir = optarg;
                        break;

                case 'm':
                        mflag++;
                        break;

                case 'o':
                        oflag++;
                        break;

                case 'p':
                        projects = optarg;
                        break;

                case 's':
                        shell = optarg;
                        break;

                case 'u':
                        uidstr = optarg;
                        break;

                case 'A':
                        change_key(USERATTR_AUTHS_KW, optarg);
                        break;

                case 'P':
                        change_key(USERATTR_PROFILES_KW, optarg);
                        break;

                case 'R':
                        if (is_role(usertype)) {
                                errmsg(M_ARUSAGE);
                                exit(EX_SYNTAX);
                        }
                        change_key(USERATTR_ROLES_KW, optarg);
                        break;

                case 'K':
                        change_key(NULL, optarg);
                        break;

                default:
                case '?':
                        if (is_role(usertype))
                                errmsg(M_ARUSAGE);
                        else
                                errmsg(M_AUSAGE);
                        exit(EX_SYNTAX);
                }

        /* get defaults for adding new users */
        usrdefs = getusrdef(usertype);

        if (Dflag) {
                /* DISPLAY mode */

                /* check syntax */
                if (optind != argc) {
                        if (is_role(usertype))
                                errmsg(M_ARUSAGE);
```

```
			else
				errmsg(M_AUSAGE);
			exit(EX_SYNTAX);
		}

		if (uidstr != NULL || oflag || grps != NULL ||
		    dir != NULL || mflag || comment != NULL) {
			if (is_role(usertype))
				errmsg(M_ARUSAGE);
			else
				errmsg(M_AUSAGE);
			exit(EX_SYNTAX);
		}

		/* Group must be an existing group */
		if (group != NULL) {
			switch (valid_group(group, &g_ptr, &warning)) {
			case INVALID:
				errmsg(M_INVALID, group, "group id");
				exit(EX_BADARG);
				/*NOTREACHED*/
			case TOOBIG:
				errmsg(M_TOOBIG, "gid", group);
				exit(EX_BADARG);
				/*NOTREACHED*/
			case RESERVED:
			case UNIQUE:
				errmsg(M_GRP_NOTUSED, group);
				exit(EX_NAME_NOT_EXIST);
			}
			if (warning)
				warningmsg(warning, group);

			usrdefs->defgroup = g_ptr->gr_gid;
			usrdefs->defgname = g_ptr->gr_name;

		}

		/* project must be an existing project */
		if (projects != NULL) {
			switch (valid_project(projects, &p_ptr, mybuf,
			    sizeof (mybuf), &warning)) {
			case INVALID:
				errmsg(M_INVALID, projects, "project id");
				exit(EX_BADARG);
				/*NOTREACHED*/
			case TOOBIG:
				errmsg(M_TOOBIG, "projid", projects);
				exit(EX_BADARG);
				/*NOTREACHED*/
			case UNIQUE:
				errmsg(M_PROJ_NOTUSED, projects);
				exit(EX_NAME_NOT_EXIST);
			}
			if (warning)
				warningmsg(warning, projects);

			usrdefs->defproj = p_ptr.pj_projid;
			usrdefs->defprojname = p_ptr.pj_name;
		}

		/* base_dir must be an existing directory */
		if (base_dir != NULL) {
			valid_input(BASEDIR, base_dir);
			usrdefs->defparent = base_dir;
		}
```

```
		/* inactivity period is an integer */
		if (inactstr != NULL) {
			/* convert inactstr to integer */
			inact = strtol(inactstr, &ptr, 10);
			if (*ptr || inact < 0) {
				errmsg(M_INVALID, inactstr,
				    "inactivity period");
				exit(EX_BADARG);
			}

			usrdefs->definact = inact;
		}

		/* expiration string is a date, newer than today */
		if (expirestr != NULL) {
			if (*expirestr) {
				if (valid_expire(expirestr, (time_t *)0)
				    == INVALID) {
					errmsg(M_INVALID, expirestr,
					    "expiration date");
					exit(EX_BADARG);
				}
				usrdefs->defexpire = expirestr;
			} else
				/* Unset the expiration date */
				usrdefs->defexpire = "";
		}

		if (shell != NULL) {
			valid_input(SHELL, shell);
			usrdefs->defshell = shell;
		}
		if (skel_dir != NULL) {
			valid_input(SKELDIR, skel_dir);
			usrdefs->defskel = skel_dir;
		}
		update_def(usrdefs);

		/* change defaults for useradd */
		if (putusrdef(usrdefs, usertype) < 0) {
			errmsg(M_UPDATE, "created");
			exit(EX_UPDATE);
		}

		/* Now, display */
		dispusrdef(stdout, (D_ALL & ~D_RID), usertype);
		exit(EX_SUCCESS);

	}

	/* ADD mode */

	/* check syntax */
	if (optind != argc - 1 || (skel_dir != NULL && !mflag)) {
		if (is_role(usertype))
			errmsg(M_ARUSAGE);
		else
			errmsg(M_AUSAGE);
		exit(EX_SYNTAX);
	}

	logname = argv[optind];
	switch (valid_login(logname, (struct passwd **)NULL, &warning)) {
	case INVALID:
		errmsg(M_INVALID, logname, "login name");
```

```
                exit(EX_BADARG);
                /*NOTREACHED*/

        case NOTUNIQUE:
                errmsg(M_USED, logname);
                exit(EX_NAME_EXISTS);
                /*NOTREACHED*/

        case LONGNAME:
                errmsg(M_TOO_LONG, logname);
                exit(EX_BADARG);
                /*NOTREACHED*/
        }

        if (warning)
                warningmsg(warning, logname);
        if (uidstr != NULL) {
                /* convert uidstr to integer */
                errno = 0;
                uid = (uid_t)strtol(uidstr, &ptr, (int)10);
                if (*ptr || errno == ERANGE) {
                        errmsg(M_INVALID, uidstr, "user id");
                        exit(EX_BADARG);
                }

                switch (valid_uid(uid, NULL)) {
                case NOTUNIQUE:
                        if (!oflag) {
                                /* override not specified */
                                errmsg(M_UID_USED, uid);
                                exit(EX_ID_EXISTS);
                        }
                        break;
                case RESERVED:
                        errmsg(M_RESERVED, uid);
                        break;
                case TOOBIG:
                        errmsg(M_TOOBIG, "uid", uid);
                        exit(EX_BADARG);
                        break;
                }

        } else {

                if ((uid = findnextuid()) < 0) {
                        errmsg(M_INVALID, "default id", "user id");
                        exit(EX_ID_EXISTS);
                }
        }

        if (group != NULL) {
                switch (valid_group(group, &g_ptr, &warning)) {
                case INVALID:
                        errmsg(M_INVALID, group, "group id");
                        exit(EX_BADARG);
                        /*NOTREACHED*/
                case TOOBIG:
                        errmsg(M_TOOBIG, "gid", group);
                        exit(EX_BADARG);
                        /*NOTREACHED*/
                case RESERVED:
                case UNIQUE:
                        errmsg(M_GRP_NOTUSED, group);
                        exit(EX_NAME_NOT_EXIST);
                        /*NOTREACHED*/
                }
```

```

                if (warning)
                        warningmsg(warning, group);
                gid = g_ptr->gr_gid;

        } else gid = usrdefs->defgroup;

        if (grps != NULL) {
                if (!*grps)
                        /* ignore -G "" */
                        grps = (char *)0;
                else if (!(gidlist = valid_lgroup(grps, gid)))
                        exit(EX_BADARG);
        }

        if (projects != NULL) {
                if (! *projects)
                        projects = (char *)0;
                else if (! (projlist = valid_lproject(projects)))
                        exit(EX_BADARG);
        }

        /* if base_dir is provided, check its validity; otherwise default */
        if (base_dir != NULL)
                valid_input(BASEDIR, base_dir);
        else
                base_dir = usrdefs->defparent;

        if (dir == NULL) {
                /* set homedir to home directory made from base_dir */
                (void) sprintf(homedir, "%s/%s", base_dir, logname);

        } else if (REL_PATH(dir)) {
                errmsg(M_RELPATH, dir);
                exit(EX_BADARG);

        } else
                (void) strcpy(homedir, dir);

        if (mflag) {
                /* Does home dir. already exist? */
                if (stat(homedir, &statbuf) == 0) {
                        /* directory exists - don't try to create */
                        mflag = 0;

                        if (check_perm(statbuf, uid, gid, S_IXOTH) != 0)
                                errmsg(M_NO_PERM, logname, homedir);
                }
        }
        /*
         * if shell, skel_dir are provided, check their validity.
         * Otherwise default.
         */
        if (shell != NULL)
                valid_input(SHELL, shell);
        else
                shell = usrdefs->defshell;

        if (skel_dir != NULL)
                valid_input(SKELDIR, skel_dir);
        else
                skel_dir = usrdefs->defskel;

        if (inactstr != NULL) {
                /* convert inactstr to integer */
                inact = strtol(inactstr, &ptr, 10);
```

```
		if (*ptr || inact < 0) {
			errmsg(M_INVALID, inactstr, "inactivity period");
			exit(EX_BADARG);
		}
	} else inact = usrdefs->definact;

	/* expiration string is a date, newer than today */
	if (expirestr != NULL) {
		if (*expirestr) {
			if (valid_expire(expirestr, (time_t *)0) == INVALID) {
				errmsg(M_INVALID, expirestr, "expiration date");
				exit(EX_BADARG);
			}
			usrdefs->defexpire = expirestr;
		} else
			/* Unset the expiration date */
			expirestr = (char *)0;

	} else expirestr = usrdefs->defexpire;

	import_def(usrdefs);

	/* must now call passmgmt */

	/* set up arguments to  passmgmt in nargv array */
	nargv = malloc((30 + nkeys * 2) * sizeof (char *));
	argindex = 0;
	nargv[argindex++] = PASSMGMT;
	nargv[argindex++] = "-a";	/* add */

	if (comment != NULL) {
		/* comment */
		nargv[argindex++] = "-c";
		nargv[argindex++] = comment;
	}

	/* flags for home directory */
	nargv[argindex++] = "-h";
	nargv[argindex++] = homedir;

	/* set gid flag */
	nargv[argindex++] = "-g";
	(void) sprintf(gidstring, "%u", gid);
	nargv[argindex++] = gidstring;

	/* shell */
	nargv[argindex++] = "-s";
	nargv[argindex++] = shell;

	/* set inactive */
	nargv[argindex++] = "-f";
	(void) sprintf(inactstring, "%ld", inact);
	nargv[argindex++] = inactstring;

	/* set expiration date */
	if (expirestr != NULL) {
		nargv[argindex++] = "-e";
		nargv[argindex++] = expirestr;
	}

	/* set uid flag */
	nargv[argindex++] = "-u";
	(void) sprintf(uidstring, "%u", uid);
	nargv[argindex++] = uidstring;

	if (oflag) nargv[argindex++] = "-o";
```

```

	if (nkeys > 1)
		addkey_args(nargv, &argindex);

	/* finally - login name */
	nargv[argindex++] = logname;

	/* set the last to null */
	nargv[argindex++] = NULL;

	/* now call passmgmt */
	ret = PEX_FAILED;
	/*
	 * If call_passmgmt fails for any reason other than PEX_BADUID, exit
	 * is invoked with an appropriate error message. If PEX_BADUID is
	 * returned, then if the user specified the ID, exit is invoked
	 * with an appropriate error message. Otherwise we try to pick a
	 * different ID and try again. If we run out of IDs, i.e. no more
	 * users can be created, then -1 is returned and we terminate via exit.
	 * If PEX_BUSY is returned we increment a count, since we will stop
	 * trying if PEX_BUSY reaches 3. For PEX_SUCCESS we immediately
	 * terminate the loop.
	 */
	while (busy < 3 && ret != PEX_SUCCESS) {
		switch (ret = call_passmgmt(nargv)) {
		case PEX_SUCCESS:
			break;
		case PEX_BUSY:
			busy++;
			break;
		case PEX_HOSED_FILES:
			errmsg(M_HOSED_FILES);
			exit(EX_INCONSISTENT);
			break;

		case PEX_SYNTAX:
		case PEX_BADARG:
			/* should NEVER occur that passmgmt usage is wrong */
			if (is_role(usertype))
				errmsg(M_ARUSAGE);
			else
				errmsg(M_AUSAGE);
			exit(EX_SYNTAX);
			break;

		case PEX_BADUID:
			/*
			 * The uid has been taken. If it was specified by a
			 * user, then we must fail. Otherwise, keep trying
			 * to get a good uid until we run out of IDs.
			 */
			if (uidstr != NULL) {
				errmsg(M_UID_USED, uid);
				exit(EX_ID_EXISTS);
			} else {
				if ((uid = findnextuid()) < 0) {
					errmsg(M_INVALID, "default id",
					    "user id");
					exit(EX_ID_EXISTS);
				}
				(void) sprintf(uidstring, "%u", uid);
			}
			break;

		case PEX_BADNAME:
			/* invalid loname */
```

```
                        errmsg(M_USED, logname);
                        exit(EX_NAME_EXISTS);
                        break;

                default:
                        errmsg(M_UPDATE, "created");
                        exit(ret);
                        break;
                }
        }
        if (busy == 3) {
                errmsg(M_UPDATE, "created");
                exit(ret);
        }

        /* add group entry */
        if ((grps != NULL) && edit_group(logname, (char *)0, gidlist, 0)) {
                errmsg(M_UPDATE, "created");
                cleanup(logname);
                exit(EX_UPDATE);
        }

        /* update project database */
        if ((projects != NULL) &&
            edit_project(logname, (char *)NULL, projlist, 0)) {
                errmsg(M_UPDATE, "created");
                cleanup(logname);
                exit(EX_UPDATE);
        }

        /* create home directory */
        if (mflag &&
            (create_home(homedir, skel_dir, uid, gid) != EX_SUCCESS)) {
                (void) edit_group(logname, (char *)0, (int **)0, 1);
                cleanup(logname);
                exit(EX_HOMEDIR);
        }

        return (ret);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   15671 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/oamuser/user/usermod.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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



#include <sys/types.h>
#include <sys/stat.h>
#include <sys/param.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <limits.h>
#include <string.h>
#include <userdefs.h>
#include <user_attr.h>
#include <nss_dbdefs.h>
#include <errno.h>
#include <project.h>
#include "users.h"
#include "messages.h"
#include "funcs.h"

/*
 *  usermod [-u uid [-o] | -g group | -G group [[,group]...] | -d dir [-m]
 *              | -s shell | -c comment | -l new_logname]
 *              | -f inactive | -e expire ]
 *              [ -A authorization [, authorization ...]]
 *              [ -P profile [, profile ...]]
 *              [ -R role [, role ...]]
 *              [ -K key=value ]
 *              [ -p project [, project]] login
 *
 *      This command adds new user logins to the system.  Arguments are:
 *
 *      uid - an integer less than MAXUID
 *      group - an existing group's integer ID or char string name
 *      dir - a directory
 *      shell - a program to be used as a shell
 *      comment - any text string
 *      skel_dir - a directory
 *      base_dir - a directory
 *      rid - an integer less than 2**16 (USHORT)
 *      login - a string of printable chars except colon (:)
 *      inactive - number of days a login maybe inactive before it is locked
 *      expire - date when a login is no longer valid
 *      authorization - One or more comma separated authorizations defined
 *                      in auth_attr(4).
 *      profile - One or more comma separated execution profiles defined
 *              in prof_attr(4)
 *      role - One or more comma-separated role names defined in user_attr(4)
 *      key=value - One or more -K options each specifying a valid user_attr(4)
 *              attribute.
 *
 */

extern int **valid_lgroup(), isbusy();
extern int valid_uid(), check_perm(), create_home(), move_dir();
extern int valid_expire(), edit_group(), call_passmgmt();
extern projid_t **valid_lproject();

static uid_t uid;               /* new uid */
static gid_t gid;                       /* gid of new login */
static char *new_logname = NULL;        /* new login name with -l option */
static char *uidstr = NULL;             /* uid from command line */
static char *group = NULL;              /* group from command line */
static char *grps = NULL;               /* multi groups from command line */
static char *dir = NULL;                /* home dir from command line */
static char *shell = NULL;              /* shell from command line */
static char *comment = NULL;            /* comment from command line */
static char *logname = NULL;            /* login name to add */
static char *inactstr = NULL;           /* inactive from command line */
static char *expire = NULL;             /* expiration date from command line */
static char *projects = NULL;           /* project ids from command line */
static char *usertype;

char *cmdname;
static char gidstring[32], uidstring[32];
char inactstring[10];

char *
strcpmalloc(str)
char *str;
{
        if (str == NULL)
                return (NULL);

        return (strdup(str));
}
______unchanged_portion_omitted_

int
main(argc, argv)
int argc;
char **argv;
{
        int ch, ret = EX_SUCCESS, call_pass = 0, oflag = 0;
        int tries, mflag = 0, inact, **gidlist, flag = 0;
        boolean_t fail_if_busy = B_FALSE;
        char *ptr;
	struct passwd *pstruct;		/* password struct for login */
	struct passwd *pw;
	struct group *g_ptr;	/* validated group from -g */
	struct stat statbuf;		/* status buffer for stat */
#ifndef att
	FILE *pwf;		/* fille ptr for opened passwd file */
#endif
	int warning;
	projid_t **projlist;
	char **nargv;			/* arguments for execvp of passmgmt */
	int argindex;			/* argument index into nargv */
	userattr_t *ua;
	char *val;
	int isrole;			/* current account is role */

	cmdname = argv[0];

	if (geteuid() != 0) {
		errmsg(M_PERM_DENIED);
		exit(EX_NO_PERM);
	}

	opterr = 0;			/* no print errors from getopt */
	/* get user type based on the program name */
	usertype = getusertype(argv[0]);

	while ((ch = getopt(argc, argv,
				"c:d:e:f:G:g:l:mop:s:u:A:P:R:K:")) != EOF)
		switch (ch) {
		case 'c':
			comment = optarg;
			flag++;
			break;
		case 'd':
			dir = optarg;
			fail_if_busy = B_TRUE;
			flag++;
			break;
		case 'e':
			expire = optarg;
			flag++;
			break;
		case 'f':
			inactstr = optarg;
			flag++;
			break;
		case 'G':
			grps = optarg;
			flag++;
			break;
		case 'g':
			group = optarg;
			fail_if_busy = B_TRUE;
			flag++;
			break;
		case 'l':
			new_logname = optarg;
			fail_if_busy = B_TRUE;
			flag++;
			break;
		case 'm':
			mflag++;
			flag++;
			fail_if_busy = B_TRUE;
			break;
		case 'o':
```

```
			oflag++;
			flag++;
			fail_if_busy = B_TRUE;
			break;
		case 'p':
			projects = optarg;
			flag++;
			break;
		case 's':
			shell = optarg;
			flag++;
			break;
		case 'u':
			uidstr = optarg;
			flag++;
			fail_if_busy = B_TRUE;
			break;
		case 'A':
			change_key(USERATTR_AUTHS_KW, optarg);
			flag++;
			break;
		case 'P':
			change_key(USERATTR_PROFILES_KW, optarg);
			flag++;
			break;
		case 'R':
			change_key(USERATTR_ROLES_KW, optarg);
			flag++;
			break;
		case 'K':
			change_key(NULL, optarg);
			flag++;
			break;
		default:
		case '?':
			if (is_role(usertype))
				errmsg(M_MRUSAGE);
			else
				errmsg(M_MUSAGE);
			exit(EX_SYNTAX);
		}

	if (optind != argc - 1 || flag == 0) {
		if (is_role(usertype))
			errmsg(M_MRUSAGE);
		else
			errmsg(M_MUSAGE);
		exit(EX_SYNTAX);
	}

	if ((!uidstr && oflag) || (mflag && !dir)) {
		if (is_role(usertype))
			errmsg(M_MRUSAGE);
		else
			errmsg(M_MUSAGE);
		exit(EX_SYNTAX);
	}

	logname = argv[optind];

	/* Determine whether the account is a role or not */
	if ((ua = getusernam(logname)) == NULL ||
	    (val = kva_match(ua->attr, USERATTR_TYPE_KW)) == NULL ||
	    strcmp(val, USERATTR_TYPE_NONADMIN_KW) != 0)
		isrole = 0;
	else
```

```
			isrole = 1;

	/* Verify that rolemod is used for roles and usermod for users */
	if (isrole != is_role(usertype)) {
		if (isrole)
			errmsg(M_ISROLE);
		else
			errmsg(M_ISUSER);
		exit(EX_SYNTAX);
	}

	/* Set the usertype key; defaults to the commandline  */
	usertype = getsetdefval(USERATTR_TYPE_KW, usertype);

	if (is_role(usertype)) {
		/* Roles can't have roles */
		if (getsetdefval(USERATTR_ROLES_KW, NULL) != NULL) {
			errmsg(M_MRUSAGE);
			exit(EX_SYNTAX);
		}
		/* If it was an ordinary user, delete its roles */
		if (!isrole)
			change_key(USERATTR_ROLES_KW, "");
	}

#ifdef att
	pw = getpwnam(logname);
#else
	/*
	 * Do this with fgetpwent to make sure we are only looking on local
	 * system (since passmgmt only works on local system).
	 */
	if ((pwf = fopen("/etc/passwd", "r")) == NULL) {
		errmsg(M_OOPS, "open", "/etc/passwd");
		exit(EX_FAILURE);
	}
	while ((pw = fgetpwent(pwf)) != NULL)
		if (strcmp(pw->pw_name, logname) == 0)
			break;

	fclose(pwf);
#endif

	if (pw == NULL) {
		char		pwdb[NSS_BUFLEN_PASSWD];
		struct passwd	pwd;

		if (getpwnam_r(logname, &pwd, pwdb, sizeof (pwdb)) == NULL) {
			/* This user does not exist. */
			errmsg(M_EXIST, logname);
			exit(EX_NAME_NOT_EXIST);
		} else {
			/* This user exists in non-local name service. */
			errmsg(M_NONLOCAL, logname);
			exit(EX_NOT_LOCAL);
		}
	}

	pstruct = passwd_cpmalloc(pw);

	/*
	 * We can't modify a logged in user if any of the following
	 * are being changed:
	 * uid (-u & -o), group (-g), home dir (-m), loginname (-l).
	 * If none of those are specified it is okay to go ahead
	 * some types of changes only take effect on next login, some
```

```
	 * like authorisations and profiles take effect instantly.
	 * One might think that -K type=role should require that the
	 * user not be logged in, however this would make it very
	 * difficult to make the root account a role using this command.
	 */
	if (isbusy(logname)) {
		if (fail_if_busy) {
			errmsg(M_BUSY, logname, "change");
			exit(EX_BUSY);
		}
		warningmsg(WARN_LOGGED_IN, logname);
	}

	if (new_logname && strcmp(new_logname, logname)) {
		switch (valid_login(new_logname, (struct passwd **)NULL,
			&warning)) {
		case INVALID:
			errmsg(M_INVALID, new_logname, "login name");
			exit(EX_BADARG);
			/*NOTREACHED*/

		case NOTUNIQUE:
			errmsg(M_USED, new_logname);
			exit(EX_NAME_EXISTS);
			/*NOTREACHED*/

		case LONGNAME:
			errmsg(M_TOO_LONG, new_logname);
			exit(EX_BADARG);
			/*NOTREACHED*/

		default:
			call_pass = 1;
			break;
		}
		if (warning)
			warningmsg(warning, logname);
	}

	if (uidstr) {
		/* convert uidstr to integer */
		errno = 0;
		uid = (uid_t)strtol(uidstr, &ptr, (int)10);
		if (*ptr || errno == ERANGE) {
			errmsg(M_INVALID, uidstr, "user id");
			exit(EX_BADARG);
		}

		if (uid != pstruct->pw_uid) {
			switch (valid_uid(uid, NULL)) {
			case NOTUNIQUE:
				if (!oflag) {
					/* override not specified */
					errmsg(M_UID_USED, uid);
					exit(EX_ID_EXISTS);
				}
				break;
			case RESERVED:
				errmsg(M_RESERVED, uid);
				break;
			case TOOBIG:
				errmsg(M_TOOBIG, "uid", uid);
				exit(EX_BADARG);
				break;
			}
```

```
                        call_pass = 1;

                } else {
                        /* uid's the same, so don't change anything */
                        uidstr = NULL;
                        oflag = 0;
                }

        } else uid = pstruct->pw_uid;

        if (group) {
                switch (valid_group(group, &g_ptr, &warning)) {
                case INVALID:
                        errmsg(M_INVALID, group, "group id");
                        exit(EX_BADARG);
                        /*NOTREACHED*/
                case TOOBIG:
                        errmsg(M_TOOBIG, "gid", group);
                        exit(EX_BADARG);
                        /*NOTREACHED*/
                case UNIQUE:
                        errmsg(M_GRP_NOTUSED, group);
                        exit(EX_NAME_NOT_EXIST);
                        /*NOTREACHED*/
                case RESERVED:
                        gid = (gid_t)strtol(group, &ptr, (int)10);
                        errmsg(M_RESERVED_GID, gid);
                        break;
                }
                if (warning)
                        warningmsg(warning, group);

                if (g_ptr != NULL)
                        gid = g_ptr->gr_gid;
                else
                        gid = pstruct->pw_gid;

                /* call passmgmt if gid is different, else ignore group */
                if (gid != pstruct->pw_gid)
                        call_pass = 1;
                else group = NULL;

        } else gid = pstruct->pw_gid;

        if (grps && *grps) {
                if (!(gidlist = valid_lgroup(grps, gid)))
                        exit(EX_BADARG);
        } else
                gidlist = (int **)0;

        if (projects && *projects) {
                if (! (projlist = valid_lproject(projects)))
                        exit(EX_BADARG);
        } else
                projlist = (projid_t **)0;

        if (dir) {
                if (REL_PATH(dir)) {
                        errmsg(M_RELPATH, dir);
                        exit(EX_BADARG);
                }
                if (strcmp(pstruct->pw_dir, dir) == 0) {
                        /* home directory is the same so ignore dflag & mflag */
                        dir = NULL;
                        mflag = 0;
                } else call_pass = 1;
```

```
        }

        if (mflag) {
                if (stat(dir, &statbuf) == 0) {
                        /* Home directory exists */
                        if (check_perm(statbuf, pstruct->pw_uid,
                            pstruct->pw_gid, S_IWOTH|S_IXOTH) != 0) {
                                errmsg(M_NO_PERM, logname, dir);
                                exit(EX_NO_PERM);
                        }

                } else ret = create_home(dir, NULL, uid, gid);

                if (ret == EX_SUCCESS)
                        ret = move_dir(pstruct->pw_dir, dir, logname);

                if (ret != EX_SUCCESS)
                        exit(ret);
        }

        if (shell) {
                if (REL_PATH(shell)) {
                        errmsg(M_RELPATH, shell);
                        exit(EX_BADARG);
                }
                if (strcmp(pstruct->pw_shell, shell) == 0) {
                        /* ignore s option if shell is not different */
                        shell = NULL;
                } else {
                        if (stat(shell, &statbuf) < 0 ||
                            (statbuf.st_mode & S_IFMT) != S_IFREG ||
                            (statbuf.st_mode & 0555) != 0555) {

                                errmsg(M_INVALID, shell, "shell");
                                exit(EX_BADARG);
                        }

                        call_pass = 1;
                }
        }

        if (comment)
                /* ignore comment if comment is not changed */
                if (strcmp(pstruct->pw_comment, comment))
                        call_pass = 1;
                else
                        comment = NULL;

        /* inactive string is a positive integer */
        if (inactstr) {
                /* convert inactstr to integer */
                inact = (int)strtol(inactstr, &ptr, 10);
                if (*ptr || inact < 0) {
                        errmsg(M_INVALID, inactstr, "inactivity period");
                        exit(EX_BADARG);
                }
                call_pass = 1;
        }

        /* expiration string is a date, newer than today */
        if (expire) {
                if (*expire &&
                    valid_expire(expire, (time_t *)0) == INVALID) {
                        errmsg(M_INVALID, expire, "expiration date");
                        exit(EX_BADARG);
                }
                call_pass = 1;
        }

        if (nkeys > 0)
                call_pass = 1;

        /* that's it for validations - now do the work */

        if (grps) {
                /* redefine login's supplentary group memberships */
                ret = edit_group(logname, new_logname, gidlist, 1);
                if (ret != EX_SUCCESS) {
                        errmsg(M_UPDATE, "modified");
                        exit(ret);
                }
        }
        if (projects) {
                ret = edit_project(logname, (char *)NULL, projlist, 0);
                if (ret != EX_SUCCESS) {
                        errmsg(M_UPDATE, "modified");
                        exit(ret);
                }
        }


        if (!call_pass) exit(ret);

        /* only get to here if need to call passmgmt */
        /* set up arguments to  passmgmt in nargv array */
        nargv = malloc((30 + nkeys * 2) * sizeof (char *));

        argindex = 0;
        nargv[argindex++] = PASSMGMT;
        nargv[argindex++] = "-m";       /* modify */

        if (comment) {  /* comment */
                nargv[argindex++] = "-c";
                nargv[argindex++] = comment;
        }

        if (dir) {
                /* flags for home directory */
                nargv[argindex++] = "-h";
                nargv[argindex++] = dir;
        }

        if (group) {
                /* set gid flag */
                nargv[argindex++] = "-g";
                (void) sprintf(gidstring, "%u", gid);
                nargv[argindex++] = gidstring;
        }

        if (shell) {    /* shell */
                nargv[argindex++] = "-s";
                nargv[argindex++] = shell;
        }

        if (inactstr) {
                nargv[argindex++] = "-f";
                nargv[argindex++] = inactstr;
        }

        if (expire) {
                nargv[argindex++] = "-e";
                nargv[argindex++] = expire;
```

```
        }

        if (uidstr) {   /* set uid flag */
                nargv[argindex++] = "-u";
                (void) sprintf(uidstring, "%u", uid);
                nargv[argindex++] = uidstring;
        }

        if (oflag) nargv[argindex++] = "-o";

        if (new_logname) {      /* redefine login name */
                nargv[argindex++] = "-l";
                nargv[argindex++] = new_logname;
        }

        if (nkeys > 0)
                addkey_args(nargv, &argindex);

        /* finally - login name */
        nargv[argindex++] = logname;

        /* set the last to null */
        nargv[argindex++] = NULL;

        /* now call passmgmt */
        ret = PEX_FAILED;
        for (tries = 3; ret != PEX_SUCCESS && tries--; ) {
                switch (ret = call_passmgmt(nargv)) {
                case PEX_SUCCESS:
                case PEX_BUSY:
                        break;

                case PEX_HOSED_FILES:
                        errmsg(M_HOSED_FILES);
                        exit(EX_INCONSISTENT);
                        break;

                case PEX_SYNTAX:
                case PEX_BADARG:
                        /* should NEVER occur that passmgmt usage is wrong */
                        if (is_role(usertype))
                                errmsg(M_MRUSAGE);
                        else
                                errmsg(M_MUSAGE);
                        exit(EX_SYNTAX);
                        break;

                case PEX_BADUID:
                        /* uid in use - shouldn't happen print message anyway */
                        errmsg(M_UID_USED, uid);
                        exit(EX_ID_EXISTS);
                        break;

                case PEX_BADNAME:
                        /* invalid loname */
                        errmsg(M_USED, logname);
                        exit(EX_NAME_EXISTS);
                        break;

                default:
                        errmsg(M_UPDATE, "modified");
                        exit(ret);
                        break;
                }
        }
        if (tries == 0) {
                errmsg(M_UPDATE, "modified");
        }

        exit(ret);
        /*NOTREACHED*/
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   44863 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/prstat/prstat.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
```

```c
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Portions Copyright 2009 Chad Mynhier
 */

#include <sys/types.h>
#include <sys/resource.h>
#include <sys/loadavg.h>
#include <sys/time.h>
#include <sys/pset.h>
#include <sys/vm_usage.h>
#include <zone.h>
#include <libzonecfg.h>

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <string.h>
#include <errno.h>
#include <poll.h>
#include <ctype.h>
#include <fcntl.h>
#include <limits.h>
#include <signal.h>
#include <time.h>
#include <project.h>

#include <langinfo.h>
#include <libintl.h>
#include <locale.h>

#include "prstat.h"
#include "prutil.h"
#include "prtable.h"
```

```c
#include "prsort.h"
#include "prfile.h"

/*
 * x86 <sys/regs.h> ERR conflicts with <curses.h> ERR.  For the purposes
 * of this file, we care about the curses.h ERR so include that last.
 */

#if     defined(ERR)
#undef  ERR
#endif

#ifndef TEXT_DOMAIN                     /* should be defined by cc -D */
#define TEXT_DOMAIN     "SYS_TEST"      /* use this only if it wasn't */
#endif

#include <curses.h>
#include <term.h>

#define LOGIN_WIDTH     8
#define ZONE_WIDTH      28
#define PROJECT_WIDTH   28

#define PSINFO_HEADER_PROC \
"   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP       "
#define PSINFO_HEADER_PROC_LGRP \
"   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU LGRP PROCESS/NLWP  "
#define PSINFO_HEADER_LWP \
"   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID      "
#define PSINFO_HEADER_LWP_LGRP \
"   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU LGRP PROCESS/LWPID "
#define USAGE_HEADER_PROC \
"   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/NLWP  "
#define USAGE_HEADER_LWP \
"   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID "
#define USER_HEADER_PROC \
" NPROC USERNAME  SWAP   RSS MEMORY      TIME  CPU                             "
#define USER_HEADER_LWP \
"  NLWP USERNAME  SWAP   RSS MEMORY      TIME  CPU                             "
#define TASK_HEADER_PROC \
"TASKID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT                     "
#define TASK_HEADER_LWP \
"TASKID     NLWP  SWAP   RSS MEMORY      TIME  CPU PROJECT                     "
#define PROJECT_HEADER_PROC \
"PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT                     "
#define PROJECT_HEADER_LWP \
"PROJID     NLWP  SWAP   RSS MEMORY      TIME  CPU PROJECT                     "
#define ZONE_HEADER_PROC \
"ZONEID    NPROC  SWAP   RSS MEMORY      TIME  CPU ZONE                        "
#define ZONE_HEADER_LWP \
"ZONEID     NLWP  SWAP   RSS MEMORY      TIME  CPU ZONE                        "
#define PSINFO_LINE \
"%6d %-8s %5s %5s %-6s %3s  %3s %9s %3.3s%% %-.16s/%d"
#define PSINFO_LINE_LGRP \
"%6d %-8s %5s %5s %-6s %3s  %3s %9s %3.3s%% %4d %-.16s/%d"
#define USAGE_LINE \
"%6d %-8s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s "\
"%3.3s %3.3s %-.12s/%d"
"%6d %-8s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s %3.3s "\
"%3.3s %-.12s/%d"
#define USER_LINE \
"%6d %-8s %5.5s %5.5s   %3.3s%% %9s %3.3s%%"
#define TASK_LINE \
"%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"
#define PROJECT_LINE \
"%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"
```

```
#define ZONE_LINE \
"%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"

#define TOTAL_LINE \
"Total: %d processes, %d lwps, load averages: %3.2f, %3.2f, %3.2f"

/* global variables */

static char     *t_ulon;                        /* termcap: start underline */
static char     *t_uloff;                       /* termcap: end underline */
static char     *t_up;                          /* termcap: cursor 1 line up */
static char     *t_eol;                         /* termcap: clear end of line */
static char     *t_smcup;                       /* termcap: cursor mvcap on */
static char     *t_rmcup;                       /* termcap: cursor mvcap off */
static char     *t_home;                        /* termcap: move cursor home */
static char     *movecur = NULL;                /* termcap: move up string */
static char     *empty_string = "\0";           /* termcap: empty string */
static uint_t   print_movecur = FALSE;          /* print movecur or not */
static int      is_curses_on = FALSE;           /* current curses state */

static table_t  pid_tbl = {0, 0, NULL};         /* selected processes */
static table_t  cpu_tbl = {0, 0, NULL};         /* selected processors */
static table_t  set_tbl = {0, 0, NULL};         /* selected processor sets */
static table_t  prj_tbl = {0, 0, NULL};         /* selected projects */
static table_t  tsk_tbl = {0, 0, NULL};         /* selected tasks */
static table_t  lgr_tbl = {0, 0, NULL};         /* selected lgroups */
static zonetbl_t zone_tbl = {0, 0, NULL};       /* selected zones */
static uidtbl_t euid_tbl = {0, 0, NULL};        /* selected effective users */
static uidtbl_t ruid_tbl = {0, 0, NULL};        /* selected real users */

static uint_t   total_procs;                    /* total number of procs */
static uint_t   total_lwps;                     /* total number of lwps */
static float    total_cpu;                      /* total cpu usage */
static float    total_mem;                      /* total memory usage */

static list_t   lwps;                           /* list of lwps/processes */
static list_t   users;                          /* list of users */
static list_t   tasks;                          /* list of tasks */
static list_t   projects;                       /* list of projects */
static list_t   zones;                          /* list of zones */
static list_t   lgroups;                        /* list of lgroups */

static volatile uint_t sigwinch = 0;
static volatile uint_t sigtstp = 0;
static volatile uint_t sigterm = 0;

static long pagesize;

/* default settings */

static optdesc_t opts = {
        5,                      /* interval between updates, seconds */
        15,                     /* number of lines in top part */
        5,                      /* number of lines in bottom part */
        -1,                     /* number of iterations; infinitely */
        OPT_PSINFO | OPT_FULLSCREEN | OPT_USEHOME | OPT_TERMCAP,
        -1                      /* sort in decreasing order */
};
```
______***unchanged_portion_omitted_***

```
/*
 * A routine to display the contents of the list on the screen
 */
static void
list_print(list_t *list)
{
```

```
        lwp_info_t *lwp;
        id_info_t *id;
        char usr[4], sys[4], trp[4], tfl[4];
        char dfl[4], lck[4], slp[4], lat[4];
        char vcx[4], icx[4], scl[4], sig[4];
        char psize[6], prssize[6], pmem[6], pcpu[6], ptime[12];
        char pstate[7], pnice[4], ppri[4];
        char pname[LOGNAME_MAX_ILLUMOS+1];
        char pname[LOGNAME_MAX+1];
        char projname[PROJNAME_MAX+1];
        char zonename[ZONENAME_MAX+1];
        float cpu, mem;
        double loadavg[3] = {0, 0, 0};
        int i, lwpid;

        if (foreach_element(&set_tbl, &loadavg, psetloadavg) == 0) {
                /*
                 * If processor sets aren't specified, we display system-wide
                 * load averages.
                 */
                (void) getloadavg(loadavg, 3);
        }

        if (((opts.o_outpmode & OPT_UDATE) || (opts.o_outpmode & OPT_DDATE)) &&
            ((list->l_type == LT_LWPS) || !(opts.o_outpmode & OPT_SPLIT)))
                print_timestamp();
        if (opts.o_outpmode & OPT_TTY)
                (void) putchar('\r');
        (void) putp(t_ulon);

        switch (list->l_type) {
        case LT_PROJECTS:
                if (opts.o_outpmode & OPT_LWPS)
                        (void) printf(PROJECT_HEADER_LWP);
                else
                        (void) printf(PROJECT_HEADER_PROC);
                break;
        case LT_TASKS:
                if (opts.o_outpmode & OPT_LWPS)
                        (void) printf(TASK_HEADER_LWP);
                else
                        (void) printf(TASK_HEADER_PROC);
                break;
        case LT_ZONES:
                if (opts.o_outpmode & OPT_LWPS)
                        (void) printf(ZONE_HEADER_LWP);
                else
                        (void) printf(ZONE_HEADER_PROC);
                break;
        case LT_USERS:
                if (opts.o_outpmode & OPT_LWPS)
                        (void) printf(USER_HEADER_LWP);
                else
                        (void) printf(USER_HEADER_PROC);
                break;
        case LT_LWPS:
                if (opts.o_outpmode & OPT_LWPS) {
                        if (opts.o_outpmode & OPT_PSINFO) {
                                if (opts.o_outpmode & OPT_LGRP)
                                        (void) printf(PSINFO_HEADER_LWP_LGRP);
                                else
                                        (void) printf(PSINFO_HEADER_LWP);
                        }
                        if (opts.o_outpmode & OPT_MSACCT)
                                (void) printf(USAGE_HEADER_LWP);
                } else {
                        if (opts.o_outpmode & OPT_PSINFO) {
                                if (opts.o_outpmode & OPT_LGRP)
                                        (void) printf(PSINFO_HEADER_PROC_LGRP);
                                else
                                        (void) printf(PSINFO_HEADER_PROC);
                        }
                        if (opts.o_outpmode & OPT_MSACCT)
                                (void) printf(USAGE_HEADER_PROC);
                }
                break;
        }

        (void) putp(t_uloff);
        (void) putp(t_eol);
        (void) putchar('\n');

        for (i = 0; i < list->l_used; i++) {
                switch (list->l_type) {
                case LT_PROJECTS:
                case LT_TASKS:
                case LT_USERS:
                case LT_ZONES:
                        id = list->l_ptrs[i];
                        /*
                         * CPU usage and memory usage normalization
                         */
                        if (total_cpu >= 100)
                                cpu = (100 * id->id_pctcpu) / total_cpu;
                        else
                                cpu = id->id_pctcpu;
                        if (id->id_sizematch == B_FALSE && total_mem >= 100)
                                mem = (100 * id->id_pctmem) / total_mem;
                        else
                                mem = id->id_pctmem;
                        if (list->l_type == LT_USERS) {
                                pwd_getname(id->id_uid, pname, sizeof (pname),
                                    opts.o_outpmode & OPT_NORESOLVE,
                                    opts.o_outpmode & OPT_TERMCAP,
                                    LOGIN_WIDTH);
                        } else if (list->l_type == LT_ZONES) {
                        if (list->l_type == LT_USERS)
                                pwd_getname(id->id_uid, pname, LOGNAME_MAX + 1,
                                    opts.o_outpmode & OPT_NORESOLVE);
                        else if (list->l_type == LT_ZONES)
                                getzonename(id->id_zoneid, zonename,
                                    sizeof (zonename) - 1,
                                    opts.o_outpmode & OPT_TERMCAP,
                                    ZONE_WIDTH);
                        } else {
                                    ZONENAME_MAX);
                        else
                                getprojname(id->id_projid, projname,
                                    sizeof (projname) - 1,
                                    opts.o_outpmode & OPT_NORESOLVE,
                                    opts.o_outpmode & OPT_TERMCAP,
                                    PROJECT_WIDTH);
                        }
                                    PROJNAME_MAX,
                                    opts.o_outpmode & OPT_NORESOLVE);
                        Format_size(psize, id->id_size, 6);
                        Format_size(prssize, id->id_rssize, 6);
                        Format_pct(pmem, mem, 4);
                        Format_pct(pcpu, cpu, 4);
                        Format_time(ptime, id->id_time, 10);
                        if (opts.o_outpmode & OPT_TTY)
                                (void) putchar('\r');
```

```
                        if (list->l_type == LT_PROJECTS)
                                (void) printf(PROJECT_LINE, (int)id->id_projid,
                                    id->id_nproc, psize, prssize, pmem, ptime,
                                    pcpu, projname);
                        else if (list->l_type == LT_TASKS)
                                (void) printf(TASK_LINE, (int)id->id_taskid,
                                    id->id_nproc, psize, prssize, pmem, ptime,
                                    pcpu, projname);
                        else if (list->l_type == LT_ZONES)
                                (void) printf(ZONE_LINE, (int)id->id_zoneid,
                                    id->id_nproc, psize, prssize, pmem, ptime,
                                    pcpu, zonename);
                        else
                                (void) printf(USER_LINE, id->id_nproc, pname,
                                    psize, prssize, pmem, ptime, pcpu);
                        (void) putp(t_eol);
                        (void) putchar('\n');
                        break;
                case LT_LWPS:
                        lwp = list->l_ptrs[i];
                        if (opts.o_outpmode & OPT_LWPS)
                                lwpid = lwp->li_info.pr_lwp.pr_lwpid;
                        else
                                lwpid = lwp->li_info.pr_nlwp +
                                    lwp->li_info.pr_nzomb;
                        pwd_getname(lwp->li_info.pr_uid, pname, sizeof (pname),
                            opts.o_outpmode & OPT_NORESOLVE,
                            opts.o_outpmode & OPT_TERMCAP,
                            LOGIN_WIDTH);
                        pwd_getname(lwp->li_info.pr_uid, pname, LOGNAME_MAX + 1,
                            opts.o_outpmode & OPT_NORESOLVE);
                        if (opts.o_outpmode & OPT_PSINFO) {
                                Format_size(psize, lwp->li_info.pr_size, 6);
                                Format_size(prssize, lwp->li_info.pr_rssize, 6);
                                Format_state(pstate,
                                    lwp->li_info.pr_lwp.pr_sname,
                                    lwp->li_info.pr_lwp.pr_onpro, 7);
                                if (strcmp(lwp->li_info.pr_lwp.pr_clname,
                                    "RT") == 0 ||
                                    strcmp(lwp->li_info.pr_lwp.pr_clname,
                                    "SYS") == 0 ||
                                    lwp->li_info.pr_lwp.pr_sname == 'Z')
                                        (void) strcpy(pnice, "  -");
                                else
                                        Format_num(pnice,
                                            lwp->li_info.pr_lwp.pr_nice - NZERO,
                                            4);
                                Format_num(ppri, lwp->li_info.pr_lwp.pr_pri, 4);
                                Format_pct(pcpu,
                                    FRC2PCT(lwp->li_info.pr_lwp.pr_pctcpu), 4);
                                if (opts.o_outpmode & OPT_LWPS)
                                        Format_time(ptime,
                                            lwp->li_info.pr_lwp.pr_time.tv_sec,
                                            10);
                                else
                                        Format_time(ptime,
                                            lwp->li_info.pr_time.tv_sec, 10);
                                if (opts.o_outpmode & OPT_TTY)
                                        (void) putchar('\r');
                                stripfname(lwp->li_info.pr_fname);
                                if (opts.o_outpmode & OPT_LGRP) {
                                        (void) printf(PSINFO_LINE_LGRP,
                                            (int)lwp->li_info.pr_pid, pname,
                                            psize, prssize, pstate,
                                            ppri, pnice, ptime, pcpu,
                                            psize, prssize, pstate, ppri, pnice,
                                                    ptime, pcpu,
                                                    (int)lwp->li_info.pr_lwp.pr_lgrp,
                                                    lwp->li_info.pr_fname, lwpid);
                                    } else {
                                            (void) printf(PSINFO_LINE,
                                                    (int)lwp->li_info.pr_pid, pname,
                                                    psize, prssize,
                                                    pstate, ppri, pnice,
                                                    psize, prssize, pstate, ppri, pnice,
                                                    ptime, pcpu,
                                                    lwp->li_info.pr_fname, lwpid);
                                    }
                                    (void) putp(t_eol);
                                    (void) putchar('\n');
                            }
                            if (opts.o_outpmode & OPT_MSACCT) {
                                    Format_pct(usr, lwp->li_usr, 4);
                                    Format_pct(sys, lwp->li_sys, 4);
                                    Format_pct(slp, lwp->li_slp, 4);
                                    Format_num(vcx, lwp->li_vcx, 4);
                                    Format_num(icx, lwp->li_icx, 4);
                                    Format_num(scl, lwp->li_scl, 4);
                                    Format_num(sig, lwp->li_sig, 4);
                                    Format_pct(trp, lwp->li_trp, 4);
                                    Format_pct(tfl, lwp->li_tfl, 4);
                                    Format_pct(dfl, lwp->li_dfl, 4);
                                    Format_pct(lck, lwp->li_lck, 4);
                                    Format_pct(lat, lwp->li_lat, 4);
                                    if (opts.o_outpmode & OPT_TTY)
                                            (void) putchar('\r');
                                    stripfname(lwp->li_info.pr_fname);
                                    (void) printf(USAGE_LINE,
                                        (int)lwp->li_info.pr_pid, pname,
                                        usr, sys, trp, tfl, dfl, lck,
                                        slp, lat, vcx, icx, scl, sig,
                                        lwp->li_info.pr_fname, lwpid);
                                    (void) putp(t_eol);
                                    (void) putchar('\n');
                            }
                            break;
                    }
            }

            if (opts.o_outpmode & OPT_TTY)
                    (void) putchar('\r');
            if (opts.o_outpmode & OPT_TERMCAP) {
                    switch (list->l_type) {
                    case LT_PROJECTS:
                    case LT_USERS:
                    case LT_TASKS:
                    case LT_ZONES:
                            while (i++ < opts.o_nbottom) {
                                    (void) putp(t_eol);
                                    (void) putchar('\n');
                            }
                            break;
                    case LT_LWPS:
                            while (i++ < opts.o_ntop) {
                                    (void) putp(t_eol);
                                    (void) putchar('\n');
                            }
                    }
            }

            if (opts.o_outpmode & OPT_TTY)
                    (void) putchar('\r');
```

```
            if ((opts.o_outpmode & OPT_SPLIT) && list->l_type == LT_LWPS)
                    return;

            (void) printf(TOTAL_LINE, total_procs, total_lwps,
                loadavg[LOADAVG_1MIN], loadavg[LOADAVG_5MIN],
                loadavg[LOADAVG_15MIN]);
            (void) putp(t_eol);
            (void) putchar('\n');
            if (opts.o_outpmode & OPT_TTY)
                    (void) putchar('\r');
            (void) putp(t_eol);
            (void) fflush(stdout);
    }
```
______***unchanged_portion_omitted_***

****
    6709 Wed Apr 3 09:33:11 2013
new/usr/src/cmd/prstat/prtable.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
****

```
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Portions Copyright 2009 Chad Mynhier
 */

#include <procfs.h>
#include <unistd.h>
#include <stdlib.h>
#include <pwd.h>
#include <ctype.h>
#include <string.h>
#include <libintl.h>
#include <errno.h>
#include <zone.h>
#include <libzonecfg.h>

#include "prstat.h"
#include "prutil.h"
#include "prtable.h"

static plwp_t   *plwp_tbl[PLWP_TBL_SZ];

void
lwpid_init()
{
        (void) memset(&plwp_tbl, 0, sizeof (plwp_t *) * PLWP_TBL_SZ);
}
```

______*unchanged_portion_omitted_*

```
void
pwd_getname(uid_t uid, char *name, size_t length, int noresolve,
    int termcap, size_t width)
pwd_getname(uid_t uid, char *name, int length, int noresolve)
{
        struct passwd *pwd;
        size_t n;
```

```
        if (noresolve || (pwd = getpwuid(uid)) == NULL) {
                (void) snprintf(name, length, "%u", uid);
        } else {
                n = strlen(pwd->pw_name);
                if (termcap && n > width)
                        (void) snprintf(name, length, "%.*s%c",
                            width - 1, pwd->pw_name, '*');
                else
                        (void) snprintf(name, length, "%s", pwd->pw_name);
        }
}
```

______*unchanged_portion_omitted_*

```
**********************************************************
    2466 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/prstat/prtable.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Portions Copyright 2009 Chad Mynhier
 */

#ifndef _PRTABLE_H
#define _PRTABLE_H

#ifdef  __cplusplus
extern "C" {
#endif

#include <limits.h>
#include <zone.h>
#include "prstat.h"

#define PLWP_TBL_SZ     4096    /* hash table of plwp_t structures */
#define LWP_ACTIVE      1

typedef struct {
        size_t          t_size;
        size_t          t_nent;
        long            *t_list;
} table_t;
______unchanged_portion_omitted_

extern void pwd_getname(uid_t, char *, size_t, int, int, size_t);
extern void pwd_getname(uid_t, char *, int, int);
extern void add_uid(uidtbl_t *, char *);
extern int has_uid(uidtbl_t *, uid_t);
extern void add_element(table_t *, long);
extern int has_element(table_t *, long);
extern void add_zone(zonetbl_t *, char *);
extern int has_zone(zonetbl_t *, zoneid_t);
extern void convert_zone(zonetbl_t *);
extern int foreach_element(table_t *, void *, void (*)(long, void *));
```

```
extern void lwpid_init();
extern void lwpid_add(lwp_info_t *, pid_t, id_t);
extern lwp_info_t *lwpid_get(pid_t, id_t);
extern int lwpid_pidcheck(pid_t);
extern void lwpid_del(pid_t, id_t);
extern void lwpid_set_active(pid_t, id_t);
extern int lwpid_is_active(pid_t, id_t);

#ifdef  __cplusplus
}
______unchanged_portion_omitted_
```

```
**********************************************************
    7551 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/prstat/prutil.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Portions Copyright 2009 Chad Mynhier
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/resource.h>
#include <sys/priocntl.h>
#include <sys/rtpriocntl.h>
#include <sys/tspriocntl.h>
#include <zone.h>

#include <libintl.h>
#include <limits.h>
#include <wchar.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdio_ext.h>
#include <errno.h>
#include <ctype.h>
#include <poll.h>
#include <project.h>

#include "prfile.h"
#include "prstat.h"
#include "prutil.h"

static char PRG_FMT[] = "%s: ";
static char ERR_FMT[] = ": %s\n";
static char *progname;
static char projbuf[PROJECT_BUFSZ];
```

```
#define RLIMIT_NOFILE_MAX       32767

/*PRINTFLIKE1*/
void
Warn(char *format, ...)
{
        int err = errno;
        va_list alist;

        if (progname != NULL)
                (void) fprintf(stderr, PRG_FMT, progname);
        va_start(alist, format);
        (void) vfprintf(stderr, format, alist);
        va_end(alist);
        if (strchr(format, '\n') == NULL)
                (void) fprintf(stderr, gettext(ERR_FMT), strerror(err));
}
______unchanged_portion_omitted_

void
getprojname(projid_t projid, char *str, size_t len, int noresolve,
    int termcap, size_t width)
getprojname(projid_t projid, char *str, int len, int noresolve)
{
        struct project proj;
        size_t n;

        if (noresolve || getprojbyid(projid, &proj, projbuf, PROJECT_BUFSZ) ==
            NULL) {
            NULL)
                (void) snprintf(str, len, "%-6d", (int)projid);
        } else {
                n = strlen(proj.pj_name);
                if (termcap && n > width)
                        (void) snprintf(str, len, "%.*s%c", width - 1,
                            proj.pj_name, '*');
                else
                        (void) snprintf(str, len, "%-28s", proj.pj_name);
        }
}

void
getzonename(zoneid_t zoneid, char *str, size_t len, int termcap, size_t width)
getzonename(zoneid_t zoneid, char *str, int len)
{
        char zone_name[ZONENAME_MAX];
        size_t n;

        if (getzonenamebyid(zoneid, zone_name, sizeof (zone_name)) < 0) {
        if (getzonenamebyid(zoneid, zone_name, sizeof (zone_name)) < 0)
                (void) snprintf(str, len, "%-6d", (int)zoneid);
        } else {
                n = strlen(zone_name);
                if (termcap && n > width)
                        (void) snprintf(str, len, "%.*s%c", width - 1,
                            zone_name, '*');
                else
                        (void) snprintf(str, len, "%-28s", zone_name);
        }
}
______unchanged_portion_omitted_
```

```
**********************************************************
    1878 Wed Apr  3 09:33:11 2013
new/usr/src/cmd/prstat/prutil.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Portions Copyright 2009 Chad Mynhier
 */

#ifndef _PRUTIL_H
#define _PRUTIL_H

#include <sys/processor.h>
#include <sys/types.h>

#ifdef  __cplusplus
extern "C" {
#endif

extern void Die(char *, ...);
extern void Warn(char *, ...);
extern void Progname(char *);
extern void Usage();
extern int Atoi(char *);
extern void Format_size(char *, size_t, int);
extern void Format_pct(char *, float, int);
extern void Format_num(char *, int, int);
extern void Format_time(char *, ulong_t, int);
extern void Format_state(char *, char, processorid_t, int);
extern void *Realloc(void *, size_t);
extern void *Malloc(size_t);
extern void *Zalloc(size_t);
extern int Setrlimit();
extern void Priocntl(char *);
extern void getprojname(projid_t, char *, size_t, int, int, size_t);
extern void getzonename(projid_t, char *, size_t, int, size_t);
extern void getprojname(projid_t, char *, int, int);
extern void getzonename(projid_t, char *, int);
extern void stripfname(char *);
```

```
#ifdef  __cplusplus
}
______unchanged_portion_omitted_
```

****
   57960 Wed Apr 3 09:33:11 2013
new/usr/src/cmd/ps/ps.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
****

```
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2012, Joyent, Inc. All rights reserved.
 */

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

/*
 * ps -- print things about processes.
 */
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <pwd.h>
#include <grp.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mkdev.h>
#include <unistd.h>
#include <stdlib.h>
#include <limits.h>
#include <dirent.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/time.h>
#include <procfs.h>
#include <locale.h>
#include <wctype.h>
#include <wchar.h>
```

```
#include <libw.h>
#include <stdarg.h>
#include <sys/proc.h>
#include <sys/pset.h>
#include <project.h>
#include <zone.h>

#define min(a, b)       ((a) > (b) ? (b) : (a))
#define max(a, b)       ((a) < (b) ? (b) : (a))

#define NTTYS   20      /* initial size of table for -t option  */
#define SIZ     30      /* initial size of tables for -p, -s, -g, -h and -z */

/*
 * Size of buffer holding args for t, p, s, g, u, U, G, z options.
 * Set to ZONENAME_MAX, the minimum value needed to allow any
 * zone to be specified.
 */
#define ARGSIZ ZONENAME_MAX

#define MAXUGNAME (LOGNAME_MAX_ILLUMOS+2)       /* max chars in a user/group */
                                                /* name or printed u/g id */
#define MAXUGNAME 10    /* max chars in a user/group name or printed u/g id */

/* Structure for storing user or group info */
struct ugdata {
        id_t    id;                     /* numeric user-id or group-id */
        char    name[MAXUGNAME+1];      /* user/group name, null terminated */
};
```
______***unchanged_portion_omitted_***

```
**********************************************************
    5363 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/pwck/pwck.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2007 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"

#include <sys/types.h>
#include <sys/param.h>
#include <sys/signal.h>
#include <sys/sysmacros.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <locale.h>
#include <errno.h>
#include <unistd.h>
#include <limits.h>

#define ERROR1  "Too many/few fields"
#define ERROR2  "Bad character(s) in logname"
#define ERROR2a "First char in logname not alphabetic"
#define ERROR2b "Logname field NULL"
#define ERROR2c "Logname contains no lower-case letters"
#define ERROR3  "Logname too long/short"
#define ERROR4  "Invalid UID"
#define ERROR5  "Invalid GID"
#define ERROR6  "Login directory not found"
#define ERROR6a "Login directory null"
#define ERROR7  "Optional shell file not found"

static int eflag, code = 0;
```

```
static int badc;
static int lc;
static char buf[512];
static void error(char *);

int
main(int argc, char **argv)
{
        int delim[512];
        char logbuf[512];
        FILE *fptr;
        struct stat obuf;
        uid_t uid;
        gid_t gid;
        int i, j, colons;
        char *pw_file;
        struct stat stat_buf;
        char *str, *lastc;

        (void) setlocale(LC_ALL, "");

#if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"
#endif
        (void) textdomain(TEXT_DOMAIN);

        if (argc == 1)
                pw_file = "/etc/passwd";
        else
                pw_file = argv[1];

        if ((fptr = fopen(pw_file, "r")) == NULL) {
                (void) fprintf(stderr, gettext("cannot open %s\n"), pw_file);
                exit(1);
        }

        if (fstat(fileno(fptr), &stat_buf) < 0) {
                (void) fprintf(stderr, gettext("fstat failed for %s\n"),
                    pw_file);
                (void) fclose(fptr);
                exit(1);
        }

        if (stat_buf.st_size == 0) {
                (void) fprintf(stderr, gettext("file %s is empty\n"), pw_file);
                (void) fclose(fptr);
                exit(1);
        }

        while (fgets(buf, sizeof (buf), fptr) != NULL) {

                colons = 0;
                badc = 0;
                lc = 0;
                eflag = 0;

                /* Check that entry is not a nameservice redirection */

                if (buf[0] == '+' || buf[0] == '-')  {
                        /*
                         * Should set flag here to allow special case checking
                         * in the rest of the code,
                         * but for now, we'll just ignore this entry.
                         */
                        continue;
                }
                /* Check number of fields */

                for (i = 0; buf[i] != NULL; i++)
                        if (buf[i] == ':') {
                                delim[colons] = i;
                                ++colons;
                        }

                if (colons != 6) {
                        error(ERROR1);
                        continue;
                }
                delim[6] = i - 1;
                delim[7] = NULL;

                /*
                 * Check the first char is alpha; the rest alphanumeric;
                 * and that the name does not consist solely of uppercase
                 * alpha chars
                 */
                if (buf[0] == ':')
                        error(ERROR2b);
                else if (!isalpha(buf[0]))
                        error(ERROR2a);

                for (i = 0; buf[i] != ':'; i++) {
                        if (!isalnum(buf[i]) &&
                            buf[i] != '_' &&
                            buf[i] != '-' &&
                            buf[i] != '.')
                                badc++;
                        else if (islower(buf[i]))
                                lc++;
                }
                if (lc == 0)
                        error(ERROR2c);
                if (badc > 0)
                        error(ERROR2);

                /* Check for valid number of characters in logname */

                if (i <= 0 || i > LOGNAME_MAX_ILLUMOS)
                if (i <= 0 || i > 8)
                        error(ERROR3);

                /* Check that UID is numeric and <= MAXUID */

                errno = 0;
                str = &buf[delim[1] + 1];
                uid = strtol(str, &lastc, 10);
                if (lastc != str + (delim[2] - delim[1]) - 1 ||
                    uid > MAXUID || errno == ERANGE)
                        error(ERROR4);

                /* Check that GID is numeric and <= MAXUID */

                errno = 0;
                str = &buf[delim[2] + 1];
                gid = strtol(str, &lastc, 10);
                if (lastc != str + (delim[3] - delim[2]) - 1 ||
                    gid > MAXUID || errno == ERANGE)
                        error(ERROR5);

                /* Check initial working directory */
```

```
                for (j = 0, i = (delim[4] + 1); i < delim[5]; j++, i++)
                        logbuf[j] = buf[i];
                logbuf[j] = '\0';

                if (logbuf[0] == NULL)
                        error(ERROR6a);
                else if ((stat(logbuf, &obuf)) == -1)
                        error(ERROR6);

                /* Check program to use as shell  */

                if ((buf[(delim[5] + 1)]) != '\n') {

                        for (j = 0, i = (delim[5] + 1); i < delim[6]; j++, i++)
                                logbuf[j] = buf[i];
                        logbuf[j] = '\0';

                        if (strcmp(logbuf, "*") == 0)   /* subsystem login */
                                continue;

                        if ((stat(logbuf, &obuf)) == -1)
                                error(ERROR7);

                        for (j = 0; j < 512; j++)
                                logbuf[j] = NULL;
                }
        }
        (void) fclose(fptr);
        return (code);
}
```

______unchanged_portion_omitted_

```
**********************************************************
   18980 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/w/w.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * 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   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

/*
 * This is the new w command which takes advantage of
 * the /proc interface to gain access to the information
 * of all the processes currently on the system.
 *
 * This program also implements 'uptime'.
 *
 * Maintenance note:
 *
 * Much of this code is replicated in whodo.c.  If you're
 * fixing bugs here, then you should probably fix 'em there too.
 */

#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <stdlib.h>
#include <ctype.h>
#include <fcntl.h>
#include <time.h>
```

```
#include <errno.h>
#include <sys/types.h>
#include <utmpx.h>
#include <sys/stat.h>
#include <dirent.h>
#include <procfs.h>             /* /proc header file */
#include <locale.h>
#include <unistd.h>
#include <sys/loadavg.h>
#include <limits.h>
#include <priv_utils.h>

/*
 * Use the full lengths from utmpx for user and line.
 * utmpx defines wider fields for user and line.  For compatibility of output,
 * we are limiting these to the old maximums in utmp. Define UTMPX_NAMELEN
 * to use the full lengths.
 */
#ifndef UTMPX_NAMELEN
/* XXX - utmp - fix name length */
#define NMAX            (_POSIX_LOGIN_NAME_MAX - 1)
#define LMAX            12
#else   /* UTMPX_NAMELEN */
static struct utmpx dummy;
#define NMAX            (sizeof (dummy.ut_user))
#define LMAX            (sizeof (dummy.ut_line))
#endif /* UTMPX_NAMELEN */

/* Print minimum field widths. */
#define LOGIN_WIDTH     8
#define LINE_WIDTH      12

#define DIV60(t)        ((t+30)/60)     /* x/60 rounded */

#ifdef ERR
#undef ERR
#endif
#define ERR             (-1)

#define HSIZE           256             /* size of process hash table   */
#define PROCDIR         "/proc"
#define INITPROCESS     (pid_t)1        /* init process pid */
#define NONE            'n'             /* no state */
#define RUNNING         'r'             /* runnable process */
#define ZOMBIE          'z'             /* zombie process */
#define VISITED         'v'             /* marked node as visited */
#define PRINTF(a)       if (printf a < 0) { \
                perror((gettext("%s: printf failed"), prog)); \
                exit(1); }

struct uproc {
        pid_t   p_upid;                 /* process id */
        char    p_state;                /* numeric value of process state */
        dev_t   p_ttyd;                 /* controlling tty of process */
        time_t  p_time;                 /* seconds of user & system time */
        time_t  p_ctime;                /* seconds of child user & sys time */
        int     p_igintr;               /* 1 = ignores SIGQUIT and SIGINT */
        char    p_comm[PRARGSZ+1];      /* command */
        char    p_args[PRARGSZ+1];      /* command line arguments */
        struct uproc    *p_child,       /* first child pointer */
                        *p_sibling,     /* sibling pointer */
                        *p_pgrpl,       /* pgrp link */
                        *p_link;        /* hash table chain pointer */
};

/*
```

```
 *      define hash table for struct uproc
 *      Hash function uses process id
 *      and the size of the hash table(HSIZE)
 *      to determine process index into the table.
 */
static struct uproc     pr_htbl[HSIZE];

static struct   uproc   *findhash(pid_t);
static time_t   findidle(char *);
static void     clnarglist(char *);
static void     showtotals(struct uproc *);
static void     calctotals(struct uproc *);
static void     prttime(time_t, char *);
static void     prtat(time_t *time);
static void     checkampm(char *str);

static char     *prog;          /* pointer to invocation name */
static int      header = 1;     /* true if -h flag: don’t print heading */
static int      lflag = 1;      /* set if -l flag; 0 for -s flag: short form */
static char     *sel_user;      /* login of particular user selected */
static char     firstchar;      /* first char of name of prog invoked as */
static int      login;          /* true if invoked as login shell */
static time_t   now;            /* current time of day */
static time_t   uptime;         /* time of last reboot & elapsed time since */
static int      nusers;         /* number of users logged in now */
static time_t   idle;           /* number of minutes user is idle */
static time_t   jobtime;        /* total cpu time visible */
static char     doing[520];     /* process attached to terminal */
static time_t   proctime;       /* cpu time of process in doing */
static pid_t    curpid, empty;
static int      add_times;      /* boolean: add the cpu times or not */

#if SIGQUIT > SIGINT
#define ACTSIZE SIGQUIT
#else
#define ACTSIZE SIGINT
#endif

int
main(int argc, char *argv[])
{
        struct utmpx    *ut;
        struct utmpx    *utmpbegin;
        struct utmpx    *utmpend;
        struct utmpx    *utp;
        struct uproc    *up, *parent, *pgrp;
        struct psinfo   info;
        struct sigaction actinfo[ACTSIZE];
        struct pstatus  statinfo;
        size_t          size;
        struct stat     sbuf;
        DIR             *dirp;
        struct  dirent  *dp;
        char            pname[64];
        char            *fname;
        int             procfd;
        char            *cp;
        int             i;
        int             days, hrs, mins;
        int             entries;
        double          loadavg[3];

        /*
         * This program needs the proc_owner privilege
         */
        (void) __init_suid_priv(PU_CLEARLIMITSET, PRIV_PROC_OWNER,
```

```
            (char *)NULL);

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)
#define TEXT_DOMAIN "SYS_TEST"
#endif
        (void) textdomain(TEXT_DOMAIN);

        login = (argv[0][0] == ’-’);
        cp = strrchr(argv[0], ’/’);
        firstchar = login ? argv[0][1] : (cp == 0) ? argv[0][0] : cp[1];
        prog = argv[0];

        while (argc > 1) {
                if (argv[1][0] == ’-’) {
                        for (i = 1; argv[1][i]; i++) {
                                switch (argv[1][i]) {

                                case ’h’:
                                        header = 0;
                                        break;

                                case ’l’:
                                        lflag++;
                                        break;
                                case ’s’:
                                        lflag = 0;
                                        break;

                                case ’u’:
                                case ’w’:
                                        firstchar = argv[1][i];
                                        break;

                                default:
                                        (void) fprintf(stderr, gettext(
                                            "%s: bad flag %s\n"),
                                            prog, argv[1]);
                                        exit(1);
                                }
                        }
                } else {
                        if (!isalnum(argv[1][0]) || argc > 2) {
                                (void) fprintf(stderr, gettext(
                                    "usage: %s [ -hlsuw ] [ user ]\n"), prog);
                                exit(1);
                        } else
                                sel_user = argv[1];
                }
                argc--; argv++;
        }

        /*
         * read the UTMP_FILE (contains information about each logged in user)
         */
        if (stat(UTMPX_FILE, &sbuf) == ERR) {
                (void) fprintf(stderr, gettext("%s: stat error of %s: %s\n"),
                    prog, UTMPX_FILE, strerror(errno));
                exit(1);
        }
        entries = sbuf.st_size / sizeof (struct futmpx);
        size = sizeof (struct utmpx) * entries;
        if ((ut = malloc(size)) == NULL) {
                (void) fprintf(stderr, gettext("%s: malloc error of %s: %s\n"),
                    prog, UTMPX_FILE, strerror(errno));
                exit(1);
        }

        (void) utmpxname(UTMPX_FILE);

        utmpbegin = ut;
        utmpend = (struct utmpx *)((char *)utmpbegin + size);

        setutxent();
        while ((ut < utmpend) && ((utp = getutxent()) != NULL))
                (void) memcpy(ut++, utp, sizeof (*ut));
        endutxent();

        (void) time(&now);      /* get current time */

        if (header) {   /* print a header */
                prtat(&now);
                for (ut = utmpbegin; ut < utmpend; ut++) {
                        if (ut->ut_type == USER_PROCESS) {
                                if (!nonuser(*ut))
                                        nusers++;
                        } else if (ut->ut_type == BOOT_TIME) {
                                uptime = now - ut->ut_xtime;
                                uptime += 30;
                                days = uptime / (60*60*24);
                                uptime %= (60*60*24);
                                hrs = uptime / (60*60);
                                uptime %= (60*60);
                                mins = uptime / 60;

                                PRINTF((gettext("  up")));
                                if (days > 0)
                                        PRINTF((gettext(
                                            " %d day(s),"), days));
                                if (hrs > 0 && mins > 0) {
                                        PRINTF((" %2d:%02d,", hrs, mins));
                                } else {
                                        if (hrs > 0)
                                                PRINTF((gettext(
                                                    " %d hr(s),"), hrs));
                                        if (mins > 0)
                                                PRINTF((gettext(
                                                    " %d min(s),"), mins));
                                }
                        }
                }

                ut = utmpbegin; /* rewind utmp data */
                PRINTF((((nusers == 1) ?
                    gettext("  %d user") : gettext("  %d users")), nusers));
                /*
                 * Print 1, 5, and 15 minute load averages.
                 */
                (void) getloadavg(loadavg, 3);
                PRINTF((gettext(",  load average: %.2f, %.2f, %.2f\n"),
                    loadavg[LOADAVG_1MIN], loadavg[LOADAVG_5MIN],
                    loadavg[LOADAVG_15MIN]));

                if (firstchar == ’u’)   /* uptime command */
                        exit(0);

                if (lflag) {
                        PRINTF((dcgettext(NULL, "User     tty           "
                            "login@  idle   JCPU   PCPU  what\n", LC_TIME)));
                } else {
                        PRINTF((dcgettext(NULL,
                            "User     tty           idle   what\n", LC_TIME)));
```

```
                }

                if (fflush(stdout) == EOF) {
                        perror((gettext("%s: fflush failed\n"), prog));
                        exit(1);
                }
        }

        /*
         * loop through /proc, reading info about each process
         * and build the parent/child tree
         */
        if (!(dirp = opendir(PROCDIR))) {
                (void) fprintf(stderr, gettext("%s: could not open %s: %s\n"),
                    prog, PROCDIR, strerror(errno));
                exit(1);
        }

        while ((dp = readdir(dirp)) != NULL) {
                if (dp->d_name[0] == ’.’)
                        continue;
retry:
                (void) sprintf(pname, "%s/%s/", PROCDIR, dp->d_name);
                fname = pname + strlen(pname);
                (void) strcpy(fname, "psinfo");
                if ((procfd = open(pname, O_RDONLY)) < 0)
                        continue;
                if (read(procfd, &info, sizeof (info)) != sizeof (info)) {
                        int err = errno;
                        (void) close(procfd);
                        if (err == EAGAIN)
                                goto retry;
                        if (err != ENOENT)
                                (void) fprintf(stderr, gettext(
                                    "%s: read() failed on %s: %s \n"),
                                    prog, pname, strerror(err));
                        continue;
                }
                (void) close(procfd);

                up = findhash(info.pr_pid);
                up->p_ttyd = info.pr_ttydev;
                up->p_state = (info.pr_nlwp == 0? ZOMBIE : RUNNING);
                up->p_time = 0;
                up->p_ctime = 0;
                up->p_igintr = 0;
                (void) strncpy(up->p_comm, info.pr_fname,
                    sizeof (info.pr_fname));
                up->p_args[0] = 0;

                if (up->p_state != NONE && up->p_state != ZOMBIE) {
                        (void) strcpy(fname, "status");

                        /* now we need the proc_owner privilege */
                        (void) __priv_bracket(PRIV_ON);

                        procfd = open(pname, O_RDONLY);

                        /* drop proc_owner privilege after open */
                        (void) __priv_bracket(PRIV_OFF);

                        if (procfd < 0)
                                continue;

                        if (read(procfd, &statinfo, sizeof (statinfo))
                            != sizeof (statinfo)) {
                                int err = errno;
                                (void) close(procfd);
                                if (err == EAGAIN)
                                        goto retry;
                                if (err != ENOENT)
                                        (void) fprintf(stderr, gettext(
                                            "%s: read() failed on %s: %s \n"),
                                            prog, pname, strerror(err));
                                continue;
                        }
                        (void) close(procfd);

                        up->p_time = statinfo.pr_utime.tv_sec +
                            statinfo.pr_stime.tv_sec;   /* seconds */
                        up->p_ctime = statinfo.pr_cutime.tv_sec +
                            statinfo.pr_cstime.tv_sec;

                        (void) strcpy(fname, "sigact");

                        /* now we need the proc_owner privilege */
                        (void) __priv_bracket(PRIV_ON);

                        procfd = open(pname, O_RDONLY);

                        /* drop proc_owner privilege after open */
                        (void) __priv_bracket(PRIV_OFF);

                        if (procfd < 0)
                                continue;

                        if (read(procfd, actinfo, sizeof (actinfo))
                            != sizeof (actinfo)) {
                                int err = errno;
                                (void) close(procfd);
                                if (err == EAGAIN)
                                        goto retry;
                                if (err != ENOENT)
                                        (void) fprintf(stderr, gettext(
                                            "%s: read() failed on %s: %s \n"),
                                            prog, pname, strerror(err));
                                continue;
                        }
                        (void) close(procfd);

                        up->p_igintr =
                            actinfo[SIGINT-1].sa_handler == SIG_IGN &&
                            actinfo[SIGQUIT-1].sa_handler == SIG_IGN;

                        /*
                         * Process args.
                         */
                        up->p_args[0] = 0;
                        clnarglist(info.pr_psargs);
                        (void) strcat(up->p_args, info.pr_psargs);
                        if (up->p_args[0] == 0 ||
                            up->p_args[0] == '-' && up->p_args[1] <= ' ' ||
                            up->p_args[0] == '?') {
                                (void) strcat(up->p_args, " (");
                                (void) strcat(up->p_args, up->p_comm);
                                (void) strcat(up->p_args, ")");
                        }
                }

                /*
                 * link pgrp together in case parents go away
                 * Pgrp chain is a single linked list originating
```

```
                 * from the pgrp leader to its group member.
                 */
                if (info.pr_pgid != info.pr_pid) {      /* not pgrp leader */
                        pgrp = findhash(info.pr_pgid);
                        up->p_pgrpl = pgrp->p_pgrpl;
                        pgrp->p_pgrpl = up;
                }
                parent = findhash(info.pr_ppid);

                /* if this is the new member, link it in */
                if (parent->p_upid != INITPROCESS) {
                        if (parent->p_child) {
                                up->p_sibling = parent->p_child;
                                up->p_child = 0;
                        }
                        parent->p_child = up;
                }
        }

        /* revert to non-privileged user after opening */
        (void) __priv_relinquish();

        (void) closedir(dirp);
        (void) time(&now);      /* get current time */

        /*
         * loop through utmpx file, printing process info
         * about each logged in user
         */
        for (ut = utmpbegin; ut < utmpend; ut++) {
                if (ut->ut_type != USER_PROCESS)
                        continue;
                if (sel_user && strncmp(ut->ut_name, sel_user, NMAX) != 0)
                        continue;       /* we're looking for somebody else */

                /* print login name of the user */
                PRINTF(("%-*.*s ", LOGIN_WIDTH, NMAX, ut->ut_name));
                PRINTF(("%-*.*s ", NMAX, NMAX, ut->ut_name));

                /* print tty user is on */
                if (lflag) {
                        PRINTF(("%-*.*s", LINE_WIDTH, LMAX, ut->ut_line));
                        PRINTF(("%-*.*s", LMAX, LMAX, ut->ut_line));
                } else {
                        if (ut->ut_line[0] == 'p' && ut->ut_line[1] == 't' &&
                            ut->ut_line[2] == 's' && ut->ut_line[3] == '/') {
                                PRINTF(("%-*.3s", LMAX, &ut->ut_line[4]));
                        } else {
                                PRINTF(("%-*.*s", LINE_WIDTH, LMAX,
                                    ut->ut_line));
                                PRINTF(("%-*.*s", LMAX, LMAX, ut->ut_line));
                        }
                }

                /* print when the user logged in */
                if (lflag) {
                        time_t tim = ut->ut_xtime;
                        prtat(&tim);
                }

                /* print idle time */
                idle = findidle(ut->ut_line);
                if (idle >= 36 * 60) {
                        PRINTF((dcgettext(NULL, "%2ddays ", LC_TIME),
                            (idle + 12 * 60) / (24 * 60)));
                } else
```

```
                                prttime(idle, " ");
                        showtotals(findhash(ut->ut_pid));
                }
                if (fclose(stdout) == EOF) {
                        perror((gettext("%s: fclose failed"), prog));
                        exit(1);
                }
                return (0);
        }
```

______***unchanged_portion_omitted_***

```
**********************************************************
   11107 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/wall/wall.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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 (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */



/*
 * Copyright 1988-2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2012 Joyent, Inc. All rights reserved.
 *
 * Copyright (c) 2013 Gary Mills
 */

#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <grp.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <sys/stat.h>
#include <utmpx.h>
#include <sys/utsname.h>
#include <dirent.h>
#include <pwd.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <locale.h>
#include <syslog.h>
#include <sys/wait.h>
#include <limits.h>
#include <libzonecfg.h>
#include <zone.h>
#include <sys/contract/process.h>
```

```
#include <libcontract.h>
#include <sys/ctfs.h>

/*
 * Use the full lengths from utmpx for user and line.
 * utmpx defines wider fields for user and line.  For compatibility of output,
 * we are limiting these to the old maximums in utmp. Define UTMPX_NAMELEN
 * to use the full lengths.
 */
#define NMAX    (sizeof (((struct utmpx *)0)->ut_user))
#define LMAX    (sizeof (((struct utmpx *)0)->ut_line))
#ifndef UTMPX_NAMELEN
/* XXX - utmp -fix name length */
#define NMAX    (_POSIX_LOGIN_NAME_MAX - 1)
#define LMAX    12
#else /* UTMPX_NAMELEN */
#define NMAX    (sizeof (((struct utmpx *)0)->ut_user)
#define LMAX    (sizeof (((struct utmpx *)0)->ut_line)
#endif /* UTMPX_NAMELEN */

static char     mesg[3000];
static char     *infile;
static int      gflag;
static struct   group *pgrp;
static char     *grpname;
static char     line[MAXNAMLEN+1] = "???";
static char     systm[MAXNAMLEN+1];
static time_t   tloc;
static struct   utsname utsn;
static char     who[NMAX+1]     = "???";
static char     who[9]  = "???";
static char     time_buf[50];
#define DATE_FMT        "%a %b %e %H:%M:%S"

static void sendmes(struct utmpx *, zoneid_t);
static void sendmes_tozone(zoneid_t, int);
static int chkgrp(char *);
static char *copy_str_till(char *, char *, char, int);

static int init_template(void);
int contract_abandon_id(ctid_t);

int
main(int argc, char *argv[])
{
        FILE    *f;
        char    *ptr, *start;
        struct  passwd *pwd;
        char    *term_name;
        int     c;
        int     aflag = 0;
        int     errflg = 0;
        int zflg = 0;
        int Zflg = 0;

        char *zonename = NULL;
        zoneid_t *zoneidlist = NULL;
        uint_t nzids_saved, nzids = 0;

        (void) setlocale(LC_ALL, "");

        while ((c = getopt(argc, argv, "g:az:Z")) != EOF)
                switch (c) {
                case 'a':
                        aflag++;
                        break;
                case ’g’:
                        if (gflag) {
                                (void) fprintf(stderr,
                                    "Only one group allowed\n");
                                return (1);
                        }
                        if ((pgrp = getgrnam(grpname = optarg)) == NULL) {
                                (void) fprintf(stderr, "Unknown group %s\n",
                                    grpname);
                                return (1);
                        }
                        gflag++;
                        break;
                case ’z’:
                        zflg++;
                        zonename = optarg;
                        if (getzoneidbyname(zonename) == -1) {
                                (void) fprintf(stderr, "Specified zone %s "
                                    "is invalid", zonename);
                                return (1);
                        }
                        break;
                case ’Z’:
                        Zflg++;
                        break;
                case ’?’:
                        errflg++;
                        break;
                }

        if (errflg) {
                (void) fprintf(stderr,
                    "Usage: wall [-a] [-g group] [-z zone] [-Z] [files...]\n");
                return (1);
        }

        if (zflg && Zflg) {
                (void) fprintf(stderr, "Cannot use -z with -Z\n");
                return (1);
        }

        if (optind < argc)
                infile = argv[optind];

        if (uname(&utsn) == -1) {
                (void) fprintf(stderr, "wall: uname() failed, %s\n",
                    strerror(errno));
                return (2);
        }
        (void) strcpy(systm, utsn.nodename);

        /*
         * Get the name of the terminal wall is running from.
         */

        if ((term_name = ttyname(fileno(stderr))) != NULL) {
                /*
                 * skip the leading "/dev/" in term_name
                 */
                (void) strncpy(line, &term_name[5], sizeof (line) - 1);
        }

        if (who[0] == ’?’) {
                if (pwd = getpwuid(getuid()))
                        (void) strncpy(&who[0], pwd->pw_name, sizeof (who));
        }
```

```
        f = stdin;
        if (infile) {
                f = fopen(infile, "r");
                if (f == NULL) {
                        (void) fprintf(stderr, "Cannot open %s\n", infile);
                        return (1);
                }
        }

        start = &mesg[0];
        ptr = start;
        while ((ptr - start) < 3000) {
                size_t n;

                if (fgets(ptr, &mesg[sizeof (mesg)] - ptr, f) == NULL)
                        break;
                if ((n = strlen(ptr)) == 0)
                        break;
                ptr += n;
        }
        (void) fclose(f);

        /*
         * If the request is from the rwall daemon then use the caller’s
         * name and host.  We determine this if all of the following is true:
         *      1) First 5 characters are "From "
         *      2) Next non-white characters are of the form "name@host:"
         */
        if (strcmp(line, "???") == 0) {
                char rwho[MAXNAMLEN+1];
                char rsystm[MAXNAMLEN+1];
                char *cp;

                if (strncmp(mesg, "From ", 5) == 0) {
                        cp = &mesg[5];
                        cp = copy_str_till(rwho, cp, ’@’, MAXNAMLEN + 1);
                        if (rwho[0] != ’\0’) {
                                cp = copy_str_till(rsystm, ++cp, ’:’,
                                    MAXNAMLEN + 1);
                                if (rsystm[0] != ’\0’) {
                                        (void) strcpy(systm, rsystm);
                                        (void) strncpy(rwho, who,
                                            sizeof (who));
                                        (void) strncpy(rwho, who, 9);
                                        (void) strcpy(line, "rpc.rwalld");
                                }
                        }
                }
        }
        (void) time(&tloc);
        (void) strftime(time_buf, sizeof (time_buf),
            DATE_FMT, localtime(&tloc));

        if (zflg != 0) {
                if ((zoneidlist =
                    malloc(sizeof (zoneid_t))) == NULL ||
                    (*zoneidlist = getzoneidbyname(zonename)) == -1)
                        return (errno);
                nzids = 1;
        } else if (Zflg != 0) {
                if (zone_list(NULL, &nzids) != 0)
                        return (errno);
again:
                nzids *= 2;
                if ((zoneidlist = malloc(nzids * sizeof (zoneid_t))) == NULL)
```

```
                        exit(errno);
                nzids_saved = nzids;
                if (zone_list(zoneidlist, &nzids) != 0) {
                        (void) free(zoneidlist);
                        return (errno);
                }
                if (nzids > nzids_saved) {
                        free(zoneidlist);
                        goto again;
                }
        }
        if (zflg || Zflg) {
                for (; nzids > 0; --nzids)
                        sendmes_tozone(zoneidlist[nzids-1], aflag);
                free(zoneidlist);
        } else
                sendmes_tozone(getzoneid(), aflag);

        return (0);
}
______unchanged_portion_omitted_

/*
 * Note to future maintainers: with the change of wall to use the
 * getutxent() API, the forked children (created by this function)
 * must call _exit as opposed to exit. This is necessary to avoid
 * unwanted fflushing of getutxent's stdio stream (caused by atexit
 * processing).
 */
static void
sendmes(struct utmpx *p, zoneid_t zid)
{
        int i;
        char *s;
        static char device[LMAX + 6];
        char *bp;
        int ibp;
        FILE *f;
        int fd, tmpl_fd;
        boolean_t zoneenter = B_FALSE;

        if (zid != getzoneid()) {
                zoneenter = B_TRUE;
                tmpl_fd = init_template();
                if (tmpl_fd == -1) {
                        (void) fprintf(stderr, "Could not initialize "
                            "process contract");
                        return;
                }
        }

        while ((i = (int)fork()) == -1) {
                (void) alarm(60);
                (void) wait((int *)0);
                (void) alarm(0);
        }

        if (i)
                return;

        if (zoneenter && zone_enter(zid) == -1) {
                char zonename[ZONENAME_MAX];
                (void) getzonenamebyid(zid, zonename, ZONENAME_MAX);
                (void) fprintf(stderr, "Could not enter zone "
                    "%s\n", zonename);
        }
```

```
        if (zoneenter)
                (void) ct_tmpl_clear(tmpl_fd);

        if (gflag)
                if (!chkgrp(p->ut_user))
                        _exit(0);

        (void) signal(SIGHUP, SIG_IGN);
        (void) alarm(60);
        s = &device[0];
        (void) snprintf(s, sizeof (device), "/dev/%.*s", LMAX, p->ut_line);

        /* check if the device is really a tty */
        if ((fd = open(s, O_WRONLY|O_NOCTTY|O_NONBLOCK)) == -1) {
                (void) fprintf(stderr, "Cannot send to %.*s on %s\n",
                    NMAX, p->ut_user, s);
                perror("open");
                (void) fflush(stderr);
                _exit(1);
        } else {
                if (!isatty(fd)) {
                        (void) fprintf(stderr,
                            "Cannot send to device %.*s %s\n",
                            LMAX, p->ut_line,
                            "because it's not a tty");
                        openlog("wall", 0, LOG_AUTH);
                        syslog(LOG_CRIT, "%.*s in utmpx is not a tty\n",
                            LMAX, p->ut_line);
                        closelog();
                        (void) fflush(stderr);
                        _exit(1);
                }
        }
#ifdef DEBUG
        (void) close(fd);
        f = fopen("wall.debug", "a");
#else
        f = fdopen(fd, "w");
#endif
        if (f == NULL) {
                (void) fprintf(stderr, "Cannot send to %-.*s on %s\n",
                    NMAX, &p->ut_user[0], s);
                perror("open");
                (void) fflush(stderr);
                _exit(1);
        }
        (void) fprintf(f,
            "\07\07\07Broadcast Message from %s (%s) on %s %19.19s",
            who, line, systm, time_buf);
        if (gflag)
                (void) fprintf(f, " to group %s", grpname);
        (void) fprintf(f, "...\n");
#ifdef DEBUG
        (void) fprintf(f, "DEBUG: To %.*s on %s\n", NMAX, p->ut_user, s);
        (void) fprintf(f, "DEBUG: To %.8s on %s\n", p->ut_user, s);
#endif
        i = strlen(mesg);
        for (bp = mesg; --i >= 0; bp++) {
                ibp = (unsigned int)((unsigned char) *bp);
                if (*bp == '\n')
                        (void) putc('\r', f);
                if (isprint(ibp) || *bp == '\r' || *bp == '\013' ||
                    *bp == ' ' || *bp == '\t' || *bp == '\n' || *bp == '\007') {
                        (void) putc(*bp, f);
                } else {
                        if (!isascii(*bp)) {
                                (void) fputs("M-", f);
                                *bp = toascii(*bp);
                        }
                        if (iscntrl(*bp)) {
                                (void) putc('^', f);
                                (void) putc(*bp + 0100, f);
                        }
                        else
                                (void) putc(*bp, f);
                }

                if (*bp == '\n')
                        (void) fflush(f);

                if (ferror(f) || feof(f)) {
                        (void) printf("\n\007Write failed\n");
                        (void) fflush(stdout);
                        _exit(1);
                }
        }
        (void) fclose(f);
        (void) close(fd);
        _exit(0);
}


static int
chkgrp(char *name)
{
        int i;
        char user[NMAX + 1];
        char *p;

        (void) strncpy(user, name, NMAX);
        user[NMAX] = '\0';
        for (i = 0; pgrp->gr_mem[i] && pgrp->gr_mem[i][0]; i++) {
                if (strcmp(user, pgrp->gr_mem[i]) == 0)
                for (p = name; *p && *p != ' '; p++)
                ;
                *p = 0;
                if (strncmp(name, pgrp->gr_mem[i], 8) == 0)
                        return (1);
        }

        return (0);
}
```

______unchanged_portion_omitted_

```
**********************************************************
   20484 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/who/who.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 1984, 1986, 1987, 1988, 1989 AT&T */
/*	  All Rights Reserved	*/


/*
 * Copyright (c) 2013 Gary Mills
 *
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 *	This program analyzes information found in /var/adm/utmpx
 *
 *	Additionally information is gathered from /etc/inittab
 *	if requested.
 *
 *
 *	Syntax:
 *
 *		who am i	Displays info on yourself
 *
 *		who -a		Displays information about All
 *				entries in /var/adm/utmpx
 *
 *		who -b		Displays info on last boot
 *
 *		who -d		Displays info on DEAD PROCESSES
 *
 *		who -H		Displays HEADERS for output
 *
 *		who -l		Displays info on LOGIN entries
 *
 *		who -m		Same as who am i
 *
 *		who -p		Displays info on PROCESSES spawned by init
 *
 *		who -q		Displays short information on
```

```
 *				current users who LOGGED ON
 *
 *		who -r		Displays info of current run-level
 *
 *		who -s		Displays requested info in SHORT form
 *
 *		who -t		Displays info on TIME changes
 *
 *		who -T		Displays writeability of each user
 *				(+ writeable, - non-writeable, ? hung)
 *
 *		who -u		Displays LONG info on users
 *				who have LOGGED ON
 */

#define		DATE_FMT	"%b %e %H:%M"

/*
 *  %b  Abbreviated month name
 *  %e  Day of month
 *  %H  hour (24-hour clock)
 *  %M  minute
 */
#include	<errno.h>
#include	<fcntl.h>
#include	<stdio.h>
#include	<string.h>
#include	<sys/types.h>
#include	<unistd.h>
#include	<stdlib.h>
#include	<sys/stat.h>
#include	<time.h>
#include	<utmpx.h>
#include	<locale.h>
#include	<pwd.h>
#include	<limits.h>

static void process(void);
static void ck_file(char *);
static void dump(void);

static struct	utmpx *utmpp;	/* pointer for getutxent()	*/

/*
 * Use the full lengths from utmpx for user and line.
 * utmpx defines wider fields for user and line.  For compatibility of output,
 * we are limiting these to the old maximums in utmp. Define UTMPX_NAMELEN
 * to use the full lengths.
 */
#ifndef UTMPX_NAMELEN
/* XXX - utmp - fix name length */
#define	NMAX	(_POSIX_LOGIN_NAME_MAX - 1)
#define	LMAX	12
#else /* UTMPX_NAMELEN */
#define	NMAX	(sizeof (utmpp->ut_user))
#define	LMAX	(sizeof (utmpp->ut_line))
#endif

/* Print minimum field widths. */
#define	LOGIN_WIDTH	8
#define	LINE_WIDTH	12

static char	comment[BUFSIZ]; /* holds inittab comment	*/
static char	errmsg[BUFSIZ];	/* used in snprintf for errors	*/
static int	fildes;		/* file descriptor for inittab	*/
static int	Hopt = 0;	/* 1 = who -H			*/
static char     *inittab;       /* ptr to inittab contents      */
static char     *iinit;         /* index into inittab           */
static int      justme = 0;     /* 1 = who am i                 */
static struct   tm *lptr;       /* holds user login time        */
static char     *myname;        /* pointer to invoker's name    */
static char     *mytty;         /* holds device user is on      */
static char     nameval[sizeof (utmpp->ut_user) + 1]; /*  invoker's name */
static int      number = 8;     /* number of users per -q line  */
static int      optcnt = 0;     /* keeps count of options       */
static char     outbuf[BUFSIZ]; /* buffer for output            */
static char     *program;       /* holds name of this program   */
#ifdef  XPG4
static int      aopt = 0;       /* 1 = who -a                   */
static int      dopt = 0;       /* 1 = who -d                   */
#endif  /* XPG4 */
static int      qopt = 0;       /* 1 = who -q                   */
static int      sopt = 0;       /* 1 = who -s                   */
static struct   stat stbuf;     /* area for stat buffer         */
static struct   stat *stbufp;   /* ptr to structure             */
static int      terse = 1;      /* 1 = print terse msgs         */
static int      Topt = 0;       /* 1 = who -T                   */
static time_t   timnow;         /* holds current time           */
static int      totlusrs = 0;   /* cntr for users on system     */
static int      uopt = 0;       /* 1 = who -u                   */
static char     user[sizeof (utmpp->ut_user) + 1]; /* holds user name */
static int      validtype[UTMAXTYPE+1]; /* holds valid types    */
static int      wrap;           /* flag to indicate wrap        */
static char     time_buf[128];  /* holds date and time string   */
static char     *end;           /* used in strtol for end pointer */

int
main(int argc, char **argv)
{
        int     goerr = 0;      /* non-zero indicates cmd error */
        int     i;
        int     optsw;          /* switch for while of getopt() */

        (void) setlocale(LC_ALL, "");

#if     !defined(TEXT_DOMAIN)   /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"  /* Use this only if it weren't */
#endif
        (void) textdomain(TEXT_DOMAIN);

        validtype[USER_PROCESS] = 1;
        validtype[EMPTY] = 0;
        stbufp = &stbuf;

        /*
         *      Strip off path name of this command
         */
        for (i = strlen(argv[0]); i >= 0 && argv[0][i] != '/'; --i)
                ;
        for (i = strlen(argv[0]); i >= 0 && argv[0][i] != '/'; --i);
        if (i >= 0)
                argv[0] += i+1;
        program = argv[0];

        /*
         *      Buffer stdout for speed
         */
        setbuf(stdout, outbuf);

        /*
         *      Retrieve options specified on command line
         *      XCU4 - add -m option
```

```
         */
        while ((optsw = getopt(argc, argv, "abdHlmn:pqrstTu")) != EOF) {
                optcnt++;
                switch (optsw) {

                case 'a':
                        optcnt += 7;
                        validtype[BOOT_TIME] = 1;
                        validtype[DEAD_PROCESS] = 1;
                        validtype[LOGIN_PROCESS] = 1;
                        validtype[INIT_PROCESS] = 1;
                        validtype[RUN_LVL] = 1;
                        validtype[OLD_TIME] = 1;
                        validtype[NEW_TIME] = 1;
                        validtype[USER_PROCESS] = 1;
#ifdef  XPG4
                        aopt = 1;
#endif  /* XPG4 */
                        uopt = 1;
                        Topt = 1;
                        if (!sopt) terse = 0;
                        break;

                case 'b':
                        validtype[BOOT_TIME] = 1;
                        if (!uopt) validtype[USER_PROCESS] = 0;
                        break;

                case 'd':
                        validtype[DEAD_PROCESS] = 1;
                        if (!uopt) validtype[USER_PROCESS] = 0;
#ifdef  XPG4
                        dopt = 1;
#endif  /* XPG4 */
                        break;

                case 'H':
                        optcnt--; /* Don't count Header */
                        Hopt = 1;
                        break;

                case 'l':
                        validtype[LOGIN_PROCESS] = 1;
                        if (!uopt) validtype[USER_PROCESS] = 0;
                        terse = 0;
                        break;
                case 'm':               /* New XCU4 option */
                        justme = 1;
                        break;

                case 'n':
                        errno = 0;
                        number = strtol(optarg, &end, 10);
                        if (errno != 0 || *end != '\0') {
                                (void) fprintf(stderr, gettext(
                                    "%s: Invalid numeric argument\n"),
                                    program);
                                exit(1);
                        }
                        if (number < 1) {
                                (void) fprintf(stderr, gettext(
                                    "%s: Number of users per line must "
                                    "be at least 1\n"), program);
                                exit(1);
                        }
                        break;
                case 'p':
                        validtype[INIT_PROCESS] = 1;
                        if (!uopt) validtype[USER_PROCESS] = 0;
                        break;

                case 'q':
                        qopt = 1;
                        break;

                case 'r':
                        validtype[RUN_LVL] = 1;
                        terse = 0;
                        if (!uopt) validtype[USER_PROCESS] = 0;
                        break;

                case 's':
                        sopt = 1;
                        terse = 1;
                        break;

                case 't':
                        validtype[OLD_TIME] = 1;
                        validtype[NEW_TIME] = 1;
                        if (!uopt) validtype[USER_PROCESS] = 0;
                        break;

                case 'T':
                        Topt = 1;
#ifdef  XPG4
                        terse = 1;      /* XPG4 requires -T */
#else   /* XPG4 */
                        terse = 0;
#endif  /* XPG4 */
                        break;

                case 'u':
                        uopt = 1;
                        validtype[USER_PROCESS] = 1;
                        if (!sopt) terse = 0;
                        break;

                case '?':
                        goerr++;
                        break;
                default:
                        break;
                }
        }
#ifdef  XPG4
        /*
         * XCU4 changes - check for illegal sopt, Topt & aopt combination
         */
        if (sopt == 1) {
                terse = 1;
                if (Topt == 1 || aopt == 1)
                goerr++;
        }
#endif  /* XPG4 */

        if (goerr > 0) {
#ifdef  XPG4
                /*
                 * XCU4 - slightly different usage with -s -a & -T
                 */
                (void) fprintf(stderr, gettext("\nUsage:\t%s"), program);
```

```
                (void) fprintf(stderr,
                    gettext(" -s [-bdHlmpqrtu] [utmpx_like_file]\n"));

                (void) fprintf(stderr, gettext(
                    "\t%s [-abdHlmpqrtTu] [utmpx_like_file]\n"), program);
#else   /* XPG4 */
                (void) fprintf(stderr, gettext(
                    "\nUsage:\t%s [-abdHlmpqrstTu] [utmpx_like_file]\n"),
                    program);
#endif  /* XPG4 */
                (void) fprintf(stderr,
                    gettext("\t%s -q [-n x] [utmpx_like_file]\n"), program);
                (void) fprintf(stderr, gettext("\t%s [am i]\n"), program);
                /*
                 * XCU4 changes - be explicit with "am i" options
                 */
                (void) fprintf(stderr, gettext("\t%s [am I]\n"), program);
                (void) fprintf(stderr, gettext(
                    "a\tall (bdlprtu options)\n"));
                (void) fprintf(stderr, gettext("b\tboot time\n"));
                (void) fprintf(stderr, gettext("d\tdead processes\n"));
                (void) fprintf(stderr, gettext("H\tprint header\n"));
                (void) fprintf(stderr, gettext("l\tlogin processes\n"));
                (void) fprintf(stderr, gettext(
                    "n #\tspecify number of users per line for -q\n"));
                (void) fprintf(stderr,
                    gettext("p\tprocesses other than getty or users\n"));
                (void) fprintf(stderr, gettext("q\tquick %s\n"), program);
                (void) fprintf(stderr, gettext("r\trun level\n"));
                (void) fprintf(stderr, gettext(
                "s\tshort form of %s (no time since last output or pid)\n"),
                    program);
                (void) fprintf(stderr, gettext("t\ttime changes\n"));
                (void) fprintf(stderr, gettext(
                    "T\tstatus of tty (+ writable, - not writable, "
                    "? hung)\n"));
                (void) fprintf(stderr, gettext("u\tuseful information\n"));
                (void) fprintf(stderr,
                    gettext("m\tinformation only about current terminal\n"));
                (void) fprintf(stderr, gettext(
                    "am i\tinformation about current terminal "
                    "(same as -m)\n"));
                (void) fprintf(stderr, gettext(
                    "am I\tinformation about current terminal "
                    "(same as -m)\n"));
                exit(1);
        }

        /*
         * XCU4: If -q option ignore all other options
         */
        if (qopt == 1) {
                Hopt = 0;
                sopt = 0;
                Topt = 0;
                uopt = 0;
                justme = 0;
                validtype[ACCOUNTING] = 0;
                validtype[BOOT_TIME] = 0;
                validtype[DEAD_PROCESS] = 0;
                validtype[LOGIN_PROCESS] = 0;
                validtype[INIT_PROCESS] = 0;
                validtype[RUN_LVL] = 0;
                validtype[OLD_TIME] = 0;
                validtype[NEW_TIME] = 0;
                validtype[USER_PROCESS] = 1;
```

```c
        }

        if (argc == optind + 1) {
                optcnt++;
                ck_file(argv[optind]);
                (void) utmpxname(argv[optind]);
        }

        /*
         *      Test for 'who am i' or 'who am I'
         *      XCU4 - check if justme was already set by -m option
         */
        if (justme == 1 || (argc == 3 && strcmp(argv[1], "am") == 0 &&
            ((argv[2][0] == 'i' || argv[2][0] == 'I') &&
            argv[2][1] == '\0'))) {
                justme = 1;
                myname = nameval;
                (void) cuserid(myname);
                if ((mytty = ttyname(fileno(stdin))) == NULL &&
                    (mytty = ttyname(fileno(stdout))) == NULL &&
                    (mytty = ttyname(fileno(stderr))) == NULL) {
                        (void) fprintf(stderr, gettext(
                        "Must be attached to terminal for 'am I' option\n"));
                        (void) fflush(stderr);
                        exit(1);
                } else
                        mytty += 5; /* bump past "/dev/" */
        }

        if (!terse) {
                if (Hopt)
                        (void) printf(gettext(
        "NAME       LINE         TIME          IDLE    PID  COMMENTS\n"));

                timnow = time(0);

                if ((fildes = open("/etc/inittab",
                    O_NONBLOCK|O_RDONLY)) == -1) {
                        (void) snprintf(errmsg, sizeof (errmsg),
                            gettext("%s: Cannot open /etc/inittab"), program);
                        perror(errmsg);
                        exit(errno);
                }

                if (fstat(fildes, stbufp) == -1) {
                        (void) snprintf(errmsg, sizeof (errmsg),
                            gettext("%s: Cannot stat /etc/inittab"), program);
                        perror(errmsg);
                        exit(errno);
                }

                if ((inittab = malloc(stbufp->st_size + 1)) == NULL) {
                        (void) snprintf(errmsg, sizeof (errmsg),
                            gettext("%s: Cannot allocate %ld bytes"),
                            program, stbufp->st_size);
                        perror(errmsg);
                        exit(errno);
                }

                if (read(fildes, inittab, stbufp->st_size)
                    != stbufp->st_size) {
                        (void) snprintf(errmsg, sizeof (errmsg),
                            gettext("%s: Error reading /etc/inittab"),
                            program);
                        perror(errmsg);
                        exit(errno);
```

```c
                }

                inittab[stbufp->st_size] = '\0';
                iinit = inittab;
        } else {
                if (Hopt) {
#ifdef  XPG4
                        if (dopt) {
                                (void) printf(gettext(
                        "NAME       LINE         TIME           COMMENTS\n"));
                        } else {
                                (void) printf(
                                    gettext("NAME       LINE         TIME\n"));
                        }
#else   /* XPG4 */
                        (void) printf(
                            gettext("NAME       LINE         TIME\n"));
#endif  /* XPG4 */
                }
        }
        process();

        /*
         *      'who -q' requires EOL upon exit,
         *      followed by total line
         */
        if (qopt)
                (void) printf(gettext("\n# users=%d\n"), totlusrs);
        return (0);
}

static void
dump()
{
        char    device[sizeof (utmpp->ut_line) + 1];
        time_t hr;
        time_t  idle;
        time_t min;
        char    path[sizeof (utmpp->ut_line) + 6];
        int     pexit;
        int     pterm;
        int     rc;
        char    w;      /* writeability indicator */

        /*
         * Get and check user name
         */
        if (utmpp->ut_user[0] == '\0')
                (void) strcpy(user, "   .");
        else {
                (void) strncpy(user, utmpp->ut_user, sizeof (user));
                user[sizeof (user) - 1] = '\0';
        }
        totlusrs++;

        /*
         * Do print in 'who -q' format
         */
        if (qopt) {
                /*
                 * XCU4 - Use non user macro for correct user count
                 */
                if (((totlusrs - 1) % number) == 0 && totlusrs > 1)
                        (void) printf("\n");
                (void) printf("%-*.*s ", LOGIN_WIDTH, NMAX, user);
                (void) printf("%-*s ", NMAX, user);
                return;
        }


        pexit = (int)' ';
        pterm = (int)' ';

        /*
         *      Get exit info if applicable
         */
        if (utmpp->ut_type == RUN_LVL || utmpp->ut_type == DEAD_PROCESS) {
                pterm = utmpp->ut_exit.e_termination;
                pexit = utmpp->ut_exit.e_exit;
        }

        /*
         *      Massage ut_xtime field
         */
        lptr = localtime(&utmpp->ut_xtime);
        (void) strftime(time_buf, sizeof (time_buf),
            dcgettext(NULL, DATE_FMT, LC_TIME), lptr);

        /*
         *      Get and massage device
         */
        if (utmpp->ut_line[0] == '\0')
                (void) strcpy(device, "     .");
        else {
                (void) strncpy(device, utmpp->ut_line,
                    sizeof (utmpp->ut_line));
                device[sizeof (utmpp->ut_line)] = '\0';
        }

        /*
         *      Get writeability if requested
         *      XCU4 - only print + or - for user processes
         */
        if (Topt && (utmpp->ut_type == USER_PROCESS)) {
                w = '-';
                (void) strcpy(path, "/dev/");
                (void) strncpy(path + 5, utmpp->ut_line,
                    sizeof (utmpp->ut_line));
                path[5 + sizeof (utmpp->ut_line)] = '\0';

                if ((rc = stat(path, stbufp)) == -1) w = '?';
                else if ((stbufp->st_mode & S_IWOTH) ||
                    (stbufp->st_mode & S_IWGRP))  /* Check group & other */
                        w = '+';

        } else
                w = ' ';

        /*
         *      Print the TERSE portion of the output
         */
        (void) printf("%-*.*s %c %-12s %s", LOGIN_WIDTH, NMAX, user,
            w, device, time_buf);
        (void) printf("%-*s %c %-12s %s", NMAX, user, w, device, time_buf);

        if (!terse) {
                /*
                 *      Stat device for idle time
                 *      (Don't complain if you can't)
                 */
                rc = -1;
                if (utmpp->ut_type == USER_PROCESS) {
```

```
                        (void) strcpy(path, "/dev/");
                        (void) strncpy(path + 5, utmpp->ut_line,
                            sizeof (utmpp->ut_line));
                        path[5 + sizeof (utmpp->ut_line)] = '\0';
                        rc = stat(path, stbufp);
                }
                if (rc != -1) {
                        idle = timnow - stbufp->st_mtime;
                        hr = idle/3600;
                        min = (unsigned)(idle/60)%60;
                        if (hr == 0 && min == 0)
                                (void) printf(gettext("   .  "));
                        else {
                                if (hr < 24)
                                        (void) printf(" %2d:%2.2d", (int)hr,
                                            (int)min);
                                else
                                        (void) printf(gettext("  old "));
                        }
                }

                /*
                 *      Add PID for verbose output
                 */
                if (utmpp->ut_type != BOOT_TIME &&
                    utmpp->ut_type != RUN_LVL &&
                    utmpp->ut_type != ACCOUNTING)
                        (void) printf("  %5ld", utmpp->ut_pid);

                /*
                 *      Handle /etc/inittab comment
                 */
                if (utmpp->ut_type == DEAD_PROCESS) {
                        (void) printf(gettext("  id=%4.4s "),
                            utmpp->ut_id);
                        (void) printf(gettext("term=%-3d "), pterm);
                        (void) printf(gettext("exit=%d  "), pexit);
                } else if (utmpp->ut_type != INIT_PROCESS) {
                        /*
                         *      Search for each entry in inittab
                         *      string. Keep our place from
                         *      search to search to try and
                         *      minimize the work. Wrap once if needed
                         *      for each entry.
                         */
                        wrap = 0;
                        /*
                         *      Look for a line beginning with
                         *      utmpp->ut_id
                         */
                        while ((rc = strncmp(utmpp->ut_id, iinit,
                            strcspn(iinit, ":"))) != 0) {
                                for (; *iinit != '\n'; iinit++)
                                        ;
                                for (; *iinit != '\n'; iinit++);
                                iinit++;

                                /*
                                 *      Wrap once if necessary to
                                 *      find entry in inittab
                                 */
                                if (*iinit == '\0') {
                                        if (!wrap) {
                                                iinit = inittab;
                                                wrap = 1;
                                        }
                                }
                        }

                        if (*iinit != ’\0’) {
                                /*
                                 *      We found our entry
                                 */
                                for (iinit++; *iinit != ’#’ &&
                                    *iinit != ’\n’; iinit++)
                                        ;
                                         *iinit != ’\n’; iinit++);
                                if (*iinit == ’#’) {
                                        for (iinit++; *iinit == ’ ’ ||
                                            *iinit == ’\t’; iinit++)
                                                ;
                                                 *iinit == ’\t’; iinit++);
                                        for (rc = 0; *iinit != ’\n’; iinit++)
                                                comment[rc++] = *iinit;
                                        comment[rc] = ’\0’;
                                } else
                                        (void) strcpy(comment, " ");

                                (void) printf("  %s", comment);
                        } else
                                iinit = inittab;        /* Reset pointer */
                }
                if (utmpp->ut_type == INIT_PROCESS)
                        (void) printf(gettext("  id=%4.4s"), utmpp->ut_id);
        }
#ifdef  XPG4
        else
                if (dopt && utmpp->ut_type == DEAD_PROCESS) {
                        (void) printf(gettext("\tterm=%-3d "), pterm);
                        (void) printf(gettext("exit=%d  "), pexit);
                }
#endif  /* XPG4 */


        /*
         *      Handle RUN_LVL process - If no alt. file - Only one!
         */
        if (utmpp->ut_type == RUN_LVL) {
                (void) printf("     %c  %5ld  %c", pterm, utmpp->ut_pid,
                    pexit);
                if (optcnt == 1 && !validtype[USER_PROCESS]) {
                        (void) printf("\n");
                        exit(0);
                }
        }

        /*
         *      Handle BOOT_TIME process -  If no alt. file - Only one!
         */
        if (utmpp->ut_type == BOOT_TIME) {
                if (optcnt == 1 && !validtype[USER_PROCESS]) {
                        (void) printf("\n");
                        exit(0);
                }
        }

        /*
         *      Get remote host from utmpx structure
         */
        if (utmpp && utmpp->ut_host[0])
                (void) printf("\t(%.*s)", sizeof (utmpp->ut_host),
                    utmpp->ut_host);
```

```
        /*
         *      Now, put on the trailing EOL
         */
        (void) printf("\n");
}

static void
process()
{
        struct passwd *pwp;
        int i = 0;
        char *ttname;

        /*
         *      Loop over each entry in /var/adm/utmpx
         */

        setutxent();
        while ((utmpp = getutxent()) != NULL) {
#ifdef DEBUG
        (void) printf(
            "ut_user ’%s’\nut_id ’%s’\nut_line ’%s’\nut_type ’%d’\n\n",
            utmpp->ut_user, utmpp->ut_id, utmpp->ut_line, utmpp->ut_type);
#endif
                if (utmpp->ut_type <= UTMAXTYPE) {
                        /*
                         *      Handle "am i"
                         */
                        if (justme) {
                                if (strncmp(myname, utmpp->ut_user,
                                    sizeof (utmpp->ut_user)) == 0 &&
                                    strncmp(mytty, utmpp->ut_line,
                                    sizeof (utmpp->ut_line)) == 0 &&
                                    utmpp->ut_type == USER_PROCESS) {
                                        /*
                                         * we have have found ourselves
                                         * in the utmp file and the entry
                                         * is a user process, this is not
                                         * meaningful otherwise
                                         *
                                         */

                                        dump();
                                        exit(0);
                                }
                                continue;
                        }

                        /*
                         *      Print the line if we want it
                         */
                        if (validtype[utmpp->ut_type]) {
#ifdef  XPG4
                                if (utmpp->ut_type == LOGIN_PROCESS) {
                                        if ((utmpp->ut_line[0] == ’\0’) ||
                                            (strcmp(utmpp->ut_user,
                                            "LOGIN") != 0))
                                        (strcmp(utmpp->ut_user, "LOGIN") != 0))
                                                continue;
                                }
#endif  /* XPG4 */
                                dump();
                        }
                } else {
                        (void) fprintf(stderr,
			    gettext("%s: Error --- entry has ut_type "
			    "of %d\n"), program, utmpp->ut_type);
			(void) fprintf(stderr,
			    gettext(" when maximum is %d\n"), UTMAXTYPE);
		}
	}

	/*
	 * If justme is set at this point than the utmp entry
	 * was not found.
	 */
	if (justme) {
		static struct utmpx utmpt;

		pwp = getpwuid(geteuid());
		if (pwp != NULL)
			while (i < (int)sizeof (utmpt.ut_user) &&
			    *pwp->pw_name != 0)
				utmpt.ut_user[i++] = *pwp->pw_name++;

		ttname = ttyname(1);

		i = 0;
		if (ttname != NULL)
			while (i < (int)sizeof (utmpt.ut_line) &&
			    *ttname != 0)
				utmpt.ut_line[i++] = *ttname++;

		utmpt.ut_id[0] = 0;
		utmpt.ut_pid = getpid();
		utmpt.ut_type = USER_PROCESS;
		(void) time(&utmpt.ut_xtime);
		utmpp = &utmpt;
		dump();
		exit(0);
	}
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   21012 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/whodo/whodo.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * 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   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

/*
 * This is the new whodo command which takes advantage of
 * the /proc interface to gain access to the information
 * of all the processes currently on the system.
 *
 * Maintenance note:
 *
 * Much of this code is replicated in w.c.  If you're
 * fixing bugs here, then you should probably fix 'em there too.
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <sys/types.h>
#include <utmpx.h>
```

```
#include <sys/utsname.h>
#include <sys/stat.h>
#include <sys/mkdev.h>
#include <dirent.h>
#include <procfs.h>             /* /proc header file */
#include <sys/wait.h>
#include <locale.h>
#include <unistd.h>
#include <limits.h>
#include <priv_utils.h>

/*
 * Use the full lengths from utmpx for user and line.
 * utmpx defines wider fields for user and line.  For compatibility of output,
 * we are limiting these to the old maximums in utmp. Define UTMPX_NAMELEN
 * to use the full lengths.
 */
#define NMAX    (sizeof (((struct utmpx *)0)->ut_user))
#define LMAX    (sizeof (((struct utmpx *)0)->ut_line))
#ifndef UTMPX_NAMELEN
/* XXX - utmp - fix name length */
#define NMAX            (_POSIX_LOGIN_NAME_MAX - 1)
#define LMAX            12
#else /* UTMPX_NAMELEN */
static struct utmpx dummy;
#define NMAX    (sizeof (dummy.ut_user))
#define LMAX    (sizeof (dummy.ut_line))
#endif /* UTMPX_NAMELEN */

/* Print minimum field widths. */
#define LOGIN_WIDTH     8
#define LINE_WIDTH      12

#define DIV60(t)        ((t+30)/60)    /* x/60 rounded */

#ifdef ERR
#undef ERR
#endif
#define ERR             (-1)

#define DEVNAMELEN      14
#define HSIZE           256             /* size of process hash table */
#define PROCDIR         "/proc"
#define INITPROCESS     (pid_t)1        /* init process pid */
#define NONE            'n'             /* no state */
#define RUNNING         'r'             /* runnable process */
#define ZOMBIE          'z'             /* zombie process */
#define VISITED         'v'             /* marked node as visited */

static int      ndevs;                  /* number of configured devices */
static int      maxdev;                 /* slots for configured devices */
#define DNINCR  100
static struct devl {                    /* device list   */
        char    dname[DEVNAMELEN];      /* device name   */
        dev_t   ddev;                   /* device number */
} *devl;
______unchanged_portion_omitted_

/*
 *      define  hash table for struct uproc
 *      Hash function uses process id
 *      and the size of the hash table(HSIZE)
 *      to determine process index into the table.
 */
static struct uproc     pr_htbl[HSIZE];
static struct   uproc   *findhash(pid_t);
static time_t   findidle(char *);
static void     clnarglist(char *);
static void     showproc(struct uproc *);
static void     showtotals(struct uproc *);
static void     calctotals(struct uproc *);
static char     *getty(dev_t);
static void     prttime(time_t, char *);
static void     prtat(time_t *);
static void     checkampm(char *);

static char     *prog;
static int      header = 1;     /* true if -h flag: don't print heading */
static int      lflag = 0;      /* true if -l flag: w command format */
static char     *sel_user;      /* login of particular user selected */
static time_t   now;            /* current time of day */
static time_t   uptime;         /* time of last reboot & elapsed time since */
static int      nusers;         /* number of users logged in now */
static time_t   idle;           /* number of minutes user is idle */
static time_t   jobtime;        /* total cpu time visible */
static char     doing[520];     /* process attached to terminal */
static time_t   proctime;       /* cpu time of process in doing */
static int      empty;
static pid_t    curpid;

#if SIGQUIT > SIGINT
#define ACTSIZE SIGQUIT
#else
#define ACTSIZE SIGINT
#endif

int
main(int argc, char *argv[])
{
        struct utmpx    *ut;
        struct utmpx    *utmpbegin;
        struct utmpx    *utmpend;
        struct utmpx    *utp;
        struct tm               *tm;
        struct uproc    *up, *parent, *pgrp;
        struct psinfo   info;
        struct sigaction actinfo[ACTSIZE];
        struct pstatus  statinfo;
        size_t          size;
        struct stat     sbuf;
        struct utsname  uts;
        DIR             *dirp;
        struct  dirent  *dp;
        char            pname[64];
        char            *fname;
        int             procfd;
        int             i;
        int             days, hrs, mins;
        int             entries;

        /*
         * This program needs the proc_owner privilege
         */
        (void) __init_suid_priv(PU_CLEARLIMITSET, PRIV_PROC_OWNER,
            (char *)NULL);

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)
#define TEXT_DOMAIN "SYS_TEST"
#endif
        (void) textdomain(TEXT_DOMAIN);
```

```
        prog = argv[0];

        while (argc > 1) {
                if (argv[1][0] == '-') {
                        for (i = 1; argv[1][i]; i++) {
                                switch (argv[1][i]) {

                                case 'h':
                                        header = 0;
                                        break;

                                case 'l':
                                        lflag++;
                                        break;

                                default:
                                        (void) printf(gettext(
                                            "usage: %s [ -hl ] [ user ]\n"),
                                            prog);
                                        exit(1);
                                }
                        }
                } else {
                        if (!isalnum(argv[1][0]) || argc > 2) {
                                (void) printf(gettext(
                                    "usage: %s [ -hl ] [ user ]\n"), prog);
                                exit(1);
                        } else
                                sel_user = argv[1];
                }
                argc--; argv++;
        }

        /*
         * read the UTMPX_FILE (contains information about
         * each logged in user)
         */
        if (stat(UTMPX_FILE, &sbuf) == ERR) {
                (void) fprintf(stderr, gettext("%s: stat error of %s: %s\n"),
                    prog, UTMPX_FILE, strerror(errno));
                exit(1);
        }
        entries = sbuf.st_size / sizeof (struct futmpx);
        size = sizeof (struct utmpx) * entries;

        if ((ut = malloc(size)) == NULL) {
                (void) fprintf(stderr, gettext("%s: malloc error of %s: %s\n"),
                    prog, UTMPX_FILE, strerror(errno));
                exit(1);
        }

        (void) utmpxname(UTMPX_FILE);

        utmpbegin = ut;
        /* LINTED pointer cast may result in improper alignment */
        utmpend = (struct utmpx *)((char *)utmpbegin + size);

        setutxent();
        while ((ut < utmpend) && ((utp = getutxent()) != NULL))
                (void) memcpy(ut++, utp, sizeof (*ut));
        endutxent();

        (void) time(&now);      /* get current time */

        if (header) {   /* print a header */
```

```
                if (lflag) {    /* w command format header */
                        prtat(&now);
                        for (ut = utmpbegin; ut < utmpend; ut++) {
                                if (ut->ut_type == USER_PROCESS) {
                                        nusers++;
                                } else if (ut->ut_type == BOOT_TIME) {
                                        uptime = now - ut->ut_xtime;
                                        uptime += 30;
                                        days = uptime / (60*60*24);
                                        uptime %= (60*60*24);
                                        hrs = uptime / (60*60);
                                        uptime %= (60*60);
                                        mins = uptime / 60;

                                        (void) printf(dcgettext(NULL,
                                            "  up %d day(s), %d hr(s), "
                                            "%d min(s)", LC_TIME),
                                            days, hrs, mins);
                                }
                        }

                        ut = utmpbegin; /* rewind utmp data */
                        (void) printf(dcgettext(NULL,
                            "  %d user(s)\n", LC_TIME), nusers);
                        (void) printf(dcgettext(NULL, "User     tty           "
                            "login@  idle   JCPU   PCPU  what\n", LC_TIME));
                } else {        /* standard whodo header */
                        char date_buf[100];

                        /*
                         * print current time and date
                         */
                        (void) strftime(date_buf, sizeof (date_buf),
                            dcgettext(NULL, "%C", LC_TIME), localtime(&now));
                        (void) printf("%s\n", date_buf);

                        /*
                         * print system name
                         */
                        (void) uname(&uts);
                        (void) printf("%s\n", uts.nodename);
                }
        }

        /*
         * loop through /proc, reading info about each process
         * and build the parent/child tree
         */
        if (!(dirp = opendir(PROCDIR))) {
                (void) fprintf(stderr, gettext("%s: could not open %s: %s\n"),
                    prog, PROCDIR, strerror(errno));
                exit(1);
        }

        while ((dp = readdir(dirp)) != NULL) {
                if (dp->d_name[0] == '.')
                        continue;
retry:
                (void) snprintf(pname, sizeof (pname),
                    "%s/%s/", PROCDIR, dp->d_name);
                fname = pname + strlen(pname);
                (void) strcpy(fname, "psinfo");
                if ((procfd = open(pname, O_RDONLY)) < 0)
                        continue;
                if (read(procfd, &info, sizeof (info)) != sizeof (info)) {
                        int err = errno;
```

```
                        (void) close(procfd);
                        if (err == EAGAIN)
                                goto retry;
                        if (err != ENOENT)
                                (void) fprintf(stderr, gettext(
                                    "%s: read() failed on %s: %s\n"),
                                    prog, pname, strerror(err));
                        continue;
                }
                (void) close(procfd);

                up = findhash(info.pr_pid);
                up->p_ttyd = info.pr_ttydev;
                up->p_state = (info.pr_nlwp == 0? ZOMBIE : RUNNING);
                up->p_time = 0;
                up->p_ctime = 0;
                up->p_igintr = 0;
                (void) strncpy(up->p_comm, info.pr_fname,
                    sizeof (info.pr_fname));
                up->p_args[0] = 0;

                if (up->p_state != NONE && up->p_state != ZOMBIE) {
                        (void) strcpy(fname, "status");

                        /* now we need the proc_owner privilege */
                        (void) __priv_bracket(PRIV_ON);

                        procfd = open(pname, O_RDONLY);

                        /* drop proc_owner privilege after open */
                        (void) __priv_bracket(PRIV_OFF);

                        if (procfd  < 0)
                                continue;

                        if (read(procfd, &statinfo, sizeof (statinfo))
                            != sizeof (statinfo)) {
                                int err = errno;
                                (void) close(procfd);
                                if (err == EAGAIN)
                                        goto retry;
                                if (err != ENOENT)
                                        (void) fprintf(stderr, gettext(
                                            "%s: read() failed on %s: %s \n"),
                                            prog, pname, strerror(err));
                                continue;
                        }
                        (void) close(procfd);

                        up->p_time = statinfo.pr_utime.tv_sec +
                            statinfo.pr_stime.tv_sec;
                        up->p_ctime = statinfo.pr_cutime.tv_sec +
                            statinfo.pr_cstime.tv_sec;

                        (void) strcpy(fname, "sigact");

                        /* now we need the proc_owner privilege */
                        (void) __priv_bracket(PRIV_ON);

                        procfd = open(pname, O_RDONLY);

                        /* drop proc_owner privilege after open */
                        (void) __priv_bracket(PRIV_OFF);

                        if (procfd  < 0)
                                continue;
                        if (read(procfd, actinfo, sizeof (actinfo))
                            != sizeof (actinfo)) {
                                int err = errno;
                                (void) close(procfd);
                                if (err == EAGAIN)
                                        goto retry;
                                if (err != ENOENT)
                                        (void) fprintf(stderr, gettext(
                                            "%s: read() failed on %s: %s \n"),
                                            prog, pname, strerror(err));
                                continue;
                        }
                        (void) close(procfd);

                        up->p_igintr =
                            actinfo[SIGINT-1].sa_handler == SIG_IGN &&
                            actinfo[SIGQUIT-1].sa_handler == SIG_IGN;

                        up->p_args[0] = 0;

                        /*
                         * Process args if there’s a chance we’ll print it.
                         */
                        if (lflag) { /* w command needs args */
                                clnarglist(info.pr_psargs);
                                (void) strcpy(up->p_args, info.pr_psargs);
                                if (up->p_args[0] == 0 ||
                                    up->p_args[0] == ’-’ &&
                                    up->p_args[1] <= ’ ’ ||
                                    up->p_args[0] == ’?’) {
                                        (void) strcat(up->p_args, " (");
                                        (void) strcat(up->p_args, up->p_comm);
                                        (void) strcat(up->p_args, ")");
                                }
                        }

                }

                /*
                 * link pgrp together in case parents go away
                 * Pgrp chain is a single linked list originating
                 * from the pgrp leader to its group member.
                 */
                if (info.pr_pgid != info.pr_pid) {      /* not pgrp leader */
                        pgrp = findhash(info.pr_pgid);
                        up->p_pgrplink = pgrp->p_pgrplink;
                        pgrp->p_pgrplink = up;
                }
                parent = findhash(info.pr_ppid);

                /* if this is the new member, link it in */
                if (parent->p_upid != INITPROCESS) {
                        if (parent->p_child) {
                                up->p_sibling = parent->p_child;
                                up->p_child = 0;
                        }
                        parent->p_child = up;
                }

        }

        /* revert to non-privileged user */
        (void) __priv_relinquish();

        (void) closedir(dirp);
        (void) time(&now);      /* get current time */
```

```

        /*
         * loop through utmpx file, printing process info
         * about each logged in user
         */
        for (ut = utmpbegin; ut < utmpend; ut++) {
                time_t tim;

                if (ut->ut_type != USER_PROCESS)
                        continue;
                if (sel_user && strncmp(ut->ut_name, sel_user, NMAX) != 0)
                        continue;       /* we’re looking for somebody else */
                if (lflag) {    /* -l flag format (w command) */
                        /* print login name of the user */
                        (void) printf("%-*.*s ", LOGIN_WIDTH, (int)NMAX,
                            ut->ut_name);
                        (void) printf("%-*.*s ", NMAX, NMAX, ut->ut_name);

                        /* print tty user is on */
                        (void) printf("%-*.*s", LINE_WIDTH, (int)LMAX,
                            ut->ut_line);
                        (void) printf("%-*.*s", LMAX, LMAX, ut->ut_line);

                        /* print when the user logged in */
                        tim = ut->ut_xtime;
                        (void) prtat(&tim);

                        /* print idle time */
                        idle = findidle(ut->ut_line);
                        if (idle >= 36 * 60)
                                (void) printf(dcgettext(NULL, "%2ddays ",
                                    LC_TIME), (idle + 12 * 60) / (24 * 60));
                        else
                                prttime(idle, " ");
                        showtotals(findhash((pid_t)ut->ut_pid));
                } else {        /* standard whodo format */
                        tim = ut->ut_xtime;
                        tm = localtime(&tim);
                        (void) printf("\n%-*.*s %-*.*s %2.1d:%2.2d\n",
                            LINE_WIDTH, (int)LMAX, ut->ut_line,
                            LOGIN_WIDTH, (int)NMAX, ut->ut_name, tm->tm_hour,
                            tm->tm_min);
                            LMAX, LMAX, ut->ut_line,
                            NMAX, NMAX, ut->ut_name, tm->tm_hour, tm->tm_min);
                        showproc(findhash((pid_t)ut->ut_pid));
                }
        }

        return (0);
}

/*
 * Used for standard whodo format.
 * This is the recursive routine descending the process
 * tree starting from the given process pointer(up).
 * It used depth-first search strategy and also marked
 * each node as printed as it traversed down the tree.
 */
static void
showproc(struct uproc *up)
{
        struct  uproc   *zp;

        if (up->p_state == VISITED) /* we already been here */
                return;
        /* print the data for this process */
        if (up->p_state == ZOMBIE)
                (void) printf("    %-*.*s %5d %4.1ld:%2.2ld %s\n",
                    LINE_WIDTH, (int)LMAX, "  ?", (int)up->p_upid, 0L, 0L,
                    "<defunct>");
                    LMAX, LMAX, "  ?", (int)up->p_upid, 0L, 0L, "<defunct>");
        else if (up->p_state != NONE) {
                (void) printf("    %-*.*s %5d %4.1ld:%2.2ld %s\n",
                    LINE_WIDTH, (int)LMAX, getty(up->p_ttyd), (int)up->p_upid,
                    LMAX, LMAX, getty(up->p_ttyd), (int)up->p_upid,
                    up->p_time / 60L, up->p_time % 60L,
                    up->p_comm);
        }
        up->p_state = VISITED;

        /* descend for its children */
        if (up->p_child) {
                showproc(up->p_child);
                for (zp = up->p_child->p_sibling; zp; zp = zp->p_sibling) {
                        showproc(zp);
                }
        }

        /* print the pgrp relation */
        if (up->p_pgrplink)
                showproc(up->p_pgrplink);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   57228 Wed Apr  3 09:33:12 2013
new/usr/src/cmd/zlogin/zlogin.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
```

```
/*
 * 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) 2013 Gary Mills
 *
 * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * zlogin provides three types of login which allow users in the global
 * zone to access non-global zones.
 *
 * - "interactive login" is similar to rlogin(1); for example, the user could
 *   issue 'zlogin my-zone' or 'zlogin -e ^ -l me my-zone'.   The user is
 *   granted a new pty (which is then shoved into the zone), and an I/O
 *   loop between parent and child processes takes care of the interactive
 *   session.  In this mode, login(1) (and its -c option, which means
 *   "already authenticated") is employed to take care of the initialization
 *   of the user's session.
 *
 * - "non-interactive login" is similar to su(1M); the user could issue
 *   'zlogin my-zone ls -l' and the command would be run as specified.
 *   In this mode, zlogin sets up pipes as the communication channel, and
 *   'su' is used to do the login setup work.
 *
 * - "console login" is the equivalent to accessing the tip line for a
 *   zone.  For example, the user can issue 'zlogin -C my-zone'.
 *   In this mode, zlogin contacts the zoneadmd process via unix domain
 *   socket.  If zoneadmd is not running, it starts it.  This allows the
 *   console to be available anytime the zone is installed, regardless of
 *   whether it is running.
 */

#include <sys/socket.h>
#include <sys/termios.h>
#include <sys/utsname.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/contract/process.h>
#include <sys/ctfs.h>
#include <sys/brand.h>
#include <sys/wait.h>
```

```
#include <alloca.h>
#include <assert.h>
#include <ctype.h>
#include <door.h>
#include <errno.h>
#include <nss_dbdefs.h>
#include <poll.h>
#include <priv.h>
#include <pwd.h>
#include <unistd.h>
#include <utmpx.h>
#include <sac.h>
#include <signal.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <stropts.h>
#include <wait.h>
#include <zone.h>
#include <fcntl.h>
#include <libdevinfo.h>
#include <libintl.h>
#include <locale.h>
#include <libzonecfg.h>
#include <libcontract.h>
#include <libbrand.h>
#include <auth_list.h>
#include <auth_attr.h>
#include <secdb.h>

static int masterfd;
static struct termios save_termios;
static struct termios effective_termios;
static int save_fd;
static struct winsize winsize;
static volatile int dead;
static volatile pid_t child_pid = -1;
static int interactive = 0;
static priv_set_t *dropprivs;

static int nocmdchar = 0;
static int failsafe = 0;
static char cmdchar = '~';

static int pollerr = 0;

static const char *pname;
static char *username;

/*
 * When forced_login is true, the user is not prompted
 * for an authentication password in the target zone.
 */
static boolean_t forced_login = B_FALSE;

#if !defined(TEXT_DOMAIN)               /* should be defined by cc -D */
#define TEXT_DOMAIN     "SYS_TEST"      /* Use this only if it wasn't */
#endif

#define SUPATH  "/usr/bin/su"
#define FAILSAFESHELL   "/sbin/sh"
#define DEFAULTSHELL    "/sbin/sh"
#define DEF_PATH        "/usr/sbin:/usr/bin"
```

```
#define CLUSTER_BRAND_NAME      "cluster"

/*
 * The ZLOGIN_BUFSIZ is larger than PIPE_BUF so we can be sure we're clearing
 * out the pipe when the child is exiting.  The ZLOGIN_RDBUFSIZ must be less
 * than ZLOGIN_BUFSIZ (because we share the buffer in doio).  This value is
 * also chosen in conjunction with the HI_WATER setting to make sure we
 * don't fill up the pipe.  We can write FIFOHIWAT (16k) into the pipe before
 * blocking.  By having ZLOGIN_RDBUFSIZ set to 1k and HI_WATER set to 8k, we
 * know we can always write a ZLOGIN_RDBUFSIZ chunk into the pipe when there
 * is less than HI_WATER data already in the pipe.
 */
#define ZLOGIN_BUFSIZ   8192
#define ZLOGIN_RDBUFSIZ 1024
#define HI_WATER        8192

/*
 * See canonify() below.  CANONIFY_LEN is the maximum length that a
 * "canonical" sequence will expand to (backslash, three octal digits, NUL).
 */
#define CANONIFY_LEN 5

static void
usage(void)
{
        (void) fprintf(stderr, gettext("usage: %s [ -CES ] [ -e cmdchar ] "
            "[-l user] zonename [command [args ...] ]\n"), pname);
        exit(2);
}
```

______***unchanged_portion_omitted_***

```
/*
 * Finish the preparation of the envp array for exec'd non-interactive
 * zlogins.  This is called in the child process *after* we zone_enter(), since
 * it derives things we can only know within the zone, such as $HOME, $SHELL,
 * etc.  We need only do this in the non-interactive, mode, since otherwise
 * login(1) will do it.  We don't do this in failsafe mode, since it presents
 * additional ways in which the command could fail, and we'd prefer to avoid
 * that.
 */
static char **
prep_env_noninteractive(const char *user_cmd, char **env)
{
        size_t size;
        char **new_env;
        int e, i;
        char *estr;
        char varmail[LOGNAME_MAX_ILLUMOS + 11]; /* strlen(/var/mail/) = */
                                                /* 10, NUL */
        char varmail[LOGNAME_MAX + 11]; /* strlen(/var/mail/) = 10, NUL */
        char pwbuf[NSS_BUFLEN_PASSWD + 1];
        struct passwd pwent;
        struct passwd *pw = NULL;

        assert(env != NULL);
        assert(failsafe == 0);

        /*
         * Exec the "user_cmd" brand hook to get a pwent for the
         * login user.  If this fails, HOME will be set to "/", SHELL
         * will be set to $DEFAULTSHELL, and we will continue to exec
         * SUPATH <login> -c <cmd>.
         */
        pw = zone_get_user_pw(user_cmd, &pwent, pwbuf, sizeof (pwbuf));

        /*
```

```
         * Get existing envp size.
         */
        for (size = 0; env[size] != NULL; size++)
                ;

        e = size;

        /*
         * Finish filling out the environment; we duplicate the environment
         * setup described in login(1), for lack of a better precedent.
         */
        if (pw != NULL)
                size += 3;      /* LOGNAME, HOME, MAIL */
        else
                size += 1;      /* HOME */

        size++; /* always fill in SHELL */
        size++; /* terminating NULL */

        if ((new_env = malloc(sizeof (char *) * size)) == NULL)
                goto malloc_fail;

        /*
         * Copy existing elements of env into new_env.
         */
        for (i = 0; env[i] != NULL; i++) {
                if ((new_env[i] = strdup(env[i])) == NULL)
                        goto malloc_fail;
        }
        assert(e == i);

        if (pw != NULL) {
                if ((estr = add_env("LOGNAME", pw->pw_name)) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;

                if ((estr = add_env("HOME", pw->pw_dir)) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;

                if (chdir(pw->pw_dir) != 0)
                        zerror(gettext("Could not chdir to home directory "
                            "%s: %s"), pw->pw_dir, strerror(errno));

                (void) snprintf(varmail, sizeof (varmail), "/var/mail/%s",
                    pw->pw_name);
                if ((estr = add_env("MAIL", varmail)) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;
        } else {
                if ((estr = add_env("HOME", "/")) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;
        }

        if (pw != NULL && strlen(pw->pw_shell) > 0) {
                if ((estr = add_env("SHELL", pw->pw_shell)) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;
        } else {
                if ((estr = add_env("SHELL", DEFAULTSHELL)) == NULL)
                        goto malloc_fail;
                new_env[e++] = estr;
        }

        new_env[e++] = NULL;    /* add terminating NULL */
        assert(e == size);
        return (new_env);

malloc_fail:
        zperror(gettext("failed to allocate memory for process environment"));
        return (NULL);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   10568 Wed Apr  3 09:33:12 2013
new/usr/src/head/limits.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
```

```
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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


#ifndef _LIMITS_H
#define _LIMITS_H

#pragma ident   "%Z%%M% %I%     %E% SMI"        /* SVr4.0 1.34  */

#include <sys/feature_tests.h>
#include <sys/isa_defs.h>
#include <iso/limits_iso.h>

/*
 * Include fixed width type limits as proposed by the ISO/JTC1/SC22/WG14 C
 * committee's working draft for the revision of the current ISO C standard,
 * ISO/IEC 9899:1990 Programming language - C.  These are not currently
 * required by any standard but constitute a useful, general purpose set
 * of type definitions and limits which is namespace clean with respect to
 * all standards.
 */
#if defined(__EXTENSIONS__) || !defined(_STRICT_STDC) || \
        defined(__XOPEN_OR_POSIX)
#include <sys/int_limits.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#if defined(__EXTENSIONS__) || !defined(_STRICT_STDC) || \
        defined(__XOPEN_OR_POSIX)
```

```
#define SSIZE_MAX       LONG_MAX        /* max value of an "ssize_t" */

/*
 * ARG_MAX is calculated as follows:
 * NCARGS - space for other stuff on initial stack
 * like aux vectors, saved registers, etc..
 */
#define _ARG_MAX32      1048320 /* max length of args to exec 32-bit program */
#define _ARG_MAX64      2096640 /* max length of args to exec 64-bit program */
#ifdef  _LP64
#define ARG_MAX         _ARG_MAX64      /* max length of arguments to exec */
#else   /* _LP64 */
#define ARG_MAX         _ARG_MAX32      /* max length of arguments to exec */
#endif  /* _LP64 */

#ifndef MAX_CANON
#define MAX_CANON       256     /* max bytes in line for canonical processing */
#endif

#ifndef MAX_INPUT
#define MAX_INPUT       512     /* max size of a char input buffer */
#endif

#define NGROUPS_MAX     16      /* max number of groups for a user */

#ifndef PATH_MAX
#define PATH_MAX        1024    /* max # of characters in a path name */
#endif

#define SYMLINK_MAX     1024    /* max # of characters a symlink can contain */

#define PIPE_BUF        5120    /* max # bytes atomic in write to a pipe */

#ifndef TMP_MAX
#define TMP_MAX         17576   /* 26 * 26 * 26 */
#endif

/*
 * POSIX conformant definitions - An implementation may define
 * other symbols which reflect the actual implementation. Alternate
 * definitions may not be as restrictive as the POSIX definitions.
 */
#define _POSIX_AIO_LISTIO_MAX       2
#define _POSIX_AIO_MAX              1
#define _POSIX_ARG_MAX           4096
#ifdef _XPG6
#define _POSIX_CHILD_MAX           25
#else
#define _POSIX_CHILD_MAX            6   /* POSIX.1-1990 default */
#endif
#define _POSIX_CLOCKRES_MIN     20000000
#define _POSIX_DELAYTIMER_MAX      32
#define _POSIX_LINK_MAX             8
#define _POSIX_MAX_CANON          255
#define _POSIX_MAX_INPUT          255
#define _POSIX_MQ_OPEN_MAX          8
#define _POSIX_MQ_PRIO_MAX         32
#define _POSIX_NAME_MAX            14
#ifdef _XPG6
#define _POSIX_NGROUPS_MAX          8
#define _POSIX_OPEN_MAX            20
#define _POSIX_PATH_MAX           256
#else                                   /* POSIX.1-1990 defaults */
#define _POSIX_NGROUPS_MAX          0
#define _POSIX_OPEN_MAX            16
```

```
#define _POSIX_PATH_MAX           255
#endif
#define _POSIX_PIPE_BUF           512
#define _POSIX_RTSIG_MAX            8
#define _POSIX_SEM_NSEMS_MAX      256
#define _POSIX_SEM_VALUE_MAX    32767
#define _POSIX_SIGQUEUE_MAX        32
#define _POSIX_SSIZE_MAX        32767
#define _POSIX_STREAM_MAX           8
#define _POSIX_TIMER_MAX           32
#ifdef _XPG6
#define _POSIX_TZNAME_MAX           6
#else
#define _POSIX_TZNAME_MAX           3   /* POSIX.1-1990 default */
#endif
/* POSIX.1c conformant */
#define _POSIX_LOGIN_NAME_MAX                   9
#define _POSIX_THREAD_DESTRUCTOR_ITERATIONS     4
#define _POSIX_THREAD_KEYS_MAX                  128
#define _POSIX_THREAD_THREADS_MAX               64
#define _POSIX_TTY_NAME_MAX                     9
/* UNIX 03 conformant */
#define _POSIX_HOST_NAME_MAX                    255
#define _POSIX_RE_DUP_MAX                       255
#define _POSIX_SYMLINK_MAX                      255
#define _POSIX_SYMLOOP_MAX                      8

/*
 * POSIX.2 and XPG4-XSH4 conformant definitions
 */

#define _POSIX2_BC_BASE_MAX               99
#define _POSIX2_BC_DIM_MAX              2048
#define _POSIX2_BC_SCALE_MAX              99
#define _POSIX2_BC_STRING_MAX           1000
#define _POSIX2_COLL_WEIGHTS_MAX           2
#define _POSIX2_EXPR_NEST_MAX             32
#define _POSIX2_LINE_MAX                2048
#define _POSIX2_RE_DUP_MAX               255
/* UNIX 03 conformant */
#define _POSIX2_CHARCLASS_NAME_MAX        14

#define BC_BASE_MAX             _POSIX2_BC_BASE_MAX
#define BC_DIM_MAX              _POSIX2_BC_DIM_MAX
#define BC_SCALE_MAX            _POSIX2_BC_SCALE_MAX
#define BC_STRING_MAX           _POSIX2_BC_STRING_MAX
#define COLL_WEIGHTS_MAX        10
#define EXPR_NEST_MAX           _POSIX2_EXPR_NEST_MAX
#define LINE_MAX                _POSIX2_LINE_MAX
#if !defined(_XPG6)
#define RE_DUP_MAX              _POSIX2_RE_DUP_MAX
#else
#define RE_DUP_MAX              _POSIX_RE_DUP_MAX
#endif /* !defined(_XPG6) */

#endif /* defined(__EXTENSIONS__) || !defined(_STRICT_STDC) ... */

#if defined(__EXTENSIONS__) || \
        (!defined(_STRICT_STDC) && !defined(_POSIX_C_SOURCE)) || \
        defined(_XOPEN_SOURCE)

/*
 * For dual definitions for PASS_MAX and sysconf.c
 */
#define _PASS_MAX_XPG   8       /* old standards PASS_MAX */
#define _PASS_MAX       256     /* modern Solaris PASS_MAX */
```

```

#if defined(_XPG3) && !defined(_XPG6)
#define PASS_MAX        _PASS_MAX_XPG   /* max # of characters in a password */
#else   /* XPG6 or just Solaris */
#define PASS_MAX        _PASS_MAX       /* max # of characters in a password */
#endif  /* defined(_XPG3) && !defined(_XPG6) */

#define CHARCLASS_NAME_MAX      _POSIX2_CHARCLASS_NAME_MAX

#define NL_ARGMAX       9       /* max value of "digit" in calls to the */
                                /* NLS printf() and scanf() */
#define NL_LANGMAX      14      /* max # of bytes in a LANG name */
#define NL_MSGMAX       32767   /* max message number */
#define NL_NMAX         1       /* max # bytes in N-to-1 mapping characters */
#define NL_SETMAX       255     /* max set number */
#define NL_TEXTMAX      2048    /* max set number */
#define NZERO           20      /* default process priority */

#define WORD_BIT        32      /* # of bits in a "word" or "int" */
#if defined(_LP64)
#define LONG_BIT        64      /* # of bits in a "long" */
#else   /* _ILP32 */
#define LONG_BIT        32      /* # of bits in a "long" */
#endif

/* Marked as LEGACY in SUSv2 and removed in UNIX 03 */
#ifndef _XPG6
#define DBL_DIG         15      /* digits of precision of a "double" */
#define DBL_MAX         1.7976931348623157081452E+308   /* max decimal value */
                                                        /* of a double */
#define FLT_DIG         6               /* digits of precision of a "float" */
#define FLT_MAX         3.4028234663852885981170E+38F   /* max decimal value */
                                                        /* of a "float" */
#endif

/* Marked as LEGACY in SUSv1 and removed in SUSv2 */
#ifndef _XPG5
#define DBL_MIN         2.2250738585072013830903E-308   /* min decimal value */
                                                        /* of a double */
#define FLT_MIN         1.1754943508222875079688E-38F   /* min decimal value */
                                                        /* of a float */
#endif

#endif  /* defined(__EXTENSIONS__) || (!defined(_STRICT_STDC) ... */

#define _XOPEN_IOV_MAX  16      /* max # iovec/process with readv()/writev() */
#define _XOPEN_NAME_MAX 255     /* max # bytes in filename excluding null */
#define _XOPEN_PATH_MAX 1024    /* max # bytes in a pathname */

#define IOV_MAX         _XOPEN_IOV_MAX

#if defined(__EXTENSIONS__) || \
        (!defined(_STRICT_STDC) && !defined(__XOPEN_OR_POSIX))

#define FCHR_MAX        1048576         /* max size of a file in bytes */
#define PID_MAX         999999          /* max value for a process ID */

/*
 * POSIX 1003.1a, section 2.9.5, table 2-5 contains [NAME_MAX] and the
 * related text states:
 *
 * A definition of one of the values from Table 2-5 shall be omitted from the
 * <limits.h> on specific implementations where the corresponding value is
 * equal to or greater than the stated minimum, but where the value can vary
 * depending on the file to which it is applied. The actual value supported for
 * a specific pathname shall be provided by the pathconf() (5.7.1) function.
 *
 * This is clear that any machine supporting multiple file system types
 * and/or a network can not include this define, regardless of protection
 * by the _POSIX_SOURCE and _POSIX_C_SOURCE flags.
 *
 * #define      NAME_MAX        14
 */

#define CHILD_MAX       25      /* max # of processes per user id */
#ifndef OPEN_MAX
#define OPEN_MAX        256     /* max # of files a process can have open */
#endif

#define PIPE_MAX        5120    /* max # bytes written to a pipe in a write */

#define STD_BLK         1024    /* # bytes in a physical I/O block */
#define UID_MAX         2147483647      /* max value for a user or group ID */
#define USI_MAX         4294967295      /* max decimal value of an "unsigned" */
#define SYSPID_MAX      1       /* max pid of system processes */

#ifndef SYS_NMLN                /* also defined in sys/utsname.h */
#define SYS_NMLN        257     /* 4.0 size of utsname elements */
#endif

#ifndef CLK_TCK

#if !defined(_CLOCK_T) || __cplusplus >= 199711L
#define _CLOCK_T
typedef long    clock_t;
#endif  /* !_CLOCK_T */

extern long _sysconf(int);      /* System Private interface to sysconf() */
#define CLK_TCK ((clock_t)_sysconf(3))  /* 3 is _SC_CLK_TCK */

#endif /* CLK_TCK */

#define LOGNAME_MAX     8       /* max # of characters in a login name */
#define LOGNAME_MAX_ILLUMOS     32      /* max # of characters in an */
                                        /* illumos login name */
#define LOGIN_NAME_MAX  (LOGNAME_MAX_ILLUMOS + 1)       /* max buffer size */
#define TTYNAME_MAX     128     /* max # of characters in a tty name */

#endif  /* if defined(__EXTENSIONS__) || (!defined(_STRICT_STDC) ... */

#if     defined(__EXTENSIONS__) || (_POSIX_C_SOURCE >= 199506L)
#include <sys/unistd.h>

#if !defined(_SIZE_T) || __cplusplus >= 199711L
#define _SIZE_T
#if defined(_LP64) || defined(_I32LPx)
typedef unsigned long size_t;   /* size of something in bytes */
#else
typedef unsigned int  size_t;   /* (historical version) */
#endif
#endif  /* _SIZE_T */

extern long _sysconf(int);      /* System Private interface to sysconf() */

#define PTHREAD_STACK_MIN       ((size_t)_sysconf(_SC_THREAD_STACK_MIN))
/* Added for UNIX98 conformance */
#define PTHREAD_DESTRUCTOR_ITERATIONS   _POSIX_THREAD_DESTRUCTOR_ITERATIONS
#define PTHREAD_KEYS_MAX                _POSIX_THREAD_KEYS_MAX
#define PTHREAD_THREADS_MAX             _POSIX_THREAD_THREADS_MAX
#endif  /* defined(__EXTENSIONS__) || (_POSIX_C_SOURCE >= 199506L) */

#ifdef  __cplusplus
```

```
}
```

______***unchanged_portion_omitted_***

```
************************************************************
   29037 Wed Apr  3 09:33:12 2013
new/usr/src/head/nss_dbdefs.h
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
************************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Database-specific definitions for the getXXXbyYYY routines
 * (e.g getpwuid_r(), ether_ntohost()) that use the name-service switch.
 * Database-independent definitions are in <nss_common.h>
 *
 * Ideally, this is the only switch header file one would add things
 * to in order to support a new database.
 *
 * NOTE:  The interfaces documented in this file may change in a minor
 *        release.  It is intended that in the future a stronger committment
 *        will be made to these interface definitions which will guarantee
 *        them across minor releases.
 */

#ifndef _NSS_DBDEFS_H
#define _NSS_DBDEFS_H

#include <sys/types.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>              /* MAXALIASES, MAXADDRS */
#include <limits.h>             /* LOGNAME_MAX */
#include <nss_common.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifndef NSS_INCLUDE_UNSAFE
#define NSS_INCLUDE_UNSAFE      1       /* Build old, MT-unsafe interfaces, */
#endif  /* NSS_INCLUDE_UNSAFE */        /*  e.g. getpwnam (c.f. getpwnam_r) */

/*
 * Names of the well-known databases.
 */
```

```
#define NSS_DBNAM_ALIASES       "aliases"       /* E-mail aliases, that is */
#define NSS_DBNAM_AUTOMOUNT     "automount"
#define NSS_DBNAM_BOOTPARAMS    "bootparams"
#define NSS_DBNAM_ETHERS        "ethers"
#define NSS_DBNAM_GROUP         "group"
#define NSS_DBNAM_HOSTS         "hosts"
#define NSS_DBNAM_IPNODES       "ipnodes"
#define NSS_DBNAM_NETGROUP      "netgroup"
#define NSS_DBNAM_NETMASKS      "netmasks"
#define NSS_DBNAM_NETWORKS      "networks"
#define NSS_DBNAM_PASSWD        "passwd"
#define NSS_DBNAM_PRINTERS      "printers"
#define NSS_DBNAM_PROJECT       "project"
#define NSS_DBNAM_PROTOCOLS     "protocols"
#define NSS_DBNAM_PUBLICKEY     "publickey"
#define NSS_DBNAM_RPC           "rpc"
#define NSS_DBNAM_SERVICES      "services"
#define NSS_DBNAM_AUDITUSER     "audit_user"
#define NSS_DBNAM_AUTHATTR      "auth_attr"
#define NSS_DBNAM_EXECATTR      "exec_attr"
#define NSS_DBNAM_PROFATTR      "prof_attr"
#define NSS_DBNAM_USERATTR      "user_attr"

#define NSS_DBNAM_TSOL_TP       "tnrhtp"
#define NSS_DBNAM_TSOL_RH       "tnrhdb"
#define NSS_DBNAM_TSOL_ZC       "tnzonecfg"

/* getspnam() et al use the "passwd" config entry but the "shadow" backend */
#define NSS_DBNAM_SHADOW        "shadow"

/* The "compat" backend gets config entries for these pseudo-databases */
#define NSS_DBNAM_PASSWD_COMPAT "passwd_compat"
#define NSS_DBNAM_GROUP_COMPAT  "group_compat"

/*
 * Default switch configuration, compiled into the front-ends.
 *
 * Absent good reasons to the contrary, this should be compatible with the
 * default /etc/nsswitch.conf file.
 */
#define NSS_FILES_ONLY          "files"
#define NSS_FILES_NS            "files nis"
#define NSS_NS_FALLBACK         "nis [NOTFOUND=return] files"
#define NSS_NS_ONLY             "nis"
#define NSS_TSOL_FALLBACK       "files ldap"

#define NSS_DEFCONF_ALIASES     NSS_FILES_NS
#define NSS_DEFCONF_AUTOMOUNT   NSS_FILES_NS
#define NSS_DEFCONF_BOOTPARAMS  NSS_NS_FALLBACK
#define NSS_DEFCONF_ETHERS      NSS_NS_FALLBACK
#define NSS_DEFCONF_GROUP       NSS_FILES_NS
#define NSS_DEFCONF_HOSTS       NSS_NS_FALLBACK
#define NSS_DEFCONF_IPNODES     NSS_NS_FALLBACK
#define NSS_DEFCONF_NETGROUP    NSS_NS_ONLY
#define NSS_DEFCONF_NETMASKS    NSS_NS_FALLBACK
#define NSS_DEFCONF_NETWORKS    NSS_NS_FALLBACK
#define NSS_DEFCONF_PASSWD      NSS_FILES_NS
#define NSS_DEFCONF_PRINTERS    "user files nis"
#define NSS_DEFCONF_PROJECT     NSS_FILES_NS
#define NSS_DEFCONF_PROTOCOLS   NSS_NS_FALLBACK
#define NSS_DEFCONF_PUBLICKEY   NSS_FILES_NS
#define NSS_DEFCONF_RPC         NSS_NS_FALLBACK
#define NSS_DEFCONF_SERVICES    NSS_FILES_NS    /* speeds up byname() */

#define NSS_DEFCONF_GROUP_COMPAT        NSS_NS_ONLY
```

```
#define NSS_DEFCONF_PASSWD_COMPAT       NSS_NS_ONLY

#define NSS_DEFCONF_ATTRDB      NSS_FILES_NS

#define NSS_DEFCONF_AUDITUSER   NSS_DEFCONF_PASSWD
#define NSS_DEFCONF_USERATTR    NSS_DEFCONF_PASSWD
#define NSS_DEFCONF_AUTHATTR    NSS_DEFCONF_ATTRDB
#define NSS_DEFCONF_PROFATTR    NSS_DEFCONF_ATTRDB
#define NSS_DEFCONF_EXECATTR    NSS_DEFCONF_PROFATTR

#define NSS_DEFCONF_TSOL_TP     NSS_TSOL_FALLBACK
#define NSS_DEFCONF_TSOL_RH     NSS_TSOL_FALLBACK
#define NSS_DEFCONF_TSOL_ZC     NSS_TSOL_FALLBACK

/*
 * Line-lengths that the "files" and "compat" backends will try to support.
 * It may be reasonable (even advisable) to use smaller values than these.
 */

#define NSS_BUFSIZ              1024

#define NSS_LINELEN_GROUP       ((NSS_BUFSIZ) * 8)
#define NSS_LINELEN_HOSTS       ((NSS_BUFSIZ) * 8)
#define NSS_LINELEN_IPNODES     ((NSS_BUFSIZ) * 8)
#define NSS_LINELEN_NETMASKS    NSS_BUFSIZ
#define NSS_LINELEN_NETWORKS    NSS_BUFSIZ
#define NSS_LINELEN_PASSWD      NSS_BUFSIZ
#define NSS_LINELEN_PRINTERS    NSS_BUFSIZ
#define NSS_LINELEN_PROJECT     ((NSS_BUFSIZ) * 4)
#define NSS_LINELEN_PROTOCOLS   NSS_BUFSIZ
#define NSS_LINELEN_PUBLICKEY   NSS_BUFSIZ
#define NSS_LINELEN_RPC         NSS_BUFSIZ
#define NSS_LINELEN_SERVICES    NSS_BUFSIZ
#define NSS_LINELEN_SHADOW      NSS_BUFSIZ
#define NSS_LINELEN_ETHERS      NSS_BUFSIZ
#define NSS_LINELEN_BOOTPARAMS  NSS_BUFSIZ

#define NSS_LINELEN_ATTRDB      NSS_BUFSIZ

#define NSS_LINELEN_AUDITUSER   NSS_LINELEN_ATTRDB
#define NSS_LINELEN_AUTHATTR    NSS_LINELEN_ATTRDB
#define NSS_LINELEN_EXECATTR    NSS_LINELEN_ATTRDB
#define NSS_LINELEN_PROFATTR    NSS_LINELEN_ATTRDB
#define NSS_LINELEN_USERATTR    NSS_LINELEN_ATTRDB

#define NSS_MMAPLEN_EXECATTR    NSS_LINELEN_EXECATTR * 8

#define NSS_LINELEN_TSOL        NSS_BUFSIZ

#define NSS_LINELEN_TSOL_TP     NSS_LINELEN_TSOL
#define NSS_LINELEN_TSOL_RH     NSS_LINELEN_TSOL
#define NSS_LINELEN_TSOL_ZC     NSS_LINELEN_TSOL

/*
 * Reasonable defaults for 'buflen' values passed to _r functions.  The BSD
 * and SunOS 4.x implementations of the getXXXbyYYY() functions used hard-
 * coded array sizes;  the values here are meant to handle anything that
 * those implementations handled.
 * === These might more reasonably go in <pwd.h>, <netdb.h> et al
 */

#define NSS_BUFLEN_GROUP        NSS_LINELEN_GROUP
#define NSS_BUFLEN_HOSTS        \
        (NSS_LINELEN_HOSTS + (MAXALIASES + MAXADDRS + 2) * sizeof (char *))
#define NSS_BUFLEN_IPNODES      \
        (NSS_LINELEN_IPNODES + (MAXALIASES + MAXADDRS + 2) * sizeof (char *))
```

```
#define NSS_BUFLEN_NETGROUP     (MAXHOSTNAMELEN * 2 + LOGNAME_MAX_ILLUMOS + 3)
#define NSS_BUFLEN_NETGROUP     (MAXHOSTNAMELEN * 2 + LOGNAME_MAX + 3)
#define NSS_BUFLEN_NETWORKS     NSS_LINELEN_NETWORKS    /* === ?  + 35 * 4 */
#define NSS_BUFLEN_PASSWD       NSS_LINELEN_PASSWD
#define NSS_BUFLEN_PROJECT      (NSS_LINELEN_PROJECT + 800 * sizeof (char *))
#define NSS_BUFLEN_PROTOCOLS    NSS_LINELEN_PROTOCOLS   /* === ?  + 35 * 4 */
#define NSS_BUFLEN_PUBLICKEY    NSS_LINELEN_PUBLICKEY
#define NSS_BUFLEN_RPC          NSS_LINELEN_RPC         /* === ?  + 35 * 4 */
#define NSS_BUFLEN_SERVICES     NSS_LINELEN_SERVICES    /* === ?  + 35 * 4 */
#define NSS_BUFLEN_SHADOW       NSS_LINELEN_SHADOW
#define NSS_BUFLEN_ETHERS       NSS_LINELEN_ETHERS
#define NSS_BUFLEN_BOOTPARAMS   NSS_LINELEN_BOOTPARAMS

#define NSS_BUFLEN_ATTRDB       NSS_LINELEN_ATTRDB

#define NSS_BUFLEN_AUDITUSER    NSS_BUFLEN_ATTRDB
#define NSS_BUFLEN_AUTHATTR     NSS_BUFLEN_ATTRDB
#define NSS_BUFLEN_EXECATTR     NSS_BUFLEN_ATTRDB
#define NSS_BUFLEN_PROFATTR     NSS_BUFLEN_ATTRDB
#define NSS_BUFLEN_USERATTR     ((NSS_BUFLEN_ATTRDB) * 8)

#define NSS_BUFLEN_TSOL         NSS_LINELEN_TSOL
#define NSS_BUFLEN_TSOL_TP      NSS_BUFLEN_TSOL
#define NSS_BUFLEN_TSOL_RH      NSS_BUFLEN_TSOL
#define NSS_BUFLEN_TSOL_ZC      NSS_BUFLEN_TSOL

/*
 * Default cache door buffer size (2x largest buffer)
 */

#define NSS_BUFLEN_DOOR         ((NSS_BUFSIZ) * 16)

/*
 * Arguments and results, passed between the frontends and backends for
 * the well-known databases.  The getXbyY_r() and getXent_r() routines
 * use a common format that is further described below;  other routines
 * use their own formats.
 */

/*
 * The nss_str2ent_t routine is the data marshaller for the nsswitch.
 * it converts 'native files' format into 'entry' format as part of the
 * return processing for a getXbyY interface.
 *
 * The nss_groupstr_t routine does the real work for any backend
 * that can supply a netgroup entry as a string in /etc/group format
 */
#if defined(__STDC__)
typedef int             (*nss_str2ent_t)(const char *in, int inlen,
                                void *ent, char *buf, int buflen);

struct nss_groupsbymem;         /* forward definition */
typedef nss_status_t    (*nss_groupstr_t)(const char *instr, int inlen,
                                struct nss_groupsbymem *);
#else
typedef int             (*nss_str2ent_t)();
typedef nss_status_t    (*nss_groupstr_t)();
#endif

/*
 * The initgroups() function [see initgroups(3c)] needs to find all the
 *   groups to which a given user belongs.  To do this it calls
 *   _getgroupsbymember(), which is part of the frontend for the "group"
 *   database.
 * We want the same effect as if we used getgrent_r() to enumerate the
 *   entire groups database (possibly from multiple sources), but getgrent_r()
 *   is too inefficient.  Most backends can do better if they know they’re
 *   meant to scan all groups;  hence there’s a separate backend operation,
 *   NSS_DBOP_GROUP_BYMEMBER, which uses the nss_groupsbymem struct.
 * Note that the normal return-value from such a backend, even when it
 *   successfully finds matching group entries, is NSS_NOTFOUND, because
 *   this tells the switch engine to keep searching in any more sources.
 *   In fact, the backends only return NSS_SUCCESS if they find enough
 *   matching entries that the gid_array is completely filled, in which
 *   case the switch engine should stop searching.
 * If the force_slow_way field is set, the backend should eschew any cached
 *   information (e.g. the YP netid.byname map or the NIS+ cred.org_dir table)
 *   and should instead grind its way through the group map/table/whatever.
 */

struct nss_groupsbymem {                        /* For _getgroupsbymember() */
/* in: */
        const char      *username;
        gid_t           *gid_array;
        int             maxgids;
        int             force_slow_way;
        nss_str2ent_t   str2ent;
        nss_groupstr_t  process_cstr;

/* in_out: */
        int             numgids;
};
```
______***unchanged_portion_omitted_***

```
**********************************************************
    6486 Wed Apr  3 09:33:12 2013
new/usr/src/lib/libbsm/common/audit_ftpd.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <stdio.h>
#include <sys/fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>
#include <unistd.h>

#include <sys/socket.h>
#include <sys/sockio.h>
#include <netinet/in.h>
#include <tsol/label.h>

#include <bsm/audit.h>
#include <bsm/audit_record.h>
#include <bsm/audit_uevents.h>
#include <bsm/libbsm.h>
#include <bsm/audit_private.h>

#include <locale.h>
#include <pwd.h>
#include <generic.h>

#define BAD_PASSWD      (1)
#define UNKNOWN_USER    (2)
#define EXCLUDED_USER   (3)
#define NO_ANONYMOUS    (4)
#define MISC_FAILURE    (5)

static char             luser[LOGNAME_MAX_ILLUMOS + 1];
static char             luser[LOGNAME_MAX + 1];

static void generate_record(char *, int, char *);
```

```
static int selected(uid_t, char *, au_event_t, int);

void
audit_ftpd_bad_pw(char *uname)
{
        if (cannot_audit(0)) {
                return;
        }
        (void) strncpy(luser, uname, LOGNAME_MAX_ILLUMOS);
        (void) strncpy(luser, uname, LOGNAME_MAX);
        generate_record(luser, BAD_PASSWD, dgettext(bsm_dom, "bad password"));
}


void
audit_ftpd_unknown(char *uname)
{
        if (cannot_audit(0)) {
                return;
        }
        (void) strncpy(luser, uname, LOGNAME_MAX_ILLUMOS);
        (void) strncpy(luser, uname, LOGNAME_MAX);
        generate_record(luser, UNKNOWN_USER, dgettext(bsm_dom, "unknown user"));
}


void
audit_ftpd_excluded(char *uname)
{
        if (cannot_audit(0)) {
                return;
        }
        (void) strncpy(luser, uname, LOGNAME_MAX_ILLUMOS);
        (void) strncpy(luser, uname, LOGNAME_MAX);
        generate_record(luser, EXCLUDED_USER, dgettext(bsm_dom,
            "excluded user"));
}
______unchanged_portion_omitted_

void
audit_ftpd_success(char *uname)
{
        if (cannot_audit(0)) {
                return;
        }
        (void) strncpy(luser, uname, LOGNAME_MAX_ILLUMOS);
        (void) strncpy(luser, uname, LOGNAME_MAX);
        generate_record(luser, 0, "");
}
______unchanged_portion_omitted_
```

```
**********************************************************
    2935 Wed Apr  3 09:33:12 2013
new/usr/src/lib/libc/port/gen/getlogin.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

#pragma weak _getlogin = getlogin
#pragma weak _getlogin_r = getlogin_r

#include "lint.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#include <limits.h>
#include "utmpx.h"
#include <unistd.h>
#include <errno.h>
#include <thread.h>
#include <synch.h>
#include <mtlib.h>
#include "tsd.h"

/*
 * Use the full length of a login name, from LOGIN_NAME_MAX .
 * The utmpx interface provides for a 32 character login name.
 * XXX - _POSIX_LOGIN_NAME_MAX limits the length of a login name.  The utmpx
 * interface provides for a 32 character login name, but for the sake of
 * compatibility, we are still using the old utmp-imposed limit.
 */

/*
```

```
 * POSIX.1c Draft-6 version of the function getlogin_r.
 * It was implemented by Solaris 2.3.
 */
char *
getlogin_r(char *answer, int namelen)
{
        int             uf;
        off64_t         me;
        struct futmpx   ubuf;

        /* Required minimum */
        if (namelen < _POSIX_LOGIN_NAME_MAX) {
                errno = ERANGE;
                return (NULL);
        }

        if ((me = (off64_t)ttyslot()) < 0)
                return (NULL);
        if ((uf = open64(UTMPX_FILE, 0)) < 0)
                return (NULL);
        (void) lseek64(uf, me * sizeof (ubuf), SEEK_SET);
        if (read(uf, &ubuf, sizeof (ubuf)) != sizeof (ubuf)) {
                (void) close(uf);
                return (NULL);
        }
        (void) close(uf);
        if (ubuf.ut_user[0] == '\0')
                return (NULL);

        /* Insufficient buffer size */
        if (namelen < strnlen(&ubuf.ut_user[0], LOGIN_NAME_MAX - 1)) {
                errno = ERANGE;
                return (NULL);
        }
        (void) strncpy(&answer[0], &ubuf.ut_user[0],
            LOGIN_NAME_MAX - 1);
        answer[LOGIN_NAME_MAX - 1] = '\0';
            _POSIX_LOGIN_NAME_MAX - 1);
        answer[_POSIX_LOGIN_NAME_MAX - 1] = '\0';
        return (&answer[0]);
}
______unchanged_portion_omitted_

char *
getlogin(void)
{
        char *answer = tsdalloc(_T_LOGIN, LOGIN_NAME_MAX, NULL);
        char *answer = tsdalloc(_T_LOGIN, _POSIX_LOGIN_NAME_MAX, NULL);

        if (answer == NULL)
                return (NULL);
        return (getlogin_r(answer, LOGIN_NAME_MAX));
        return (getlogin_r(answer, _POSIX_LOGIN_NAME_MAX));
}
______unchanged_portion_omitted_
```

```
**********************************************************
   12091 Wed Apr  3 09:33:13 2013
new/usr/src/lib/libc/port/gen/sysconf.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/* sysconf(3C) - returns system configuration information */

#pragma weak _sysconf = sysconf

#include "lint.h"
#include <mtlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/sysconfig.h>
#include <limits.h>
#include <time.h>
#include <errno.h>
#include <nss_dbdefs.h>
#include <thread.h>
#include <xti.h>
#include "libc.h"
#include "xpg6.h"

/* from nss_common.c */
extern size_t _nss_get_bufsizes(int);

long
sysconf(int name)
{
        static int _pagesize = 0;
        static int _hz = 0;
        static pid_t _maxpid = 0;
        static int _stackprot = 0;
        static int _ngroups_max;
```

```
        extern int __xpg4;

        switch (name) {
                default:
                        errno = EINVAL;
                        return (-1L);

                case _SC_ARG_MAX:
                        return ((long)ARG_MAX);

                case _SC_CLK_TCK:
                        if (_hz <= 0)
                                _hz = _sysconfig(_CONFIG_CLK_TCK);
                        return (_hz);

                case _SC_JOB_CONTROL:
                        return ((long)_POSIX_JOB_CONTROL);

                case _SC_SAVED_IDS:
                        return ((long)_POSIX_SAVED_IDS);

                case _SC_CHILD_MAX:
                        return (_sysconfig(_CONFIG_CHILD_MAX));

                case _SC_NGROUPS_MAX:
                        if (_ngroups_max <= 0)
                                _ngroups_max = _sysconfig(_CONFIG_NGROUPS);
                        return (_ngroups_max);

                case _SC_OPEN_MAX:
                        return (_sysconfig(_CONFIG_OPEN_FILES));

                case _SC_VERSION:
                        if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
                                return (200112L);
                        else
                                return (199506L);

                case _SC_PAGESIZE:
                        if (_pagesize <= 0)
                                _pagesize = _sysconfig(_CONFIG_PAGESIZE);
                        return (_pagesize);

                case _SC_XOPEN_VERSION:
                        if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
                                return (600L);
                        else if (__xpg4 == 0)
                                return (_sysconfig(_CONFIG_XOPEN_VER));
                        else
                                return (4L);

                case _SC_XOPEN_XCU_VERSION:
                        if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
                                return (600L);
                        else
                                return (4L);

                /*
                 * old value for pre XPG5 conformant systems to match
                 * getpass() length.
                 * XPG5 special cased with __sysconf_xpg5()
                 * new value for default and modern XPG systems.
                 */
                case _SC_PASS_MAX:
                        if ((__xpg4 == 1) &&
                            (!(__xpg6 & _C99SUSv3_XPG6_sysconf_version)))
```

```
				return ((long)_PASS_MAX_XPG);
			else
				return ((long)_PASS_MAX);

		case _SC_LOGNAME_MAX:
			return ((long)LOGNAME_MAX);

		case _SC_STREAM_MAX:
			return (_sysconfig(_CONFIG_OPEN_FILES));

		case _SC_TZNAME_MAX:
			return (-1L);

		case _SC_NPROCESSORS_CONF:
			return (_sysconfig(_CONFIG_NPROC_CONF));

		case _SC_NPROCESSORS_ONLN:
			return (_sysconfig(_CONFIG_NPROC_ONLN));

		case _SC_NPROCESSORS_MAX:
			return (_sysconfig(_CONFIG_NPROC_MAX));

		case _SC_STACK_PROT:
			if (_stackprot == 0)
				_stackprot = _sysconfig(_CONFIG_STACK_PROT);
			return (_stackprot);

		/* POSIX.4 names */

		/*
		 * Each of the following also have _POSIX_* symbols
		 * defined in <unistd.h>. Values here should align
		 * with values in the header. Up until the SUSv3 standard
		 * we defined these simply as 1. With the introduction
		 * of the new revision, these were changed to 200112L.
		 * The standard allows us to change the value, however,
		 * we have kept both values in case application programs
		 * are relying on the previous value even though an
		 * application doing so is technically wrong.
		 */
		case _SC_ASYNCHRONOUS_IO:
		case _SC_FSYNC:
		case _SC_MAPPED_FILES:
		case _SC_MEMLOCK:
		case _SC_MEMLOCK_RANGE:
		case _SC_MEMORY_PROTECTION:
		case _SC_MESSAGE_PASSING:
		case _SC_PRIORITY_SCHEDULING:
		case _SC_REALTIME_SIGNALS:
		case _SC_SEMAPHORES:
		case _SC_SHARED_MEMORY_OBJECTS:
		case _SC_SYNCHRONIZED_IO:
		case _SC_TIMERS:
			if (__xpg6 & _C99SUSv3_mode_ON)
				return (200112L);
			else
				return (1L);

		case _SC_PRIORITIZED_IO:
#ifdef _POSIX_PRIORITIZED_IO
			return (1L);
#else
			return (-1L);
#endif

		case _SC_AIO_LISTIO_MAX:
```

```
			return (_sysconfig(_CONFIG_AIO_LISTIO_MAX));

		case _SC_AIO_MAX:
			return (_sysconfig(_CONFIG_AIO_MAX));

		case _SC_AIO_PRIO_DELTA_MAX:
			return (_sysconfig(_CONFIG_AIO_PRIO_DELTA_MAX));

		case _SC_DELAYTIMER_MAX:
			return (_sysconfig(_CONFIG_DELAYTIMER_MAX));

		case _SC_MQ_OPEN_MAX:
			return (_sysconfig(_CONFIG_MQ_OPEN_MAX));

		case _SC_MQ_PRIO_MAX:
			return (_sysconfig(_CONFIG_MQ_PRIO_MAX));

		case _SC_RTSIG_MAX:
			return (_sysconfig(_CONFIG_RTSIG_MAX));

		case _SC_SEM_NSEMS_MAX:
			return (_sysconfig(_CONFIG_SEM_NSEMS_MAX));

		case _SC_SEM_VALUE_MAX:
			return (_sysconfig(_CONFIG_SEM_VALUE_MAX));

		case _SC_SIGQUEUE_MAX:
			return (_sysconfig(_CONFIG_SIGQUEUE_MAX));

		case _SC_SIGRT_MAX:
			return (_sysconfig(_CONFIG_SIGRT_MAX));

		case _SC_SIGRT_MIN:
			return (_sysconfig(_CONFIG_SIGRT_MIN));

		case _SC_TIMER_MAX:
			return (_sysconfig(_CONFIG_TIMER_MAX));

		case _SC_PHYS_PAGES:
			return (_sysconfig(_CONFIG_PHYS_PAGES));

		case _SC_AVPHYS_PAGES:
			return (_sysconfig(_CONFIG_AVPHYS_PAGES));

		/* XPG4/POSIX.1-1990/POSIX.2-1992 names */
		case _SC_2_C_BIND:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_CHAR_TERM:
			return ((long)_POSIX2_CHAR_TERM);

		case _SC_2_C_DEV:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_C_VERSION:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (199209L);
		case _SC_2_FORT_DEV:
			return (-1L);

		case _SC_2_FORT_RUN:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_LOCALEDEF:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_SW_DEV:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_UPE:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (1L);

		case _SC_2_VERSION:
			if (__xpg6 & _C99SUSv3_XPG6_sysconf_version)
				return (200112L);
			else
				return (199209L);

		case _SC_BC_BASE_MAX:
			return ((long)BC_BASE_MAX);

		case _SC_BC_DIM_MAX:
			return ((long)BC_DIM_MAX);

		case _SC_BC_SCALE_MAX:
			return ((long)BC_SCALE_MAX);

		case _SC_BC_STRING_MAX:
			return ((long)BC_STRING_MAX);

		case _SC_COLL_WEIGHTS_MAX:
			return ((long)COLL_WEIGHTS_MAX);

		case _SC_EXPR_NEST_MAX:
			return ((long)EXPR_NEST_MAX);

		case _SC_LINE_MAX:
			return ((long)LINE_MAX);

		case _SC_RE_DUP_MAX:
			return ((long)RE_DUP_MAX);

		case _SC_XOPEN_CRYPT:
			return (1L);

		case _SC_XOPEN_ENH_I18N:
			return ((long)_XOPEN_ENH_I18N);

		case _SC_XOPEN_SHM:
			return ((long)_XOPEN_SHM);
```

```
		/* XPG4v2 (SUS) names */
		case _SC_XOPEN_UNIX:
			return (1L);

		case _SC_XOPEN_LEGACY:
			return (1L);

		case _SC_ATEXIT_MAX:
			return (-1L);

		case _SC_IOV_MAX:
			return ((long)IOV_MAX);

		case _SC_T_IOV_MAX:
			return ((long)T_IOV_MAX);

		/* XPG5 (SUSv2) names */
		case _SC_XOPEN_REALTIME:
			return (1L);

		case _SC_XOPEN_REALTIME_THREADS:
#if defined(_POSIX_THREAD_PRIORITY_SCHEDULING) && \
	defined(_POSIX_THREAD_PRIO_INHERIT) && \
	defined(_POSIX_THREAD_PRIO_PROTECT)
			return (1L);
#else
			return (-1L);
#endif

		case _SC_XBS5_ILP32_OFF32:
			return (1L);

		case _SC_XBS5_ILP32_OFFBIG:
			return (1L);

		case _SC_XBS5_LP64_OFF64:
			return (1L);

		case _SC_XBS5_LPBIG_OFFBIG:
			return (1L);

		/* POSIX.1c names */
		case _SC_THREAD_DESTRUCTOR_ITERATIONS:
			return (-1L);

		case _SC_GETGR_R_SIZE_MAX:
			return ((long)_nss_get_bufsizes(_SC_GETGR_R_SIZE_MAX));

		case _SC_GETPW_R_SIZE_MAX:
			return ((long)NSS_BUFLEN_PASSWD);

		case _SC_LOGIN_NAME_MAX:
			return ((long)(LOGIN_NAME_MAX));
			return ((long)(LOGNAME_MAX + 1));

		case _SC_THREAD_KEYS_MAX:
			return (-1L);

		case _SC_THREAD_STACK_MIN:
			return ((long)thr_min_stack());

		case _SC_THREAD_THREADS_MAX:
			return (-1L);

		case _SC_TTY_NAME_MAX:
			return ((long)TTYNAME_MAX);
		case _SC_BARRIERS:
			return ((long)_POSIX_BARRIERS);

		case _SC_CLOCK_SELECTION:
			return ((long)_POSIX_CLOCK_SELECTION);

		case _SC_MONOTONIC_CLOCK:
			return ((long)_POSIX_MONOTONIC_CLOCK);

		case _SC_SPAWN:
			return ((long)_POSIX_SPAWN);

		case _SC_SPIN_LOCKS:
			return ((long)_POSIX_SPIN_LOCKS);

		case _SC_THREADS:
		case _SC_THREAD_ATTR_STACKADDR:
		case _SC_THREAD_ATTR_STACKSIZE:
		case _SC_THREAD_PRIORITY_SCHEDULING:
		case _SC_THREAD_PRIO_INHERIT:
		case _SC_THREAD_PRIO_PROTECT:
		case _SC_THREAD_PROCESS_SHARED:
		case _SC_THREAD_SAFE_FUNCTIONS:
			if (__xpg6 & _C99SUSv3_mode_ON)
				return (200112L);
			else
				return (1L);

		case _SC_TIMEOUTS:
			return ((long)_POSIX_TIMEOUTS);

		/* 1216676 - cache info */
		case _SC_COHER_BLKSZ:
			return (_sysconfig(_CONFIG_COHERENCY));

		case _SC_SPLIT_CACHE:
			return (_sysconfig(_CONFIG_SPLIT_CACHE));

		case _SC_ICACHE_SZ:
			return (_sysconfig(_CONFIG_ICACHESZ));

		case _SC_DCACHE_SZ:
			return (_sysconfig(_CONFIG_DCACHESZ));

		case _SC_ICACHE_LINESZ:
			return (_sysconfig(_CONFIG_ICACHELINESZ));

		case _SC_DCACHE_LINESZ:
			return (_sysconfig(_CONFIG_DCACHELINESZ));

		case _SC_ICACHE_BLKSZ:
			return (_sysconfig(_CONFIG_ICACHEBLKSZ));

		case _SC_DCACHE_BLKSZ:
			return (_sysconfig(_CONFIG_DCACHEBLKSZ));

		case _SC_DCACHE_TBLKSZ:
			return (_sysconfig(_CONFIG_DCACHETBLKSZ));

		case _SC_ICACHE_ASSOC:
			return (_sysconfig(_CONFIG_ICACHE_ASSOC));

		case _SC_DCACHE_ASSOC:
			return (_sysconfig(_CONFIG_DCACHE_ASSOC));
```

```
		case _SC_MAXPID:
			if (_maxpid <= 0)
				_maxpid = _sysconfig(_CONFIG_MAXPID);
			return (_maxpid);

		case _SC_CPUID_MAX:
			return (_sysconfig(_CONFIG_CPUID_MAX));

		case _SC_EPHID_MAX:
			return (_sysconfig(_CONFIG_EPHID_MAX));

		/* UNIX 03 names - XPG6/SUSv3/POSIX.1-2001 */

		case _SC_REGEXP:
			return ((long)_POSIX_REGEXP);

		case _SC_SHELL:
			return ((long)_POSIX_SHELL);

		case _SC_ADVISORY_INFO:
			return ((long)_POSIX_ADVISORY_INFO);

		case _SC_HOST_NAME_MAX:
			return ((long)_POSIX_HOST_NAME_MAX);

		case _SC_READER_WRITER_LOCKS:
			return ((long)_POSIX_READER_WRITER_LOCKS);

		case _SC_IPV6:
			return ((long)_POSIX_IPV6);

		case _SC_RAW_SOCKETS:
			return ((long)_POSIX_RAW_SOCKETS);

		case _SC_XOPEN_STREAMS:
			return ((long)_XOPEN_STREAMS);

		case _SC_SYMLOOP_MAX:
			return (_sysconfig(_CONFIG_SYMLOOP_MAX));

		case _SC_V6_ILP32_OFF32:
			return (1L);

		case _SC_V6_ILP32_OFFBIG:
			return (1L);

		case _SC_V6_LP64_OFF64:
			return (1L);

		case _SC_V6_LPBIG_OFFBIG:
			return (1L);

		/* Unsupported UNIX 03 options */
		case _SC_2_PBS:
		case _SC_2_PBS_ACCOUNTING:
		case _SC_2_PBS_CHECKPOINT:
		case _SC_2_PBS_LOCATE:
		case _SC_2_PBS_MESSAGE:
		case _SC_2_PBS_TRACK:
		case _SC_CPUTIME:
		case _SC_SPORADIC_SERVER:
		case _SC_SS_REPL_MAX:
		case _SC_THREAD_CPUTIME:
		case _SC_THREAD_SPORADIC_SERVER:
		case _SC_TRACE:
		case _SC_TRACE_EVENT_FILTER:
```

```
                case _SC_TRACE_EVENT_NAME_MAX:
                case _SC_TRACE_INHERIT:
                case _SC_TRACE_LOG:
                case _SC_TRACE_NAME_MAX:
                case _SC_TRACE_SYS_MAX:
                case _SC_TRACE_USER_EVENT_MAX:
                case _SC_TYPED_MEMORY_OBJECTS:
                        return (-1L);
        }
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   23360 Wed Apr  3 09:33:13 2013
new/usr/src/lib/nsswitch/ldap/common/getnetgrent.c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
/*
 * 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) 2013 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */


#include <syslog.h>
#include "ldap_common.h"

/* netgroup attributes filters */
#define _N_TRIPLE               "nisnetgrouptriple"
#define _N_MEMBER               "membernisnetgroup"

#define PRINT_VAL(a)            (((a).argc == 0) || ((a).argv == NULL) || \
                                    ((a).argv[0] == NULL)) ? "*" : (a).argv[0]
#define ISNULL(a)               (a == NULL ? "<NULL>" : a)
#define MAX_DOMAIN_LEN          1024
#define MAX_TRIPLE_LEN          (MAXHOSTNAMELEN + LOGNAME_MAX_ILLUMOS + \
#define MAX_TRIPLE_LEN          (MAXHOSTNAMELEN + LOGNAME_MAX + \
                                        MAX_DOMAIN_LEN + 5)

#define _F_SETMEMBER            "(&(objectClass=nisNetGroup)(cn=%s))"
#define _F_SETMEMBER_SSD        "(&(%%s)(cn=%s))"

#define N_HASH          257
#define COMMA           ','

static const char *netgrent_attrs[] = {
        _N_TRIPLE,
        _N_MEMBER,
        (char *)NULL
};
______unchanged_portion_omitted_
```

```
**********************************************************
   16720 Wed Apr  3 09:33:13 2013
new/usr/src/man/man1m/prstat.1m
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
'\" te
.\" Copyright (c) 2013 Gary Mills
.\" Copyright (c) 2006, 2009 Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH PRSTAT 1M "Jun 25, 2009"
.SH NAME
prstat \- report active process statistics
.SH SYNOPSIS
.LP
.nf
\fBprstat\fR [\fB-acHJLmRrtTv\fR] [\fB-d\fR u | d] [\fB-C\fR \fIpsrsetlist\fR] [
     [\fB-j\fR \fIprojlist\fR] [\fB-k\fR \fItasklist\fR] [\fB-n\fR \fIntop\fR[,\
     [\fB-p\fR \fIpidlist\fR] [\fB-P\fR \fIcpulist\fR] [\fB-s\fR \fIkey\fR | \fB
     [\fB-u\fR \fIeuidlist\fR] [\fB-U\fR \fIuidlist\fR] [\fB-z\fR \fIzoneidlist\
     [\fIinterval\fR [\fIcount\fR]]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBprstat\fR utility iteratively examines all active processes on the
system and reports statistics based on the selected output mode and sort order.
\fBprstat\fR provides options to examine only processes matching specified
\fBPID\fRs, \fBUID\fRs, zone \fBID\fRs, \fBCPU\fR \fBID\fRs, and processor set
\fBID\fRs.
.sp
.LP
The \fB-j\fR, \fB-k\fR, \fB-C\fR, \fB-p\fR, \fB-P\fR, \fB-u\fR, \fB-U\fR, and
\fB-z\fR options accept lists as arguments. Items in a list can be either
separated by commas or enclosed in quotes and separated by commas or spaces.
.sp
.LP
If you do not specify an option, \fBprstat\fR examines all processes and
reports statistics sorted by \fBCPU\fR usage.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
Report information about processes and users. In this mode \fBprstat\fR
displays separate reports about processes and users at the same time.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.sp .6
.RS 4n
Print new reports below previous reports instead of overprinting them.
Long names are not truncated in this mode.
```

```
.RE

.sp
.ne 2
.na
\fB\fB-C\fR \fIpsrsetlist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or lwps that are bound to processor sets in the given
list. Each processor set is identified by an integer as reported by
\fBpsrset\fR(1M). The load averages displayed are the sum of the load averages
of the specified processor sets (see \fBpset_getloadavg\fR(3C)). Processes with
one or more LWPs bound to processor sets in the given list are reported even
when the \fB-L\fR option is not used.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR \fBu | d\fR\fR
.ad
.sp .6
.RS 4n
Specify \fBu\fR for a printed representation of the internal representation of
time. See \fBtime\fR(2). Specify \fBd\fR for standard date format. See
\fBdate\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-h\fR \fIlgrplist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or lwps whose home \fIlgroup\fR is in the given list of
\fIlgroups\fR. No processes or lwps will be listed for invalid \fIlgroups\fR.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.sp .6
.RS 4n
Report information about home \fIlgroup\fR. In this mode, \fBprstat\fR adds an
extra column showing process or lwps home \fIlgroup\fR with the header LGRP.
.RE

.sp
.ne 2
.na
\fB\fB-j\fR \fIprojlist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or lwps whose project \fBID\fR is in the given list. Each
project \fBID\fR can be specified as either a project name or a numerical
project \fBID\fR. See \fBproject\fR(4).
.RE

.sp
.ne 2
.na
```

```
\fB\fB-J\fR\fR
.ad
.sp .6
.RS 4n
Report information about processes and projects. In this mode \fBprstat\fR
displays separate reports about processes and projects at the same time.
A trailing asterisk marks a long name that has been truncated
to fit the column.
.RE

.sp
.ne 2
.na
\fB\fB-k\fR \fItasklist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or lwps whose task \fBID\fR is in \fItasklist\fR.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.sp .6
.RS 4n
Report statistics for each light-weight process (\fBLWP\fR). By default,
\fBprstat\fR reports only the number of \fBLWP\fRs for each process.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Report microstate process accounting information. In addition to all fields
listed in \fB-v\fR mode, this mode also includes the percentage of time the
process has spent processing system traps, text page faults, data page faults,
waiting for user locks and waiting for \fBCPU\fR (latency time).
.RE

.sp
.ne 2
.na
\fB\fB-n\fR \fIntop\fR[\fI,nbottom\fR]\fR
.ad
.sp .6
.RS 4n
Restrict number of output lines. The \fIntop\fR argument determines how many
lines of process or \fBlwp\fR statistics are reported, and the \fInbottom\fR
argument determines how many lines of user, task, or projects statistics are
reported if the \fB-a\fR, \fB-t\fR, \fB-T\fR, or \fB-J\fR options are
specified. By default, \fBprstat\fR displays as many lines of output that fit
in a window or terminal. When you specify the \fB-c\fR option or direct the
output to a file, the default values for \fBntop\fR and \fBnbottom\fR are
\fB15\fR and \fB5\fR.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR \fIpidlist\fR\fR
.ad
```

```
.sp .6
.RS 4n
Report only processes whose process \fBID\fR is in the given list.
.RE

.sp
.ne 2
.na
\fB\fB-P\fR \fIcpulist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or \fBlwp\fRs which have most recently executed on a
\fBCPU\fR in the given list. Each \fBCPU\fR is identified by an integer as
reported by \fBpsrinfo\fR(1M).
.RE

.sp
.ne 2
.na
\fB\fB-R\fR\fR
.ad
.sp .6
.RS 4n
Put \fBprstat\fR in the real time scheduling class. When this option is used,
\fBprstat\fR is given priority over time-sharing and interactive processes.
This option is available only for superuser.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.sp .6
.RS 4n
Disable lookups for user names and project names. (Note that this does not
apply to lookups for the \fB-j\fR, \fB-u\fR, or \fB-U\fR options.)
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIkey\fR\fR
.ad
.sp .6
.RS 4n
Sort output lines (that is, processes, \fBlwp\fRs, or users) by \fIkey\fR in
descending order. Only one \fIkey\fR can be used as an argument.
.sp
There are five possible key values:
.sp
.ne 2
.na
\fBcpu\fR
.ad
.sp .6
.RS 4n
Sort by process \fBCPU\fR usage. This is the default.
.RE

.sp
.ne 2
.na
\fBpri\fR
.ad
```

```
.sp .6
.RS 4n
Sort by process priority.
.RE

.sp
.ne 2
.na
\fBrss\fR
.ad
.sp .6
.RS 4n
Sort by resident set size.
.RE

.sp
.ne 2
.na
\fBsize\fR
.ad
.sp .6
.RS 4n
Sort by size of process image.
.RE

.sp
.ne 2
.na
\fBtime\fR
.ad
.sp .6
.RS 4n
Sort by process execution time.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-S\fR \fIkey\fR\fR
.ad
.sp .6
.RS 4n
Sort output lines by \fIkey\fR in ascending order. Possible \fIkey\fR values
are the same as for the \fB-s\fR option. See \fB-s\fR.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.sp .6
.RS 4n
Report total usage summary for each user. The summary includes the total number
of processes or \fBLWP\fRs owned by the user, total size of process images,
total resident set size, total cpu time, and percentages of recent cpu time and
system memory.
.RE

.sp
.ne 2
.na
\fB\fB-T\fR\fR
.ad
```

```
.sp .6
.RS 4n
Report information about processes and tasks. In this mode \fBprstat\fR
displays separate reports about processes and tasks at the same time.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR \fIeuidlist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes whose effective user \fBID\fR is in the given list. Each
user \fBID\fR may be specified as either a login name or a numerical user
\fBID\fR.
.RE

.sp
.ne 2
.na
\fB\fB-U\fR \fIuidlis\fRt\fR
.ad
.sp .6
.RS 4n
Report only processes whose real user \fBID\fR is in the given list. Each user
\fBID\fR may be specified as either a login name or a numerical user \fBID\fR.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.sp .6
.RS 4n
Report verbose process usage. This output format includes the percentage of
time the process has spent in user mode, in system mode, and sleeping. It also
includes the number of voluntary and involuntary context switches, system calls
and the number of signals received. Statistics that are not reported are marked
with the \fB-\fR sign.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fIzoneidlist\fR\fR
.ad
.sp .6
.RS 4n
Report only processes or LWPs whose zone ID is in the given list. Each zone ID
can be specified as either a zone name or a numerical zone ID. See
\fBzones\fR(5).
.RE

.sp
.ne 2
.na
\fB\fB-Z\fR\fR
.ad
.sp .6
.RS 4n
Report information about processes and zones. In this mode, \fBprstat\fR
displays separate reports about processes and zones at the same time.
A trailing asterisk marks a long name that has been truncated
to fit the column.
```

```
.RE

.SH OUTPUT
.sp
.LP
The following list defines the column headings and the meanings of a
\fBprstat\fR report:
.sp
.ne 2
.na
\fBPID\fR
.ad
.sp .6
.RS 4n
The process \fBID\fR of the process.
.RE

.sp
.ne 2
.na
\fBUSERNAME\fR
.ad
.sp .6
.RS 4n
The real user (login) name or real user \fBID\fR.
A trailing asterisk marks a long name that has been truncated
to fit the column.
.RE

.sp
.ne 2
.na
\fBSWAP\fR
.ad
.sp .6
.RS 4n
The total virtual memory size of the process, including all mapped files and
devices, in kilobytes (\fBK\fR), megabytes (\fBM\fR), or gigabytes (\fBG\fR).
.RE

.sp
.ne 2
.na
\fBRSS\fR
.ad
.sp .6
.RS 4n
The resident set size of the process (\fBRSS\fR), in kilobytes (\fBK\fR),
megabytes (\fBM\fR), or gigabytes (\fBG\fR). The RSS value is an estimate
provided by \fBproc\fR(4) that might underestimate the actual resident set
size. Users who want to get more accurate usage information for capacity
planning should use the \fB-x\fR option to \fBpmap\fR(1) instead.
.RE

.sp
.ne 2
.na
\fBSTATE\fR
.ad
.sp .6
.RS 4n
The state of the process:
.sp
.ne 2
.na
\fBcpu\fIN\fR\fR
```

```
.ad
.sp .6
.RS 4n
Process is running on \fBCPU\fR \fIN\fR.
.RE

.sp
.ne 2
.na
\fBsleep\fR
.ad
.sp .6
.RS 4n
Sleeping: process is waiting for an event to complete.
.RE

.sp
.ne 2
.na
\fBwait\fR
.ad
.sp .6
.RS 4n
Waiting: process is waiting for CPU usage to drop to the CPU-caps enforced
limits. See the description of \fBCPU-caps\fR in \fBresource_controls\fR(5).
.RE

.sp
.ne 2
.na
\fBrun\fR
.ad
.sp .6
.RS 4n
Runnable: process in on run queue.
.RE

.sp
.ne 2
.na
\fBzombie\fR
.ad
.sp .6
.RS 4n
Zombie state: process terminated and parent not waiting.
.RE

.sp
.ne 2
.na
\fBstop\fR
.ad
.sp .6
.RS 4n
Process is stopped.
.RE

.RE

.sp
.ne 2
.na
\fBPRI\fR
.ad
.sp .6
.RS 4n
```

```
The priority of the process. Larger numbers mean higher priority.
.RE

.sp
.ne 2
.na
\fBNICE\fR
.ad
.sp .6
.RS 4n
Nice value used in priority computation. Only processes in certain scheduling
classes have a nice value.
.RE

.sp
.ne 2
.na
\fBTIME\fR
.ad
.sp .6
.RS 4n
The cumulative execution time for the process.
.RE

.sp
.ne 2
.na
\fBCPU\fR
.ad
.sp .6
.RS 4n
The percentage of recent \fBCPU\fR time used by the process. If executing in a
non-global \fBzone\fR and the pools facility is active, the percentage will be
that of the processors in the processor set in use by the pool to which the
\fBzone\fR is bound.
.RE

.sp
.ne 2
.na
\fBPROCESS\fR
.ad
.sp .6
.RS 4n
The name of the process (name of executed file).
.RE

.sp
.ne 2
.na
\fBLWPID\fR
.ad
.sp .6
.RS 4n
The \fBlwp\fR \fBID\fR of the \fBlwp\fR being reported.
.RE

.sp
.ne 2
.na
\fBNLWP\fR
.ad
.sp .6
.RS 4n
The number of \fBlwp\fRs in the process.
.RE
```

```
.sp
.LP
With the some options, in addition to a number of the column headings shown
above, there are:
.sp
.ne 2
.na
\fBNPROC\fR
.ad
.sp .6
.RS 4n
Number of processes in a specified collection.
.RE

.sp
.ne 2
.na
\fBMEMORY\fR
.ad
.sp .6
.RS 4n
Percentage of memory used by a specified collection of processes.
.RE

.sp
.LP
The following columns are displayed when the \fB-v\fR or \fB-m\fR option is
specified
.sp
.ne 2
.na
\fBUSR\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent in user mode.
.RE

.sp
.ne 2
.na
\fBSYS\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent in system mode.
.RE

.sp
.ne 2
.na
\fBTRP\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent in processing system traps.
.RE

.sp
.ne 2
.na
\fBTFL\fR
.ad
.sp .6
.RS 4n
```

```
The percentage of time the process has spent processing text page faults.
.RE

.sp
.ne 2
.na
\fBDFL\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent processing data page faults.
.RE

.sp
.ne 2
.na
\fBLCK\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent waiting for user locks.
.RE

.sp
.ne 2
.na
\fBSLP\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent sleeping.
.RE

.sp
.ne 2
.na
\fBLAT\fR
.ad
.sp .6
.RS 4n
The percentage of time the process has spent waiting for CPU.
.RE

.sp
.ne 2
.na
\fBVCX\fR
.ad
.sp .6
.RS 4n
The number of voluntary context switches.
.RE

.sp
.ne 2
.na
\fBICX\fR
.ad
.sp .6
.RS 4n
The number of involuntary context switches.
.RE

.sp
.ne 2
.na
```

```
\fBSCL\fR
.ad
.sp .6
.RS 4n
The number of system calls.
.RE

.sp
.ne 2
.na
\fBSIG\fR
.ad
.sp .6
.RS 4n
The number of signals received.
.RE

.sp
.LP
Under the \fB-L\fR option, one line is printed for each \fBlwp\fR in the
process and some reporting fields show the values for the \fBlwp\fR, not the
process.
.sp
.LP
The following column is displayed when the \fB-H\fR option is specified:
.sp
.ne 2
.na
\fBLGRP\fR
.ad
.sp .6
.RS 4n
The home \fIlgroup\fR of the process or lwp.
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIcount\fR\fR
.ad
.sp .6
.RS 4n
Specifies the number of times that the statistics are repeated. By default,
\fBprstat\fR reports statistics until a termination signal is received.
.RE

.sp
.ne 2
.na
\fB\fIinterval\fR\fR
.ad
.sp .6
.RS 4n
Specifies the sampling interval in seconds; the default interval is \fB5\fR
seconds.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRReporting the Five Most Active Super-User Processes
.sp
.LP
```

```
The following command reports the five most active super-user processes running
on \fBCPU1\fR and \fBCPU2\fR:

.sp
.in +2
.nf
example% prstat -u root -n 5 -P 1,2 1 1

PID   USERNAME  SWAP   RSS STATE  PRI  NICE      TIME  CPU PROCESS/LWP
 306   root     3024K 1448K sleep   58    0   0:00.00 0.3% sendmail/1
 102   root     1600K  592K sleep   59    0   0:00.00 0.1% in.rdisc/1
 250   root     1000K  552K sleep   58    0   0:00.00 0.0% utmpd/1
 288   root     1720K 1032K sleep   58    0   0:00.00 0.0% sac/1
   1   root      744K  168K sleep   58    0   0:00.00 0.0% init/1
TOTAL:       25, load averages:  0.05, 0.08, 0.12
.fi
.in -2
.sp

.LP
\fBExample 2 \fRDisplaying Verbose Process Usage Information
.sp
.LP
The following command displays verbose process usage information about
processes with lowest resident set sizes owned by users \fBroot\fR and
\fBjohn\fR.

.sp
.in +2
.nf
example% prstat -S rss -n 5 -vc -u root,john

 PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWP
   1 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 init/1
 102 root     0.0 0.0  -   -   -   -  100  -    0   0   3   0 in.rdisc/1
 250 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 utmpd/1
1185 john    0.0 0.0  -   -   -   -  100  -    0   0   0   0 csh/1
 240 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 powerd/4
 TOTAL:       71, load averages:  0.02, 0.04, 0.08

.fi
.in -2
.sp

.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.sp .6
.RS 4n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fB1\fR\fR
.ad
.sp .6
.RS 4n
An error occurred.
```

```
.RE

.SH SEE ALSO
.sp
.LP
\fBdate\fR(1), \fBlgrpinfo\fR(1), \fBplgrp\fR(1), \fBproc\fR(1), \fBps\fR(1),
\fBtime\fR(2), \fBpsrinfo\fR(1M), \fBpsrset\fR(1M), \fBsar\fR(1M),
\fBpset_getloadavg\fR(3C), \fBproc\fR(4), \fBproject\fR(4),
\fBattributes\fR(5), \fBresource_controls\fR(5), \fBzones\fR(5)
.SH NOTES
.sp
.LP
The snapshot of system usage displayed by \fBprstat\fR is true only for a
split-second, and it may not be accurate by the time it is displayed. When the
\fB-m\fR option is specified, \fBprstat\fR tries to turn on microstate
accounting for each process; the original state is restored when \fBprstat\fR
exits. See \fBproc\fR(4) for additional information about the microstate
accounting facility.
.sp
.LP
The total memory size reported in the SWAP and RSS columns for groups of
processes can sometimes overestimate the actual amount of memory used by
processes with shared memory segments.
```

```
************************************************************
   28316 Wed Apr  3 09:33:13 2013
new/usr/src/man/man3c/sysconf.3c
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
************************************************************
'\" te
.\" Copyright (c) 2013 Gary Mills
.\" Copyright (c) 2008, Sun Microsystems, Inc.  All Rights Reserved.
.\" Portions Copyright (c) 1992, X/Open Company Limited.  All Rights Reserved.
.\" Copyright 1989 AT&T
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 SYSCONF 3C "Mar 26, 2008"
.SH NAME
sysconf \- get configurable system variables
.SH SYNOPSIS
.LP
.nf
#include <unistd.h>

\fBlong\fR \fBsysconf\fR(\fBint\fR \fIname\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBsysconf()\fR function provides a method for an application to determine
the current value of a configurable system limit or option (variable).
.sp
.LP
The \fIname\fR argument represents the system variable to be queried. The
following table lists the minimal set of system variables from \fB<limits.h>\fR
and \fB<unistd.h>\fR that can be returned by \fBsysconf()\fR and the symbolic
constants defined in \fB<unistd.h>\fR that are the corresponding values used
for \fIname\fR on the  SPARC and x86 platforms.
.sp
.in +2
.nf
    Name                       Return Value               Meaning
_____________________________________________________________________
_SC_2_C_BIND            _POSIX2_C_BIND             Supports the C lang-
                                                   uage binding option
_SC_2_C_DEV             _POSIX2_C_DEV              Supports the C lang-
                                                   uage development
                                                   utilities option
_SC_2_C_VERSION         _POSIX2_C_VERSION          Integer value
                                                   indicates version
                                                   of ISO POSIX-2
                                                   standard (Commands)
_SC_2_CHAR_TERM         _POSIX2_CHAR_TERM          Supports at least
                                                   one terminal
_SC_2_FORT_DEV          _POSIX2_FORT_DEV           Supports FORTRAN
                                                   Development
                                                   Utilities Option
_SC_2_FORT_RUN          _POSIX2_FORT_RUN           Supports FORTRAN
                                                   Run-time Utilities
                                                   Option
_SC_2_LOCALEDEF         _POSIX2_LOCALEDEF          Supports creation
                                                   of locales by the
                                                   localedef utility
```

```
_SC_2_SW_DEV            _POSIX2_SW_DEV             Supports Software
                                                   Development Utility
                                                   Option
_SC_2_UPE               _POSIX2_UPE                Supports User
                                                   Portability
                                                   Utilities Option
_SC_2_VERSION           _POSIX2_VERSION            Integer value
                                                   indicates version
                                                   of ISO POSIX-2
                                                   standard (C language
                                                   binding)
_SC_AIO_LISTIO_MAX      AIO_LISTIO_MAX             Max number of I/O
                                                   operations in a
                                                   single list I/O call
                                                   supported
_SC_AIO_MAX             AIO_MAX                    Max number of
                                                   outstanding
                                                   asynchronous I/O
                                                   operations supported
_SC_AIO_PRIO_DELTA_MAX  AIO_PRIO_DELTA_MAX         Max amount by which
                                                   process can decrease
                                                   its asynchronous
                                                   I/O priority level
                                                   from its own
                                                   scheduling priority
_SC_ARG_MAX             ARG_MAX                    Max size of argv[]
                                                   plus envp[]
_SC_ASYNCHRONOUS_IO     _POSIX_ASYNCHRONOUS_IO     Supports
                                                   Asynchronous I/O
_SC_ATEXIT_MAX          ATEXIT_MAX                 Max number of
                                                   functions that can
                                                   be registered with
                                                   atexit()
_SC_AVPHYS_PAGES                                   Number of physical
                                                   memory pages not
                                                   currently in use by
                                                   system
_SC_BARRIERS            _POSIX_BARRIERS            Supports Barriers
                                                   option
_SC_BC_BASE_MAX         BC_BASE_MAX                Maximum obase values
                                                   allowed by bc
_SC_BC_DIM_MAX          BC_DIM_MAX                 Max number of
                                                   elements permitted
                                                   in array by bc
_SC_BC_SCALE_MAX        BC_SCALE_MAX               Max scale value
                                                   allowed by bc
_SC_BC_STRING_MAX       BC_STRING_MAX              Max length of string
                                                   constant allowed by
                                                   bc
_SC_CHILD_MAX           CHILD_MAX                  Max processes
                                                   allowed to a UID
_SC_CLK_TCK             CLK_TCK                    Ticks per second
                                                   (clock_t)
_SC_CLOCK_SELECTION     _POSIX_CLOCK_SELECTION     Supports Clock
                                                   Selection option
_SC_COLL_WEIGHTS_MAX    COLL_WEIGHTS_MAX           Max number of
                                                   weights that can be
                                                   assigned to entry of
                                                   the LC_COLLATE order
                                                   keyword in locale
                                                   definition file
_SC_CPUID_MAX                                      Max possible
                                                   processor ID
_SC_DELAYTIMER_MAX      DELAYTIMER_MAX             Max number of timer
                                                   expiration overruns
_SC_EXPR_NEST_MAX       EXPR_NEST_MAX              Max number of
```

```
                                                   parentheses by expr
_SC_FSYNC               _POSIX_FSYNC               Supports File
                                                   Synchronization
_SC_GETGR_R_SIZE_MAX                               Max size of group
                                                   entry buffer
_SC_GETPW_R_SIZE_MAX                               Max size of password
                                                   entry buffer
_SC_HOST_NAME_MAX       _POSIX_HOST_NAME_MAX       Maximum length of a
                                                   host name (excluding
                                                   terminating null)
_SC_IOV_MAX             IOV_MAX                    Max number of iovec
                                                   structures available
                                                   to one process for
                                                   use with readv()
                                                   and writev()
_SC_JOB_CONTROL         _POSIX_JOB_CONTROL         Job control
                                                   supported?
_SC_LINE_MAX            LINE_MAX                   Max length of input
                                                   line
_SC_LOGIN_NAME_MAX      LOGIN_NAME_MAX             Max length of login
_SC_LOGIN_NAME_MAX      LOGNAME_MAX + 1            Max length of login
                                                   name
_SC_LOGNAME_MAX         LOGNAME_MAX
_SC_MAPPED_FILES        _POSIX_MAPPED_FILES        Supports Memory
                                                   Mapped Files
_SC_MAXPID                                         Max pid value
_SC_MEMLOCK             _POSIX_MEMLOCK             Supports Process
                                                   Memory Locking
_SC_MEMLOCK_RANGE       _POSIX_MEMLOCK_RANGE       Supports Range
                                                   Memory Locking
_SC_MEMORY_PROTECTION   _POSIX_MEMORY_PROTECTION   Supports Memory
                                                   Protection
_SC_MESSAGE_PASSING     _POSIX_MESSAGE_PASSING     Supports Message
                                                   Passing
_SC_MONOTONIC_CLOCK     _POSIX_MONOTONIC_CLOCK     Supports Monotonic
                                                   Clock option
_SC_MQ_OPEN_MAX         MQ_OPEN_MAX                Max number of open
                                                   message queues a
                                                   process can hold
_SC_MQ_PRIO_MAX         MQ_PRIO_MAX                Max number of
                                                   message priorities
                                                   supported
_SC_NGROUPS_MAX         NGROUPS_MAX                Max simultaneous
                                                   groups to which
                                                   one can belong
_SC_NPROCESSORS_CONF                               Number of processors
                                                   configured
_SC_NPROCESSORS_MAX                                Max number of
                                                   processors supported
                                                   by platform
_SC_NPROCESSORS_ONLN                               Number of processors
                                                   online
_SC_OPEN_MAX            OPEN_MAX                   Max open files per
                                                   process
_SC_PAGESIZE            PAGESIZE                   System memory page
                                                   size
_SC_PAGE_SIZE           PAGESIZE                   Same as _SC_PAGESIZE
_SC_PASS_MAX            PASS_MAX                   Max number of
                                                   significant bytes
                                                   in a password
_SC_PHYS_PAGES                                     Total number of
                                                   pages of physical
                                                   memory in system
_SC_PRIORITIZED_IO      _POSIX_PRIORITIZED_IO      Supports Prioritized
                                                   I/O
_SC_PRIORITY_SCHEDULING _POSIX_PRIORITY_SCHEDULING Supports Process
```

```
                                                   Scheduling
_SC_RAW_SOCKETS         _POSIX_RAW_SOCKETS         Supports Raw Sockets
                                                   option
_SC_RE_DUP_MAX          RE_DUP_MAX                 Max number of
                                                   repeated occurrences
                                                   of a regular
                                                   expression permitted
                                                   when using interval
                                                   notation \e{m,n\e}
_SC_READER_WRITER_LOCKS _POSIX_READER_WRITER_LOCKS Supports IPV6 option
_SC_REALTIME_SIGNALS    _POSIX_REALTIME_SIGNALS    Supports Realtime
                                                   Signals
_SC_REGEXP              _POSIX_REGEXP              Supports Regular
                                                   Expression Handling
                                                   option
_SC_RTSIG_MAX           RTSIG_MAX                  Max number of
                                                   realtime signals
                                                   reserved for
                                                   application use
_SC_SAVED_IDS           _POSIX_SAVED_IDS           Saved IDs
                                                   (seteuid())
                                                   supported?
_SC_SEM_NSEMS_MAX       SEM_NSEMS_MAX              Max number of POSIX
                                                   semaphores a process
                                                   can have
_SC_SEM_VALUE_MAX       SEM_VALUE_MAX              Max value a POSIX
                                                   semaphore can have
_SC_SEMAPHORES          _POSIX_SEMAPHORES          Supports Semaphores
_SC_SHARED_MEMORY_      _POSIX_SHARED_MEMORY_      Supports Shared
   OBJECTS                 OBJECTS                 Memory Objects
_SC_SHELL               _POSIX_SHELL               Supports POSIX shell
_SC_SIGQUEUE_MAX        SIGQUEUE_MAX               Max number of queued
                                                   signals that a
                                                   process can send and
                                                   have pending at
                                                   receiver(s) at a
                                                   time
_SC_SPAWN               _POSIX_SPAWN               Supports Spawn option
_SC_SPIN_LOCKS          _POSIX_SPIN_LOCKS          Supports Spin Locks
                                                   option
_SC_STACK_PROT                                     Default stack
                                                   protection
_SC_STREAM_MAX          STREAM_MAX                 Number of streams
                                                   one process can
                                                   have open at a time
_SC_SYMLOOP_MAX         _POSIX_SYMLOOP_MAX         Max number of symbolic
                                                   links that can be
                                                   reliably traversed in
                                                   the resolution of a
                                                   pathname in the absence
                                                   of a loop
_SC_SYNCHRONIZED_IO     _POSIX_SYNCHRONIZED_IO     Supports
                                                   Synchronized I/O
_SC_THREAD_ATTR_        _POSIX_THREAD_ATTR_        Supports Thread
   STACKADDR               STACKADDR               Stack Address
                                                   Attribute option
_SC_THREAD_ATTR_        _POSIX_THREAD_ATTR_        Supports Thread
   STACKSIZE               STACKSIZE               Stack Size
                                                   Attribute option
_SC_THREAD_DESTRUCTOR_  PTHREAD_DESTRUCTOR_        Number attempts made
   ITERATIONS              ITERATIONS              to destroy thread-
                                                   specific data on
                                                   thread exit
_SC_THREAD_KEYS_MAX     PTHREAD_KEYS_MAX           Max number of data
                                                   keys per process
_SC_THREAD_PRIO_        _POSIX_THREAD_PRIO_        Supports Priority
```

```
   INHERIT                 INHERIT                 Inheritance option
_SC_THREAD_PRIO_        _POSIX_THREAD_PRIO_        Supports Priority
   PROTECT                 PROTECT                 Protection option
_SC_THREAD_PRIORITY_    _POSIX_THREAD_PRIORITY_    Supports Thread
   SCHEDULING              SCHEDULING              Execution
                                                   Scheduling option
_SC_THREAD_PROCESS_     _POSIX_THREAD_PROCESS_     Supports
   SHARED                  SHARED                  Process-Shared
                                                   Synchronization
                                                   option
_SC_THREAD_SAFE_        _POSIX_THREAD_SAFE_        Supports Thread-Safe
   FUNCTIONS               FUNCTIONS               Functions option
_SC_THREAD_STACK_MIN    PTHREAD_STACK_MIN          Min byte size of
                                                   thread stack storage
_SC_THREAD_THREADS_MAX  PTHREAD_THREADS_MAX        Max number of
                                                   threads per process
_SC_THREADS             _POSIX_THREADS             Supports Threads
                                                   option
_SC_TIMEOUTS            _POSIX_TIMEOUTS            Supports Timeouts
                                                   option
_SC_TIMER_MAX           TIMER_MAX                  Max number of timer
                                                   per process
                                                   supported
_SC_TIMERS              _POSIX_TIMERS              Supports Timers
_SC_TTY_NAME_MAX        TTYNAME_MAX                Max length of tty
                                                   device name
_SC_TZNAME_MAX          TZNAME_MAX                 Max number of bytes
                                                   supported for name
                                                   of a time zone
_SC_V6_ILP32_OFF32      _POSIX_V6_ILP32_OFF32      Supports  X/Open
                                                   ILP32 w/32-bit
                                                   offset build
                                                   environment
_SC_V6_ILP32_OFFBIG     _POSIX_V6_ILP32_OFFBIG     Supports  X/Open
                                                   ILP32 w/64-bit
                                                   offset build
                                                   environment
_SC_V6_LP64_OFF64       _POSIX_V6_LP64_OFF64       Supports  X/Open
                                                   LP64 w/64-bit
                                                   offset build
                                                   environment
_SC_V6_LPBIG_OFFBIG     _POSIX_V6_LPBIG_OFFBIG     Same as
                                                   _SC_V6_LP64_OFF64
_SC_VERSION             _POSIX_VERSION             POSIX.1 version
                                                   supported
_SC_XBS5_ILP32_OFF32    _XBS_ILP32_OFF32           Indicates support
                                                   for X/Open ILP32
                                                   w/32-bit offset
                                                   build environment
_SC_XBS5_ILP32_OFFBIG   _XBS5_ILP32_OFFBIG         Indicates support
                                                   for X/Open ILP32
                                                   w/64-bit offset
                                                   build environment
_SC_XBS5_LP64_OFF64     _XBS5_LP64_OFF64           Indicates support of
                                                   X/Open LP64,
                                                   64-bit offset
                                                   build environment
_SC_XBS5_LPBIG_OFFBIG   _XBS5_LP64_OFF64           Same as
                                                   _SC_XBS5_LP64_OFF64
_SC_XOPEN_CRYPT         _XOPEN_CRYPT               Supports X/Open
                                                   Encryption Feature
                                                   Group
_SC_XOPEN_ENH_I18N      _XOPEN_ENH_I18N            Supports X/Open
                                                   Enhanced
                                                   Internationalization
                                                   Feature Group
```

```
_SC_XOPEN_LEGACY        _XOPEN_LEGACY              Supports X/Open
                                                   Legacy Feature Group
_SC_XOPEN_REALTIME      _XOPEN_REALTIME            Supports X/Open
                                                   POSIX Realtime
                                                   Feature Group
_SC_XOPEN_REALTIME_     _XOPEN_REALTIME_THREADS    Supports X/Open
     THREADS                                       POSIX Reatime
                                                   Threads Feature
                                                   Group
_SC_XOPEN_SHM           _XOPEN_SHM                 Supports X/Open
                                                   Shared Memory
                                                   Feature Group
_SC_XOPEN_STREAMS       _POSIX_XOPEN_STREAMS       Supports XSI Streams
                                                   option group
_SC_XOPEN_UNIX          _XOPEN_UNIX                Supports X/Open CAE
                                                   Specification,
                                                   August 1994, System
                                                   Interfaces and
                                                   Headers, Issue 4,
                                                   Version 2
_SC_XOPEN_VERSION       _XOPEN_VERSION             Integer value
                                                   indicates version of
                                                   X/Open Portability
                                                   Guide to which
                                                   implementation
                                                   conforms
_SC_XOPEN_XCU_VERSION   _XOPEN_XCU_VERSION         Integer value
                                                   indicates version of
                                                   XCU specification to
                                                   which implementation
                                                   conforms
.fi
.in -2
.sp

.sp
.LP
The following options are not supported and return \(mi1:
.sp

.sp
.TS
l l
l l .
\fB_SC_2_PBS\fR \fB_POSIX2_PBS\fR
\fB_SC_2_PBS_ACCOUNTING\fR      \fB_POSIX2_PBS_ACCOUNTING\fR
\fB_SC_2_PBS_CHECKPOINT\fR      \fB_POSIX2_PBS_CHECKPOINT\fR
\fB_SC_2_PBS_LOCATE\fR  \fB_POSIX2_PBS_LOCATE\fR
\fB_SC_2_PBS_MESSAGE\fR \fB_POSIX2_PBS_MESSAGE\fR
\fB_SC_2_PBS_TRACK\fR   \fB_POSIX2_PBS_TRACK\fR
\fB_SC_ADVISORY_INFO\fR \fB_POSIX_ADVISORY_INFO\fR
\fB_SC_CPUTIME\fR       \fB_POSIX_CPUTIME\fR
\fB_SC_SPORADIC_SERVER\fR       \fB_POSIX_SPORADIC_SERVER\fR
\fB_SC_SS_REPL_MAX\fR   \fB_POSIX_SS_REPL_MAX\fR
\fB_SC_THREAD_CPUTIME\fR        \fB_POSIX_THREAD_CPUTIME\fR
\fB_SC_THREAD_SPORADIC_SERVER\fR        \fB_POSIX_THREAD_SPORADIC_SERVER\fR
\fB_SC_TRACE\fR \fB_POSIX_TRACE\fR
\fB_SC_TRACE_EVENT_FILTER\fR    \fB_POSIX_TRACE_EVENT_FILTER\fR
\fB_SC_TRACE_EVENT_NAME_MAX\fR  \fB_POSIX_TRACE_EVENT_NAME_MAX\fR
\fB_SC_TRACE_INHERIT\fR \fB_POSIX_TRACE_INHERIT\fR
\fB_SC_TRACE_LOG\fR     \fB_POSIX_TRACE_LOG\fR
\fB_SC_TRACE_NAME_MAX\fR        \fB_POSIX_TRACE_NAME_MAX\fR
\fB_SC_TRACE_SYS_MAX\fR \fB_POSIX_TRACE_SYS_MAX\fR
\fB_SC_TRACE_USER_EVENT_MAX\fR  \fB_POSIX_TRACE_USER_EVENT_MAX\fR
\fB_SC_TYPED_MEMORY_OBJECTS\fR  \fB_POSIX_TYPED_MEMORY_OBJECTS\fR
.TE
```

```
.SH RETURN VALUES
.sp
.LP
Upon successful completion, \fBsysconf()\fR returns the current variable value
on the system. The value returned will not be more restrictive than the
corresponding value described to the application when it was compiled with the
implementation’s <\fBlimits.h\fR>, <\fBunistd.h\fR> or <\fBtime.h\fR>. With
only a few obvious exceptions such as \fB_SC_AVPHYS_PAGES\fR and
\fB_SC_NPROCESSORS_ONLN\fR, the value will not change during the lifetime of
the calling process.
.sp
.LP
If \fIname\fR is an invalid value, \fBsysconf()\fR returns \fB\(mi1\fR and sets
\fBerrno\fR to indicate the error. If the variable corresponding to \fIname\fR
is associated with functionality that is not supported by the system,
\fBsysconf()\fR returns \fB\(mi1\fR without changing the value of \fIerrno\fR.
.sp
.LP
Calling \fBsysconf()\fR with the following returns \fB\(mi1\fR without setting
\fBerrno\fR, because no maximum limit can be determined. The system supports at
least the minimum values and can support higher values depending upon system
resources.
.sp
.in +2
.nf
Variable                            Minimum supported value
_SC_AIO_MAX                        _POSIX_AIO_MAX
_SC_ATEXIT_MAX                     32
_SC_MQ_OPEN_MAX                    32
_SC_THREAD_THREADS_MAX             _POSIX_THREAD_THREADS_MAX
_SC_THREAD_KEYS_MAX                _POSIX_THREAD_KEYS_MAX
_SC_THREAD_DESTRUCTOR_ITERATIONS   _POSIX_THREAD_DESTRUCTOR_ITERATIONS
.fi
.in -2

.sp
.LP
The following SPARC and x86 platform variables return \fBEINVAL\fR:
.sp
.in +2
.nf
_SC_COHER_BLKSZ        _SC_DCACHE_ASSOC
_SC_DCACHE_BLKSZ       _SC_DCACHE_LINESZ
_SC_DCACHE_SZ          _SC_DCACHE_TBLKSZ
_SC_ICACHE_ASSOC       _SC_ICACHE_BLKSZ
_SC_ICACHE_LINESZ      _SC_ICACHE_SZ
_SC_SPLIT_CACHE
.fi
.in -2

.SH ERRORS
.sp
.LP
The \fBsysconf()\fR function will fail if:
.sp
.ne 2
.na
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
The value of the \fIname\fR argument is invalid.
.RE

.SH ATTRIBUTES
.sp
```

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

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Architecture    SPARC and x86
_
Interface Stability     Committed
_
MT-Level        MT-Safe, Async-Signal-Safe
_
Standard        See \fBstandards\fR(5).
.TE

.SH SEE ALSO
.sp
.LP
\fBpooladm\fR(1M), \fBzoneadm\fR(1M), \fBfpathconf\fR(2), \fBseteuid\fR(2),
\fBsetrlimit\fR(2), \fBconfstr\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
A call to \fBsetrlimit()\fR can cause the value of \fBOPEN_MAX\fR to change.
.sp
.LP
Multiplying  \fBsysconf\fR(\fB_SC_PHYS_PAGES\fR) or
\fBsysconf\fR(\fB_SC_AVPHYS_PAGES\fR) by \fBsysconf\fR(\fB_SC_PAGESIZE\fR) to
determine memory amount in bytes can exceed the maximum values representable in
a 32-bit signed or unsigned integer.
.sp
.LP
The value of \fBCLK_TCK\fR can be variable and it should not be assumed that
\fBCLK_TCK\fR is a compile-time constant.
.sp
.LP
If the caller is in a non-global zone and the pools facility is active,
\fBsysconf\fR(\fB_SC_NPROCESSORS_CONF\fR) and
\fBsysconf\fR(\fB_SC_NPROCESSORS_ONLN\fR) return the number of processors in
the processor set of the pool to which the zone is bound.
```

```
**********************************************************
    9883 Wed Apr  3 09:33:13 2013
new/usr/src/man/man4/passwd.4
2989 Eliminate use of LOGNAME_MAX in ON
1166 useradd have warning with name more 8 chars
**********************************************************
'\" te
.\" Copyright (c) 2013 Gary Mills
.\" Copyright (c) 2004, 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 PASSWD 4 "Jul 28, 2004"
.SH NAME
passwd \- password file
.SH SYNOPSIS
.LP
.nf
\fB/etc/passwd\fR
.fi

.SH DESCRIPTION
.sp
.LP
The file \fB/etc/passwd\fR is a local source of information about users'
accounts. The password file can be used in conjunction with other naming
sources, such as the \fBNIS\fR maps \fBpasswd.byname\fR and \fBpasswd.bygid\fR,
data from the \fBNIS+\fR \fBpasswd\fR table, or password data stored on an LDAP
server. Programs use the \fBgetpwnam\fR(3C) routines to access this
information.
.sp
.LP
Each \fBpasswd\fR entry is a single line of the form:
.sp
.in +2
.nf
\fIusername\fR\fB:\fR\fIpassword\fR\fB:\fR\fIuid\fR\fB:\fR
\fIgid\fR\fB:\fR\fIgcos-field\fR\fB:\fR\fIhome-dir\fR\fB:\fR
\fIlogin-shell\fR
.fi
.in -2
.sp

.sp
.LP
where
.sp
.ne 2
.na
\fB\fIusername\fR\fR
.ad
.RS 15n
is the user's login name.
.sp
The login (\fBlogin\fR) and role (\fBrole\fR) fields accept a string of no more
than 32 bytes consisting of characters from the set of alphabetic
than eight bytes consisting of characters from the set of alphabetic
characters, numeric characters, period (\fB\&.\fR), underscore (\fB_\fR), and
hyphen (\fB-\fR). The first character should be alphabetic and the field should
contain at least one lower case alphabetic character. A warning message is
displayed if these restrictions are not met.
.sp
The \fBlogin\fR and \fBrole\fR fields must contain at least one character and
must not contain a colon (\fB:\fR) or a newline (\fB\en\fR).
.RE
```

```
.sp
.ne 2
.na
\fB\fIpassword\fR\fR
.ad
.RS 15n
is an empty field. The encrypted password for the user is in the corresponding
entry in the \fB/etc/shadow\fR file. \fBpwconv\fR(1M) relies on a special value
of '\fBx\fR' in the password field of \fB/etc/passwd\fR. If this value
of '\fBx\fR' exists in the password field of \fB/etc/passwd\fR, this indicates
that the password for the user is already in \fB/etc/shadow\fR and should not
be modified.
.RE

.sp
.ne 2
.na
\fB\fIuid\fR\fR
.ad
.RS 15n
is the user's unique numerical \fBID\fR for the system.
.RE

.sp
.ne 2
.na
\fB\fIgid\fR\fR
.ad
.RS 15n
is the unique numerical \fBID\fR of the group that the user belongs to.
.RE

.sp
.ne 2
.na
\fB\fIgcos-field\fR\fR
.ad
.RS 15n
is the user's real name, along with information to pass along in a mail-message
heading. (It is called the gcos-field for historical reasons.) An ''\fB&\fR\&''
(ampersand) in this field stands for the login name (in cases where the login
name appears in a user's real name).
.RE

.sp
.ne 2
.na
\fB\fIhome-dir\fR\fR
.ad
.RS 15n
is the pathname to the directory in which the user is initially positioned upon
logging in.
.RE

.sp
.ne 2
.na
\fB\fIlogin-shell\fR\fR
.ad
.RS 15n
is the user's initial shell program. If this field is empty, the default shell
is \fB/usr/bin/sh\fR.
.RE

.sp
```

```
.LP
The maximum value of the \fIuid\fR and \fIgid\fR fields is \fB2147483647\fR. To
maximize interoperability and compatibility, administrators are recommended to
assign users a range of \fBUID\fRs and \fBGID\fRs below \fB60000\fR where
possible. (\fBUID\fRs from \fB0\fR-\fB99\fR inclusive are reserved by the
operating system vendor for use in future applications. Their use by end system
users or vendors of layered products is not supported and may cause security
related issues with future applications.)
.sp
.LP
The password file is an \fBASCII\fR file that resides in the \fB/etc\fR
directory. Because the encrypted passwords on a secure system are always kept
in the \fBshadow\fR file, \fB/etc/passwd\fR has general read permission on all
systems and can be used by routines that map between numerical user \fBID\fRs
and user names.
.sp
.LP
Blank lines are treated as malformed entries in the \fBpasswd\fR file and cause
consumers of the file , such as \fBgetpwnam\fR(3C), to fail.
.sp
.LP
The password file can contain entries beginning with a ‘+’ (plus sign) or ’-’
(minus sign) to selectively incorporate entries from another naming service
source, such as NIS, NIS+, or LDAP.
.sp
.LP
A line beginning with a ’+’ means to incorporate entries from the naming
service source. There are three styles of the ’+’ entries in this file. A
single + means to insert all the entries from the alternate naming service
source at that point, while a +\fIname\fR means to insert the specific entry,
if one exists, from the naming service source. A +@\fInetgroup\fR means to
insert the entries for all members of the network group \fInetgroup\fR from the
alternate naming service. If a +\fIname\fR entry has a non-null \fBpassword\fR,
\fIgcos\fR, \fIhome-dir\fR, or \fIlogin-shell\fR field, the value of that field
overrides what is contained in the alternate naming service. The \fIuid\fR and
\fIgid\fR fields cannot be overridden.
.sp
.LP
A line beginning with a ‘\(mi’ means to disallow entries from the alternate
naming service. There are two styles of ‘-‘ entries in this file. -\fIname\fR
means to disallow any subsequent entries (if any) for \fIname\fR (in this file
or in a naming service), and -@\fInetgroup\fR means to disallow any subsequent
entries for all members of the network group \fInetgroup\fR.
.sp
.LP
This is also supported by specifying ‘‘passwd : compat’’ in
\fBnsswitch.conf\fR(4). The "compat" source might not be supported in future
releases. The preferred sources are \fBfiles\fR followed by the identifier of a
name service, such as \fBnis\fR or \fBldap\fR. This has the effect of
incorporating the entire contents of the naming service’s \fBpasswd\fR database
or password-related information after the \fBpasswd\fR file.
.sp
.LP
Note that in compat mode, for every \fB/etc/passwd\fR entry, there must be a
corresponding entry in the \fB/etc/shadow\fR file.
.sp
.LP
Appropriate precautions must be taken to lock the \fB/etc/passwd\fR file
against simultaneous changes if it is to be edited with a text editor;
\fBvipw\fR(1B) does the necessary locking.
.SH EXAMPLES
.LP
\fBExample 1 \fRSample \fBpasswd\fR File
.sp
.LP
The following is a sample \fBpasswd\fR file:
```

```

.sp
.in +2
.nf
root:x:0:1:Super-User:/:/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
.fi
.in -2
.sp

.sp
.LP
and the sample password entry from \fBnsswitch.conf\fR:

.sp
.in +2
.nf
passwd: files ldap
.fi
.in -2
.sp

.sp
.LP
In this example, there are specific entries for users \fBroot\fR and \fBfred\fR
to assure that they can login even when the system is running single-user. In
addition, anyone whose password information is stored on an LDAP server will be
able to login with their usual password, shell, and home directory.

.sp
.LP
If the password file is:

.sp
.in +2
.nf
root:x:0:1:Super-User:/:/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
+
.fi
.in -2
.sp

.sp
.LP
and the password entry in \fBnsswitch.conf\fR is:

.sp
.in +2
.nf
passwd: compat
.fi
.in -2
.sp

.sp
.LP
then all the entries listed in the \fBNIS\fR \fBpasswd.byuid\fR and
\fBpasswd.byname\fR maps will be effectively incorporated after the entries for
\fBroot\fR and \fBfred\fR. If the password entry in \fBnsswitch.conf\fR is:

.sp
.in +2
.nf
passwd_compat: ldap
passwd: compat
```

```
.fi
.in -2

.sp
.LP
then all password-related entries stored on the LDAP server will be
incorporated after the entries for \fBroot\fR and \fBfred\fR.

.sp
.LP
The following is a sample \fBpasswd\fR file when \fBshadow\fR does not exist:

.sp
.in +2
.nf
root:q.mJzTnu8icf.:0:1:Super-User:/:/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
.fi
.in -2
.sp

.sp
.LP
The following is a sample \fBpasswd\fR file when \fBshadow\fR does exist:

.sp
.in +2
.nf
root:##root:0:1:Super-User:/:/sbin/sh
fred:##fred:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
.fi
.in -2
.sp

.sp
.LP
In this example, there are specific entries for users \fBroot\fR and
\fBfred\fR, to assure that they can log in even when the system is running
standalone. The user \fBjohn\fR will have his password entry in the naming
service source incorporated without change, anyone in the netgroup
\fBdocumentation\fR will have their password field disabled, and anyone else
will be able to log in with their usual password, shell, and home directory,
but with a \fIgcos\fR field of \fBGuest\fR

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/nsswitch.conf\fR\fR
.ad
.RS 22n

.RE

.sp
.ne 2
.na
\fB\fB/etc/passwd\fR\fR
.ad
.RS 22n
```

```

.RE

.sp
.ne 2
.na
\fB\fB/etc/shadow\fR\fR
.ad
.RS 22n

.RE

.SH SEE ALSO
.sp
.LP
\fBchgrp\fR(1), \fBchown\fR(1), \fBfinger\fR(1), \fBgroups\fR(1),
\fBlogin\fR(1), \fBnewgrp\fR(1), \fBnispasswd\fR(1), \fBpasswd\fR(1),
\fBsh\fR(1), \fBsort\fR(1), \fBdomainname\fR(1M), \fBgetent\fR(1M),
\fBin.ftpd\fR(1M), \fBpassmgmt\fR(1M), \fBpwck\fR(1M), \fBpwconv\fR(1M),
\fBsu\fR(1M), \fBuseradd\fR(1M), \fBuserdel\fR(1M), \fBusermod\fR(1M),
\fBa64l\fR(3C), \fBcrypt\fR(3C), \fBgetpw\fR(3C), \fBgetpwnam\fR(3C),
\fBgetspnam\fR(3C), \fBputpwent\fR(3C), \fBgroup\fR(4), \fBhosts.equiv\fR(4),
\fBnsswitch.conf\fR(4), \fBshadow\fR(4), \fBenviron\fR(5),
\fBunistd.h\fR(3HEAD)
.sp
.LP
\fISystem Administration Guide: Basic Administration\fR
```