```
**********************************************************
    9191 Mon Mar 25 12:53:25 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) {
```

```
143                 t = (struct node *)emalloc(sizeof (*t) + strlen(pwp->pw_name));
144                 t->next = root;
145                 root = t;
146                 strcpy(t->user, pwp->pw_name);
147                 t->ngroups = 1;
148                 if (!ngroups_max)
149                         t->groups = NULL;
150                 else {
151                         t->groups = (struct group *)
152                             emalloc(sizeof (struct group));
153                         t->groups->grp = pwp->pw_gid;
154                         t->groups->cnt = 1;
155                         t->groups->nxt = NULL;
156                 }
157         }
158 #endif

160         bufsize = MYBUFSIZE;
161         if ((buf = malloc(bufsize)) == NULL) {
162                 (void) fprintf(stderr, gettext(NOMEM));
163                 exit(1);
164         }
165         while (!feof(fptr) && !ferror(fptr)) {
166                 buf_len = 0;
167                 buf_off = buf;
168                 while (fgets(buf_off, (bufsize - buf_len), fptr) != NULL) {
169                         buf_len += strlen(buf_off);
170                         if (buf[buf_len - 1] == '\n' || feof(fptr))
171                                 break;
172                         tmpbuf = realloc(buf, (bufsize + MYBUFSIZE));
173                         if (tmpbuf == NULL) {
174                                 (void) fprintf(stderr, gettext(NOMEM));
175                                 exit(1);
176                         }
177                         bufsize += MYBUFSIZE;
178                         buf = tmpbuf;
179                         buf_off = buf + buf_len;
180                 }
181                 if (buf_len == 0)
182                         continue;

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

188                 lineno++;
189                 if (buf[0] == '\n')    /* blank lines are ignored */
190                 {
191                         code = 1;              /* exit with error code = 1 */
192                         eflag = 0;      /* force print of "blank" line */
193                         fprintf(stderr, "\n%s %d\n", gettext(BLANKLINE),
194                             lineno);
195                         continue;
196                 }

198                 if (buf[buf_len - 1] == '\n') {
199                         if ((tmpbuf = strdup(buf)) == NULL) {
200                                 (void) fprintf(stderr, gettext(NOMEM));
201                                 exit(1);
202                         }
203                         tmpbuf[buf_len - 1] = ',';
204                 } else {
205                         if ((tmpbuf = malloc(buf_len + 2)) == NULL) {
206                                 (void) fprintf(stderr, gettext(NOMEM));
207                                 exit(1);
208                         }
```



```
209                         (void) strcpy(tmpbuf, buf);
210                         tmpbuf[buf_len++] = ',';
211                         tmpbuf[buf_len] = '\0';
212                 }

214                 colons = 0;
215                 eflag = 0;
216                 badchar = 0;
217                 baddigit = 0;
218                 badlognam = 0;
219                 gid = 0;

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

224                 if (buf[0] == '+' || buf[0] == '-')  {
225                         /*
226                          * Should set flag here to allow special case checking
227                          * in the rest of the code,
228                          * but for now, we'll just ignore this entry.
229                          */
230                         free(tmpbuf);
231                         continue;
232                 }

234                 /*      Check number of fields  */

236                 for (i = 0; buf[i] != NULL; i++) {
237                         if (buf[i] == ':') {
238                                 delim[colons] = i;
239                                 if (++colons > NUM_COLONS)
240                                         break;
241                         }
242                 }
243                 if (colons != NUM_COLONS) {
244                         error(BADLINE);
245                         free(tmpbuf);
246                         continue;
247                 }

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

252                 if (buf[0] == ':')
253                         error(NONAME);
254                 else {
255                         for (i = 0; buf[i] != ':'; i++) {
256                                 if (i >= _LOGNAME_MAX)
250                                 i (i >= LOGNAME_MAX)
257                                         longnam++;
258                                 if (!(islower(buf[i]) || isdigit(buf[i])))
259                                         badchar++;
260                         }
261                         if (longnam > 0)
262                                 error(LONGNAME);
263                         if (badchar > 0)
264                                 error(BADNAME);
265                 }

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

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

271                 if (len > 10 || len < 1)
272                         error(BADGID);
273                 else {
```

```
274                         for (i = (delim[1]+1); i < delim[2]; i++) {
275                                 if (! (isdigit(buf[i])))
276                                         baddigit++;
277                                 else if (baddigit == 0)
278                                         gid = gid * 10 + (gid_t)(buf[i] - '0');
279                                 /* converts ascii GID to decimal */
280                         }
281                         if (baddigit > 0)
282                                 error(BADGID);
283                         else if (gid > (gid_t)MAXUID)
284                                 error(BADGID);
285                 }

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

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

292                 listlen = strlen(nptr) - 1;

294                 while ((cptr = strchr(nptr, ',')) != NULL) {
295                         *cptr = NULL;
296                         if (*nptr == NULL) {
297                                 if (listlen)
298                                         error(NULLNAME);
299                                 nptr++;
300                                 continue;
301                         }

303                         for (t = root; t != NULL; t = t->next) {
304                                 if (strcmp(t->user, nptr) == 0)
305                                         break;
306                         }
307                         if (t == NULL) {
308 #ifndef ORIG_SVR4
309                                 /*
310                                  * User entry not found, so check if in
311                                  *  password file
312                                  */
313                                 struct passwd *pwp;

315                                 if ((pwp = getpwnam(nptr)) == NULL) {
316 #endif
317                                         badlognam++;
318                                         error(NOTFOUND);
319                                         goto getnext;
320 #ifndef ORIG_SVR4
321                                 }

323                                 /* Usrname found, so add entry to user-list */
324                                 t = (struct node *)
325                                     emalloc(sizeof (*t) + strlen(nptr));
326                                 t->next = root;
327                                 root = t;
328                                 strcpy(t->user, nptr);
329                                 t->ngroups = 1;
330                                 if (!ngroups_max)
331                                         t->groups = NULL;
332                                 else {
333                                         t->groups = (struct group *)
334                                             emalloc(sizeof (struct group));
335                                         t->groups->grp = pwp->pw_gid;
336                                         t->groups->cnt = 1;
337                                         t->groups->nxt = NULL;
338                                 }
339                         }
```



```
340 #endif
341                         if (!ngroups_max)
342                                 goto getnext;

344                         t->ngroups++;

346                         /*
347                          * check for duplicate logname in group
348                          */

350                         for (gp = t->groups; gp != NULL; gp = gp->nxt) {
351                                 if (gid == gp->grp) {
352                                         if (gp->cnt++ == 1) {
353                                                 badlognam++;
354                                                 if (gp->nxt == NULL)
355                                                         error(DUPNAME2);
356                                                 else
357                                                         error(DUPNAME);
358                                         }
359                                         goto getnext;
360                                 }
361                         }

363                         gp = (struct group *)emalloc(sizeof (struct group));
364                         gp->grp = gid;
365                         gp->cnt = 1;
366                         gp->nxt = t->groups;
367                         t->groups = gp;
368 getnext:
369                         nptr = ++cptr;
370                 }
371                 free(tmpbuf);
372         }

374         if (ngroups == 0) {
375                 fprintf(stderr, gettext("Group file '%s' is empty\n"), argv[1]);
376                 code = 1;
377         }

379         if (ngroups_max) {
380                 for (t = root; t != NULL; t = t->next) {
381                         if (t->ngroups > ngroups_max) {
382                                 fprintf(stderr, "\n\n%s%s (%d)\n",
383                                     NGROUPS, t->user, t->ngroups);
384                                 code = 1;
385                         }
386                 }
387         }
388         return (code);
389 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   21668 Mon Mar 25 12:53:25 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>

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

#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+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 ||
```

```
677                     (pw = getpwnam(tmp_name)) == NULL || (pw->pw_uid != uid) ||
678                     (pw->pw_name == NULL))
679                         pw = NULL;
680         }

682         /*
683          * If the login name doesn't work,  we try the first match for
684          * the current uid in the password file.
685          */
686         if (pw == NULL) {
687                 if (((pw = getpwuid(uid)) == NULL) || pw->pw_name == NULL) {
688                         preserve_error(gettext("cannot find username "
689                             "for uid %d"), uid);
690                         return (NULL);
691                 }
692         }

694         /*
695          * If projname wasn't supplied, we've done our best, so just return
696          * what we've got now. Alternatively, if newtask's invoker has
697          * superuser privileges, return the pw structure we've got now, with
698          * no further checking from inproj(). Superuser should be able to
699          * join any project, and the subsequent call to setproject() will
700          * allow this.
701          */
702         if (projname == NULL || getuid() == (uid_t)0)
703                 return (pw);

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

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

712                 /*
713                  * If the previous guesses didn't work, walk through all
714                  * project members and test for UID-equivalence.
715                  */

717                 if (getprojbyname(projname, &prj, prbuf,
718                     PROJECT_BUFSZ) == NULL) {
719                         preserve_error(gettext("unknown project \"%s\""),
720                             projname);
721                         return (NULL);
722                 }

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

728                         if (pw->pw_uid == uid) {
729                                 tmp_name = pw->pw_name;
730                                 break;
731                         }
732                 }

734                 if (tmp_name == NULL) {
735                         preserve_error(gettext("user \"%s\" is not a member of "
736                             "project \"%s\""), username, projname);
737                         return (NULL);
738                 }
739         }

741         return (pw);
```



```
742 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    2641 Mon Mar 25 12:53:25 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 */
```

```
**********************************************************
    2175 Mon Mar 25 12:53:25 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++;
        }

#ifdef  LOGNAME_MAX_ILLUMOS
        if (len > LOGNAME_MAX_ILLUMOS)
                return (LONGNAME);
#endif /* LOGNAME_MAX_ILLUMOS */

        /*
         * 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 Mon Mar 25 12:53:25 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 Mon Mar 25 12:53:26 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
```

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

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

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

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

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

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

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

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

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

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

155 #endif /* _MESSAGES_H */
```

```
**********************************************************
   17433 Mon Mar 25 12:53:26 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
```

```
127 } path_opt_t;


130 static void valid_input(path_opt_t, const char *);

132 int
133 main(argc, argv)
134 int argc;
135 char *argv[];
136 {
137         int ch, ret, mflag = 0, oflag = 0, Dflag = 0, **gidlist;
138         projid_t **projlist;
139         char *ptr;                      /* loc in a str, may be set by strtol */
140         struct group *g_ptr;
141         struct project p_ptr;
142         char mybuf[PROJECT_BUFSZ];
143         struct stat statbuf;            /* status buffer for stat */
144         int warning;
145         int busy = 0;
146         char **nargv;                   /* arguments for execvp of passmgmt */
147         int argindex;                   /* argument index into nargv */

149         cmdname = argv[0];

151         if (geteuid() != 0) {
152                 errmsg(M_PERM_DENIED);
153                 exit(EX_NO_PERM);
154         }

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

159         change_key(USERATTR_TYPE_KW, usertype);

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

168                 case 'c':
169                         comment = optarg;
170                         break;

172                 case 'D':
173                         Dflag++;
174                         break;

176                 case 'd':
177                         dir = optarg;
178                         break;

180                 case 'e':
181                         expirestr = optarg;
182                         break;

184                 case 'f':
185                         inactstr = optarg;
186                         break;

188                 case 'G':
189                         grps = optarg;
190                         break;

192                 case 'g':
```



```
193                         group = optarg;
194                         break;

196                 case 'k':
197                         skel_dir = optarg;
198                         break;

200                 case 'm':
201                         mflag++;
202                         break;

204                 case 'o':
205                         oflag++;
206                         break;

208                 case 'p':
209                         projects = optarg;
210                         break;

212                 case 's':
213                         shell = optarg;
214                         break;

216                 case 'u':
217                         uidstr = optarg;
218                         break;

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

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

228                 case 'R':
229                         if (is_role(usertype)) {
230                                 errmsg(M_ARUSAGE);
231                                 exit(EX_SYNTAX);
232                         }
233                         change_key(USERATTR_ROLES_KW, optarg);
234                         break;

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

240                 default:
241                 case '?':
242                         if (is_role(usertype))
243                                 errmsg(M_ARUSAGE);
244                         else
245                                 errmsg(M_AUSAGE);
246                         exit(EX_SYNTAX);
247                 }

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

252         if (Dflag) {
253                 /* DISPLAY mode */

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

```
259                     else
260                             errmsg(M_AUSAGE);
261                     exit(EX_SYNTAX);
262             }

264             if (uidstr != NULL || oflag || grps != NULL ||
265                 dir != NULL || mflag || comment != NULL) {
266                     if (is_role(usertype))
267                             errmsg(M_ARUSAGE);
268                     else
269                             errmsg(M_AUSAGE);
270                     exit(EX_SYNTAX);
271             }

273             /* Group must be an existing group */
274             if (group != NULL) {
275                     switch (valid_group(group, &g_ptr, &warning)) {
276                     case INVALID:
277                             errmsg(M_INVALID, group, "group id");
278                             exit(EX_BADARG);
279                             /*NOTREACHED*/
280                     case TOOBIG:
281                             errmsg(M_TOOBIG, "gid", group);
282                             exit(EX_BADARG);
283                             /*NOTREACHED*/
284                     case RESERVED:
285                     case UNIQUE:
286                             errmsg(M_GRP_NOTUSED, group);
287                             exit(EX_NAME_NOT_EXIST);
288                     }
289                     if (warning)
290                             warningmsg(warning, group);

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

295             }

297             /* project must be an existing project */
298             if (projects != NULL) {
299                     switch (valid_project(projects, &p_ptr, mybuf,
300                         sizeof (mybuf), &warning)) {
301                     case INVALID:
302                             errmsg(M_INVALID, projects, "project id");
303                             exit(EX_BADARG);
304                             /*NOTREACHED*/
305                     case TOOBIG:
306                             errmsg(M_TOOBIG, "projid", projects);
307                             exit(EX_BADARG);
308                             /*NOTREACHED*/
309                     case UNIQUE:
310                             errmsg(M_PROJ_NOTUSED, projects);
311                             exit(EX_NAME_NOT_EXIST);
312                     }
313                     if (warning)
314                             warningmsg(warning, projects);

316                     usrdefs->defproj = p_ptr.pj_projid;
317                     usrdefs->defprojname = p_ptr.pj_name;
318             }

320             /* base_dir must be an existing directory */
321             if (base_dir != NULL) {
322                     valid_input(BASEDIR, base_dir);
323                     usrdefs->defparent = base_dir;
324             }
```



```
326             /* inactivity period is an integer */
327             if (inactstr != NULL) {
328                     /* convert inactstr to integer */
329                     inact = strtol(inactstr, &ptr, 10);
330                     if (*ptr || inact < 0) {
331                             errmsg(M_INVALID, inactstr,
332                                 "inactivity period");
333                             exit(EX_BADARG);
334                     }

336                     usrdefs->definact = inact;
337             }

339             /* expiration string is a date, newer than today */
340             if (expirestr != NULL) {
341                     if (*expirestr) {
342                             if (valid_expire(expirestr, (time_t *)0)
343                                 == INVALID) {
344                                     errmsg(M_INVALID, expirestr,
345                                         "expiration date");
346                                     exit(EX_BADARG);
347                             }
348                             usrdefs->defexpire = expirestr;
349                     } else
350                             /* Unset the expiration date */
351                             usrdefs->defexpire = "";
352             }

354             if (shell != NULL) {
355                     valid_input(SHELL, shell);
356                     usrdefs->defshell = shell;
357             }
358             if (skel_dir != NULL) {
359                     valid_input(SKELDIR, skel_dir);
360                     usrdefs->defskel = skel_dir;
361             }
362             update_def(usrdefs);

364             /* change defaults for useradd */
365             if (putusrdef(usrdefs, usertype) < 0) {
366                     errmsg(M_UPDATE, "created");
367                     exit(EX_UPDATE);
368             }

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

374         }

376         /* ADD mode */

378         /* check syntax */
379         if (optind != argc - 1 || (skel_dir != NULL && !mflag)) {
380                 if (is_role(usertype))
381                         errmsg(M_ARUSAGE);
382                 else
383                         errmsg(M_AUSAGE);
384                 exit(EX_SYNTAX);
385         }

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

```
391                 exit(EX_BADARG);
392                 /*NOTREACHED*/
393
394         case NOTUNIQUE:
395                 errmsg(M_USED, logname);
396                 exit(EX_NAME_EXISTS);
397                 /*NOTREACHED*/
398
399         case LONGNAME:
400                 errmsg(M_TOO_LONG, logname);
401                 exit(EX_BADARG);
402                 /*NOTREACHED*/
403         }
404
405         if (warning)
406                 warningmsg(warning, logname);
407         if (uidstr != NULL) {
408                 /* convert uidstr to integer */
409                 errno = 0;
410                 uid = (uid_t)strtol(uidstr, &ptr, (int)10);
411                 if (*ptr || errno == ERANGE) {
412                         errmsg(M_INVALID, uidstr, "user id");
413                         exit(EX_BADARG);
414                 }
415
416                 switch (valid_uid(uid, NULL)) {
417                 case NOTUNIQUE:
418                         if (!oflag) {
419                                 /* override not specified */
420                                 errmsg(M_UID_USED, uid);
421                                 exit(EX_ID_EXISTS);
422                         }
423                         break;
424                 case RESERVED:
425                         errmsg(M_RESERVED, uid);
426                         break;
427                 case TOOBIG:
428                         errmsg(M_TOOBIG, "uid", uid);
429                         exit(EX_BADARG);
430                         break;
431                 }
432
433         } else {
434
435                 if ((uid = findnextuid()) < 0) {
436                         errmsg(M_INVALID, "default id", "user id");
437                         exit(EX_ID_EXISTS);
438                 }
439         }
440
441         if (group != NULL) {
442                 switch (valid_group(group, &g_ptr, &warning)) {
443                 case INVALID:
444                         errmsg(M_INVALID, group, "group id");
445                         exit(EX_BADARG);
446                         /*NOTREACHED*/
447                 case TOOBIG:
448                         errmsg(M_TOOBIG, "gid", group);
449                         exit(EX_BADARG);
450                         /*NOTREACHED*/
451                 case RESERVED:
452                 case UNIQUE:
453                         errmsg(M_GRP_NOTUSED, group);
454                         exit(EX_NAME_NOT_EXIST);
455                         /*NOTREACHED*/
456                 }
```



```
457
458                 if (warning)
459                         warningmsg(warning, group);
460                 gid = g_ptr->gr_gid;
461
462         } else gid = usrdefs->defgroup;
463
464         if (grps != NULL) {
465                 if (!*grps)
466                         /* ignore -G "" */
467                         grps = (char *)0;
468                 else if (!(gidlist = valid_lgroup(grps, gid)))
469                         exit(EX_BADARG);
470         }
471
472         if (projects != NULL) {
473                 if (! *projects)
474                         projects = (char *)0;
475                 else if (! (projlist = valid_lproject(projects)))
476                         exit(EX_BADARG);
477         }
478
479         /* if base_dir is provided, check its validity; otherwise default */
480         if (base_dir != NULL)
481                 valid_input(BASEDIR, base_dir);
482         else
483                 base_dir = usrdefs->defparent;
484
485         if (dir == NULL) {
486                 /* set homedir to home directory made from base_dir */
487                 (void) sprintf(homedir, "%s/%s", base_dir, logname);
488
489         } else if (REL_PATH(dir)) {
490                 errmsg(M_RELPATH, dir);
491                 exit(EX_BADARG);
492
493         } else
494                 (void) strcpy(homedir, dir);
495
496         if (mflag) {
497                 /* Does home dir. already exist? */
498                 if (stat(homedir, &statbuf) == 0) {
499                         /* directory exists - don't try to create */
500                         mflag = 0;
501
502                         if (check_perm(statbuf, uid, gid, S_IXOTH) != 0)
503                                 errmsg(M_NO_PERM, logname, homedir);
504                 }
505         }
506         /*
507          * if shell, skel_dir are provided, check their validity.
508          * Otherwise default.
509          */
510         if (shell != NULL)
511                 valid_input(SHELL, shell);
512         else
513                 shell = usrdefs->defshell;
514
515         if (skel_dir != NULL)
516                 valid_input(SKELDIR, skel_dir);
517         else
518                 skel_dir = usrdefs->defskel;
519
520         if (inactstr != NULL) {
521                 /* convert inactstr to integer */
522                 inact = strtol(inactstr, &ptr, 10);
```

```
523                 if (*ptr || inact < 0) {
524                         errmsg(M_INVALID, inactstr, "inactivity period");
525                         exit(EX_BADARG);
526                 }
527         } else inact = usrdefs->definact;

529         /* expiration string is a date, newer than today */
530         if (expirestr != NULL) {
531                 if (*expirestr) {
532                         if (valid_expire(expirestr, (time_t *)0) == INVALID) {
533                                 errmsg(M_INVALID, expirestr, "expiration date");
534                                 exit(EX_BADARG);
535                         }
536                         usrdefs->defexpire = expirestr;
537                 } else
538                         /* Unset the expiration date */
539                         expirestr = (char *)0;

541         } else expirestr = usrdefs->defexpire;

543         import_def(usrdefs);

545         /* must now call passmgmt */

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

553         if (comment != NULL) {
554                 /* comment */
555                 nargv[argindex++] = "-c";
556                 nargv[argindex++] = comment;
557         }

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

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

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

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

577         /* set expiration date */
578         if (expirestr != NULL) {
579                 nargv[argindex++] = "-e";
580                 nargv[argindex++] = expirestr;
581         }

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

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



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

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

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

599         /* now call passmgmt */
600         ret = PEX_FAILED;
601         /*
602          * If call_passmgmt fails for any reason other than PEX_BADUID, exit
603          * is invoked with an appropriate error message. If PEX_BADUID is
604          * returned, then if the user specified the ID, exit is invoked
605          * with an appropriate error message. Otherwise we try to pick a
606          * different ID and try again. If we run out of IDs, i.e. no more
607          * users can be created, then -1 is returned and we terminate via exit.
608          * If PEX_BUSY is returned we increment a count, since we will stop
609          * trying if PEX_BUSY reaches 3. For PEX_SUCCESS we immediately
610          * terminate the loop.
611          */
612         while (busy < 3 && ret != PEX_SUCCESS) {
613                 switch (ret = call_passmgmt(nargv)) {
614                 case PEX_SUCCESS:
615                         break;
616                 case PEX_BUSY:
617                         busy++;
618                         break;
619                 case PEX_HOSED_FILES:
620                         errmsg(M_HOSED_FILES);
621                         exit(EX_INCONSISTENT);
622                         break;

624                 case PEX_SYNTAX:
625                 case PEX_BADARG:
626                         /* should NEVER occur that passmgmt usage is wrong */
627                         if (is_role(usertype))
628                                 errmsg(M_ARUSAGE);
629                         else
630                                 errmsg(M_AUSAGE);
631                         exit(EX_SYNTAX);
632                         break;

634                 case PEX_BADUID:
635                         /*
636                          * The uid has been taken. If it was specified by a
637                          * user, then we must fail. Otherwise, keep trying
638                          * to get a good uid until we run out of IDs.
639                          */
640                         if (uidstr != NULL) {
641                                 errmsg(M_UID_USED, uid);
642                                 exit(EX_ID_EXISTS);
643                         } else {
644                                 if ((uid = findnextuid()) < 0) {
645                                         errmsg(M_INVALID, "default id",
646                                             "user id");
647                                         exit(EX_ID_EXISTS);
648                                 }
649                                 (void) sprintf(uidstring, "%u", uid);
650                         }
651                         break;

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

```
655                         errmsg(M_USED, logname);
656                         exit(EX_NAME_EXISTS);
657                         break;

659                 default:
660                         errmsg(M_UPDATE, "created");
661                         exit(ret);
662                         break;
663                 }
664         }
665         if (busy == 3) {
666                 errmsg(M_UPDATE, "created");
667                 exit(ret);
668         }

670         /* add group entry */
671         if ((grps != NULL) && edit_group(logname, (char *)0, gidlist, 0)) {
672                 errmsg(M_UPDATE, "created");
673                 cleanup(logname);
674                 exit(EX_UPDATE);
675         }

677         /* update project database */
678         if ((projects != NULL) &&
679             edit_project(logname, (char *)NULL, projlist, 0)) {
680                 errmsg(M_UPDATE, "created");
681                 cleanup(logname);
682                 exit(EX_UPDATE);
683         }

685         /* create home directory */
686         if (mflag &&
687             (create_home(homedir, skel_dir, uid, gid) != EX_SUCCESS)) {
688                 (void) edit_group(logname, (char *)0, (int **)0, 1);
689                 cleanup(logname);
690                 exit(EX_HOMEDIR);
691         }

693         return (ret);
694 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   15671 Mon Mar 25 12:53:26 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;
```

```
150         struct passwd *pstruct;         /* password struct for login */
151         struct passwd *pw;
152         struct group *g_ptr;    /* validated group from -g */
153         struct stat statbuf;            /* status buffer for stat */
154 #ifndef att
155         FILE *pwf;              /* fille ptr for opened passwd file */
156 #endif
157         int warning;
158         projid_t **projlist;
159         char **nargv;                   /* arguments for execvp of passmgmt */
160         int argindex;                   /* argument index into nargv */
161         userattr_t *ua;
162         char *val;
163         int isrole;                     /* current account is role */

165         cmdname = argv[0];

167         if (geteuid() != 0) {
168                 errmsg(M_PERM_DENIED);
169                 exit(EX_NO_PERM);
170         }

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

176         while ((ch = getopt(argc, argv,
177                         "c:d:e:f:G:g:l:mop:s:u:A:P:R:K:")) != EOF)
178                 switch (ch) {
179                 case 'c':
180                         comment = optarg;
181                         flag++;
182                         break;
183                 case 'd':
184                         dir = optarg;
185                         fail_if_busy = B_TRUE;
186                         flag++;
187                         break;
188                 case 'e':
189                         expire = optarg;
190                         flag++;
191                         break;
192                 case 'f':
193                         inactstr = optarg;
194                         flag++;
195                         break;
196                 case 'G':
197                         grps = optarg;
198                         flag++;
199                         break;
200                 case 'g':
201                         group = optarg;
202                         fail_if_busy = B_TRUE;
203                         flag++;
204                         break;
205                 case 'l':
206                         new_logname = optarg;
207                         fail_if_busy = B_TRUE;
208                         flag++;
209                         break;
210                 case 'm':
211                         mflag++;
212                         flag++;
213                         fail_if_busy = B_TRUE;
214                         break;
215                 case 'o':
```



```
216                         oflag++;
217                         flag++;
218                         fail_if_busy = B_TRUE;
219                         break;
220                 case 'p':
221                         projects = optarg;
222                         flag++;
223                         break;
224                 case 's':
225                         shell = optarg;
226                         flag++;
227                         break;
228                 case 'u':
229                         uidstr = optarg;
230                         flag++;
231                         fail_if_busy = B_TRUE;
232                         break;
233                 case 'A':
234                         change_key(USERATTR_AUTHS_KW, optarg);
235                         flag++;
236                         break;
237                 case 'P':
238                         change_key(USERATTR_PROFILES_KW, optarg);
239                         flag++;
240                         break;
241                 case 'R':
242                         change_key(USERATTR_ROLES_KW, optarg);
243                         flag++;
244                         break;
245                 case 'K':
246                         change_key(NULL, optarg);
247                         flag++;
248                         break;
249                 default:
250                 case '?':
251                         if (is_role(usertype))
252                                 errmsg(M_MRUSAGE);
253                         else
254                                 errmsg(M_MUSAGE);
255                         exit(EX_SYNTAX);
256                 }

258         if (optind != argc - 1 || flag == 0) {
259                 if (is_role(usertype))
260                         errmsg(M_MRUSAGE);
261                 else
262                         errmsg(M_MUSAGE);
263                 exit(EX_SYNTAX);
264         }

266         if ((!uidstr && oflag) || (mflag && !dir)) {
267                 if (is_role(usertype))
268                         errmsg(M_MRUSAGE);
269                 else
270                         errmsg(M_MUSAGE);
271                 exit(EX_SYNTAX);
272         }

274         logname = argv[optind];

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

```
282                 isrole = 1;

284         /* Verify that rolemod is used for roles and usermod for users */
285         if (isrole != is_role(usertype)) {
286                 if (isrole)
287                         errmsg(M_ISROLE);
288                 else
289                         errmsg(M_ISUSER);
290                 exit(EX_SYNTAX);
291         }

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

296         if (is_role(usertype)) {
297                 /* Roles can't have roles */
298                 if (getsetdefval(USERATTR_ROLES_KW, NULL) != NULL) {
299                         errmsg(M_MRUSAGE);
300                         exit(EX_SYNTAX);
301                 }
302                 /* If it was an ordinary user, delete its roles */
303                 if (!isrole)
304                         change_key(USERATTR_ROLES_KW, "");
305         }

307 #ifdef att
308         pw = getpwnam(logname);
309 #else
310         /*
311          * Do this with fgetpwent to make sure we are only looking on local
312          * system (since passmgmt only works on local system).
313          */
314         if ((pwf = fopen("/etc/passwd", "r")) == NULL) {
315                 errmsg(M_OOPS, "open", "/etc/passwd");
316                 exit(EX_FAILURE);
317         }
318         while ((pw = fgetpwent(pwf)) != NULL)
319                 if (strcmp(pw->pw_name, logname) == 0)
320                         break;

322         fclose(pwf);
323 #endif

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

329                 if (getpwnam_r(logname, &pwd, pwdb, sizeof (pwdb)) == NULL) {
330                         /* This user does not exist. */
331                         errmsg(M_EXIST, logname);
332                         exit(EX_NAME_NOT_EXIST);
333                 } else {
334                         /* This user exists in non-local name service. */
335                         errmsg(M_NONLOCAL, logname);
336                         exit(EX_NOT_LOCAL);
337                 }
338         }

340         pstruct = passwd_cpmalloc(pw);

342         /*
343          * We can't modify a logged in user if any of the following
344          * are being changed:
345          * uid (-u & -o), group (-g), home dir (-m), loginname (-l).
346          * If none of those are specified it is okay to go ahead
347          * some types of changes only take effect on next login, some
```



```
348          * like authorisations and profiles take effect instantly.
349          * One might think that -K type=role should require that the
350          * user not be logged in, however this would make it very
351          * difficult to make the root account a role using this command.
352          */
353         if (isbusy(logname)) {
354                 if (fail_if_busy) {
355                         errmsg(M_BUSY, logname, "change");
356                         exit(EX_BUSY);
357                 }
358                 warningmsg(WARN_LOGGED_IN, logname);
359         }

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

369                 case NOTUNIQUE:
370                         errmsg(M_USED, new_logname);
371                         exit(EX_NAME_EXISTS);
372                         /*NOTREACHED*/

374                 case LONGNAME:
375                         errmsg(M_TOO_LONG, new_logname);
376                         exit(EX_BADARG);
377                         /*NOTREACHED*/

379                 default:
380                         call_pass = 1;
381                         break;
382                 }
383                 if (warning)
384                         warningmsg(warning, logname);
385         }

387         if (uidstr) {
388                 /* convert uidstr to integer */
389                 errno = 0;
390                 uid = (uid_t)strtol(uidstr, &ptr, (int)10);
391                 if (*ptr || errno == ERANGE) {
392                         errmsg(M_INVALID, uidstr, "user id");
393                         exit(EX_BADARG);
394                 }

396                 if (uid != pstruct->pw_uid) {
397                         switch (valid_uid(uid, NULL)) {
398                         case NOTUNIQUE:
399                                 if (!oflag) {
400                                         /* override not specified */
401                                         errmsg(M_UID_USED, uid);
402                                         exit(EX_ID_EXISTS);
403                                 }
404                                 break;
405                         case RESERVED:
406                                 errmsg(M_RESERVED, uid);
407                                 break;
408                         case TOOBIG:
409                                 errmsg(M_TOOBIG, "uid", uid);
410                                 exit(EX_BADARG);
411                                 break;
412                         }
```

```
414                         call_pass = 1;

416                 } else {
417                         /* uid's the same, so don't change anything */
418                         uidstr = NULL;
419                         oflag = 0;
420                 }

422         } else uid = pstruct->pw_uid;

424         if (group) {
425                 switch (valid_group(group, &g_ptr, &warning)) {
426                 case INVALID:
427                         errmsg(M_INVALID, group, "group id");
428                         exit(EX_BADARG);
429                         /*NOTREACHED*/
430                 case TOOBIG:
431                         errmsg(M_TOOBIG, "gid", group);
432                         exit(EX_BADARG);
433                         /*NOTREACHED*/
434                 case UNIQUE:
435                         errmsg(M_GRP_NOTUSED, group);
436                         exit(EX_NAME_NOT_EXIST);
437                         /*NOTREACHED*/
438                 case RESERVED:
439                         gid = (gid_t)strtol(group, &ptr, (int)10);
440                         errmsg(M_RESERVED_GID, gid);
441                         break;
442                 }
443                 if (warning)
444                         warningmsg(warning, group);

446                 if (g_ptr != NULL)
447                         gid = g_ptr->gr_gid;
448                 else
449                         gid = pstruct->pw_gid;

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

456         } else gid = pstruct->pw_gid;

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

464         if (projects && *projects) {
465                 if (! (projlist = valid_lproject(projects)))
466                         exit(EX_BADARG);
467         } else
468                 projlist = (projid_t **)0;

470         if (dir) {
471                 if (REL_PATH(dir)) {
472                         errmsg(M_RELPATH, dir);
473                         exit(EX_BADARG);
474                 }
475                 if (strcmp(pstruct->pw_dir, dir) == 0) {
476                         /* home directory is the same so ignore dflag & mflag */
477                         dir = NULL;
478                         mflag = 0;
479                 } else call_pass = 1;
```



```
480         }

482         if (mflag) {
483                 if (stat(dir, &statbuf) == 0) {
484                         /* Home directory exists */
485                         if (check_perm(statbuf, pstruct->pw_uid,
486                             pstruct->pw_gid, S_IWOTH|S_IXOTH) != 0) {
487                                 errmsg(M_NO_PERM, logname, dir);
488                                 exit(EX_NO_PERM);
489                         }

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

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

496                 if (ret != EX_SUCCESS)
497                         exit(ret);
498         }

500         if (shell) {
501                 if (REL_PATH(shell)) {
502                         errmsg(M_RELPATH, shell);
503                         exit(EX_BADARG);
504                 }
505                 if (strcmp(pstruct->pw_shell, shell) == 0) {
506                         /* ignore s option if shell is not different */
507                         shell = NULL;
508                 } else {
509                         if (stat(shell, &statbuf) < 0 ||
510                             (statbuf.st_mode & S_IFMT) != S_IFREG ||
511                             (statbuf.st_mode & 0555) != 0555) {

513                                 errmsg(M_INVALID, shell, "shell");
514                                 exit(EX_BADARG);
515                         }

517                         call_pass = 1;
518                 }
519         }

521         if (comment)
522                 /* ignore comment if comment is not changed */
523                 if (strcmp(pstruct->pw_comment, comment))
524                         call_pass = 1;
525                 else
526                         comment = NULL;

528         /* inactive string is a positive integer */
529         if (inactstr) {
530                 /* convert inactstr to integer */
531                 inact = (int)strtol(inactstr, &ptr, 10);
532                 if (*ptr || inact < 0) {
533                         errmsg(M_INVALID, inactstr, "inactivity period");
534                         exit(EX_BADARG);
535                 }
536                 call_pass = 1;
537         }

539         /* expiration string is a date, newer than today */
540         if (expire) {
541                 if (*expire &&
542                     valid_expire(expire, (time_t *)0) == INVALID) {
543                         errmsg(M_INVALID, expire, "expiration date");
544                         exit(EX_BADARG);
545                 }
```

```
546                 call_pass = 1;
547         }

549         if (nkeys > 0)
550                 call_pass = 1;

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

554         if (grps) {
555                 /* redefine login's supplentary group memberships */
556                 ret = edit_group(logname, new_logname, gidlist, 1);
557                 if (ret != EX_SUCCESS) {
558                         errmsg(M_UPDATE, "modified");
559                         exit(ret);
560                 }
561         }
562         if (projects) {
563                 ret = edit_project(logname, (char *)NULL, projlist, 0);
564                 if (ret != EX_SUCCESS) {
565                         errmsg(M_UPDATE, "modified");
566                         exit(ret);
567                 }
568         }


571         if (!call_pass) exit(ret);

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

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

581         if (comment) {  /* comment */
582                 nargv[argindex++] = "-c";
583                 nargv[argindex++] = comment;
584         }

586         if (dir) {
587                 /* flags for home directory */
588                 nargv[argindex++] = "-h";
589                 nargv[argindex++] = dir;
590         }

592         if (group) {
593                 /* set gid flag */
594                 nargv[argindex++] = "-g";
595                 (void) sprintf(gidstring, "%u", gid);
596                 nargv[argindex++] = gidstring;
597         }

599         if (shell) {    /* shell */
600                 nargv[argindex++] = "-s";
601                 nargv[argindex++] = shell;
602         }

604         if (inactstr) {
605                 nargv[argindex++] = "-f";
606                 nargv[argindex++] = inactstr;
607         }

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



```
612         }

614         if (uidstr) {   /* set uid flag */
615                 nargv[argindex++] = "-u";
616                 (void) sprintf(uidstring, "%u", uid);
617                 nargv[argindex++] = uidstring;
618         }

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

622         if (new_logname) {      /* redefine login name */
623                 nargv[argindex++] = "-l";
624                 nargv[argindex++] = new_logname;
625         }

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

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

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

636         /* now call passmgmt */
637         ret = PEX_FAILED;
638         for (tries = 3; ret != PEX_SUCCESS && tries--; ) {
639                 switch (ret = call_passmgmt(nargv)) {
640                 case PEX_SUCCESS:
641                 case PEX_BUSY:
642                         break;

644                 case PEX_HOSED_FILES:
645                         errmsg(M_HOSED_FILES);
646                         exit(EX_INCONSISTENT);
647                         break;

649                 case PEX_SYNTAX:
650                 case PEX_BADARG:
651                         /* should NEVER occur that passmgmt usage is wrong */
652                         if (is_role(usertype))
653                                 errmsg(M_MRUSAGE);
654                         else
655                                 errmsg(M_MUSAGE);
656                         exit(EX_SYNTAX);
657                         break;

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

665                 case PEX_BADNAME:
666                         /* invalid loname */
667                         errmsg(M_USED, logname);
668                         exit(EX_NAME_EXISTS);
669                         break;

671                 default:
672                         errmsg(M_UPDATE, "modified");
673                         exit(ret);
674                         break;
675                 }
676         }
677         if (tries == 0) {
```

```
 678                 errmsg(M_UPDATE, "modified");
 679         }

 681         exit(ret);
 682         /*NOTREACHED*/
 683 }
```

unchanged_portion_omitted_

```
**********************************************************
   45023 Mon Mar 25 12:53:26 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
**********************************************************
```

```
/*
 * 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"
```



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

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

/*
 * 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 %-.12s/%d"
```

```
125 #define USER_LINE \
126 "%6d %-8s %5.5s %5.5s   %3.3s%% %9s %3.3s%%"
127 #define TASK_LINE \
128 "%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"
129 #define PROJECT_LINE \
130 "%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"
131 #define ZONE_LINE \
132 "%6d %8d %5s %5s   %3.3s%% %9s %3.3s%% %28s"

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

137 /* global variables */

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

151 static table_t  pid_tbl = {0, 0, NULL};         /* selected processes */
152 static table_t  cpu_tbl = {0, 0, NULL};         /* selected processors */
153 static table_t  set_tbl = {0, 0, NULL};         /* selected processor sets */
154 static table_t  prj_tbl = {0, 0, NULL};         /* selected projects */
155 static table_t  tsk_tbl = {0, 0, NULL};         /* selected tasks */
156 static table_t  lgr_tbl = {0, 0, NULL};         /* selected lgroups */
157 static zonetbl_t zone_tbl = {0, 0, NULL};       /* selected zones */
158 static uidtbl_t euid_tbl = {0, 0, NULL};        /* selected effective users */
159 static uidtbl_t ruid_tbl = {0, 0, NULL};        /* selected real users */

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

166 static list_t   lwps;                           /* list of lwps/processes */
167 static list_t   users;                          /* list of users */
168 static list_t   tasks;                          /* list of tasks */
169 static list_t   projects;                       /* list of projects */
170 static list_t   zones;                          /* list of zones */
171 static list_t   lgroups;                        /* list of lgroups */

173 static volatile uint_t sigwinch = 0;
174 static volatile uint_t sigtstp = 0;
175 static volatile uint_t sigterm = 0;

177 static long pagesize;

179 /* default settings */

181 static optdesc_t opts = {
182         5,                      /* interval between updates, seconds */
183         15,                     /* number of lines in top part */
184         5,                      /* number of lines in bottom part */
185         -1,                     /* number of iterations; infinitely */
186         OPT_PSINFO | OPT_FULLSCREEN | OPT_USEHOME | OPT_TERMCAP,
187         -1                      /* sort in decreasing order */
188 };
```

_____***unchanged_portion_omitted_***


```
356 /*
357  * A routine to display the contents of the list on the screen
358  */
359 static void
360 list_print(list_t *list)
361 {
362         lwp_info_t *lwp;
363         id_info_t *id;
364         char usr[4], sys[4], trp[4], tfl[4];
365         char dfl[4], lck[4], slp[4], lat[4];
366         char vcx[4], icx[4], scl[4], sig[4];
367         char psize[6], prssize[6], pmem[6], pcpu[6], ptime[12];
368         char pstate[7], pnice[4], ppri[4];
369         char pname[_LOGNAME_MAX+1];
357         char pname[LOGNAME_MAX+1];
370         char projname[PROJNAME_MAX+1];
371         char zonename[ZONENAME_MAX+1];
372         float cpu, mem;
373         double loadavg[3] = {0, 0, 0};
374         int i, lwpid;

376         if (foreach_element(&set_tbl, &loadavg, psetloadavg) == 0) {
377                 /*
378                  * If processor sets aren't specified, we display system-wide
379                  * load averages.
380                  */
381                 (void) getloadavg(loadavg, 3);
382         }

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

391         switch (list->l_type) {
392         case LT_PROJECTS:
393                 if (opts.o_outpmode & OPT_LWPS)
394                         (void) printf(PROJECT_HEADER_LWP);
395                 else
396                         (void) printf(PROJECT_HEADER_PROC);
397                 break;
398         case LT_TASKS:
399                 if (opts.o_outpmode & OPT_LWPS)
400                         (void) printf(TASK_HEADER_LWP);
401                 else
402                         (void) printf(TASK_HEADER_PROC);
403                 break;
404         case LT_ZONES:
405                 if (opts.o_outpmode & OPT_LWPS)
406                         (void) printf(ZONE_HEADER_LWP);
407                 else
408                         (void) printf(ZONE_HEADER_PROC);
409                 break;
410         case LT_USERS:
411                 if (opts.o_outpmode & OPT_LWPS)
412                         (void) printf(USER_HEADER_LWP);
413                 else
414                         (void) printf(USER_HEADER_PROC);
415                 break;
416         case LT_LWPS:
417                 if (opts.o_outpmode & OPT_LWPS) {
418                         if (opts.o_outpmode & OPT_PSINFO) {
419                                 if (opts.o_outpmode & OPT_LGRP)
420                                         (void) printf(PSINFO_HEADER_LWP_LGRP);
```

```
421                                 else
422                                         (void) printf(PSINFO_HEADER_LWP);
423                         }
424                         if (opts.o_outpmode & OPT_MSACCT)
425                                 (void) printf(USAGE_HEADER_LWP);
426                 } else {
427                         if (opts.o_outpmode & OPT_PSINFO) {
428                                 if (opts.o_outpmode & OPT_LGRP)
429                                         (void) printf(PSINFO_HEADER_PROC_LGRP);
430                                 else
431                                         (void) printf(PSINFO_HEADER_PROC);
432                         }
433                         if (opts.o_outpmode & OPT_MSACCT)
434                                 (void) printf(USAGE_HEADER_PROC);
435                 }
436                 break;
437         }

439         (void) putp(t_uloff);
440         (void) putp(t_eol);
441         (void) putchar('\n');

443         for (i = 0; i < list->l_used; i++) {
444                 switch (list->l_type) {
445                 case LT_PROJECTS:
446                 case LT_TASKS:
447                 case LT_USERS:
448                 case LT_ZONES:
449                         id = list->l_ptrs[i];
450                         /*
451                          * CPU usage and memory usage normalization
452                          */
453                         if (total_cpu >= 100)
454                                 cpu = (100 * id->id_pctcpu) / total_cpu;
455                         else
456                                 cpu = id->id_pctcpu;
457                         if (id->id_sizematch == B_FALSE && total_mem >= 100)
458                                 mem = (100 * id->id_pctmem) / total_mem;
459                         else
460                                 mem = id->id_pctmem;
461                         if (list->l_type == LT_USERS) {
462                                 pwd_getname(id->id_uid, pname, sizeof (pname),
463                                     opts.o_outpmode & OPT_NORESOLVE,
464                                     opts.o_outpmode & OPT_TERMCAP,
465                                     LOGIN_WIDTH);
466                         } else if (list->l_type == LT_ZONES) {
449                         if (list->l_type == LT_USERS)
450                                 pwd_getname(id->id_uid, pname, LOGNAME_MAX + 1,
451                                     opts.o_outpmode & OPT_NORESOLVE);
452                         else if (list->l_type == LT_ZONES)
467                                 getzonename(id->id_zoneid, zonename,
468                                     sizeof (zonename) - 1,
469                                     opts.o_outpmode & OPT_TERMCAP,
470                                     ZONE_WIDTH);
471                         } else {
454                                 ZONENAME_MAX);
455                         else
472                                 getprojname(id->id_projid, projname,
473                                     sizeof (projname) - 1,
474                                     opts.o_outpmode & OPT_NORESOLVE,
475                                     opts.o_outpmode & OPT_TERMCAP,
476                                     PROJECT_WIDTH);
477                         }
457                                 PROJNAME_MAX,
458                                 opts.o_outpmode & OPT_NORESOLVE);
478                         Format_size(psize, id->id_size, 6);
```



```
479                         Format_size(prssize, id->id_rssize, 6);
480                         Format_pct(pmem, mem, 4);
481                         Format_pct(pcpu, cpu, 4);
482                         Format_time(ptime, id->id_time, 10);
483                         if (opts.o_outpmode & OPT_TTY)
484                                 (void) putchar('\r');
485                         if (list->l_type == LT_PROJECTS)
486                                 (void) printf(PROJECT_LINE, (int)id->id_projid,
487                                     id->id_nproc, psize, prssize, pmem, ptime,
488                                     pcpu, projname);
489                         else if (list->l_type == LT_TASKS)
490                                 (void) printf(TASK_LINE, (int)id->id_taskid,
491                                     id->id_nproc, psize, prssize, pmem, ptime,
492                                     pcpu, projname);
493                         else if (list->l_type == LT_ZONES)
494                                 (void) printf(ZONE_LINE, (int)id->id_zoneid,
495                                     id->id_nproc, psize, prssize, pmem, ptime,
496                                     pcpu, zonename);
497                         else
498                                 (void) printf(USER_LINE, id->id_nproc, pname,
499                                     psize, prssize, pmem, ptime, pcpu);
500                         (void) putp(t_eol);
501                         (void) putchar('\n');
502                         break;
503                 case LT_LWPS:
504                         lwp = list->l_ptrs[i];
505                         if (opts.o_outpmode & OPT_LWPS)
506                                 lwpid = lwp->li_info.pr_lwp.pr_lwpid;
507                         else
508                                 lwpid = lwp->li_info.pr_nlwp +
509                                     lwp->li_info.pr_nzomb;
510                         pwd_getname(lwp->li_info.pr_uid, pname, sizeof (pname),
511                             opts.o_outpmode & OPT_NORESOLVE,
512                             opts.o_outpmode & OPT_TERMCAP,
513                             LOGIN_WIDTH);
491                         pwd_getname(lwp->li_info.pr_uid, pname, LOGNAME_MAX + 1,
492                             opts.o_outpmode & OPT_NORESOLVE);
514                         if (opts.o_outpmode & OPT_PSINFO) {
515                                 Format_size(psize, lwp->li_info.pr_size, 6);
516                                 Format_size(prssize, lwp->li_info.pr_rssize, 6);
517                                 Format_state(pstate,
518                                     lwp->li_info.pr_lwp.pr_sname,
519                                     lwp->li_info.pr_lwp.pr_onpro, 7);
520                                 if (strcmp(lwp->li_info.pr_lwp.pr_clname,
521                                     "RT") == 0 ||
522                                     strcmp(lwp->li_info.pr_lwp.pr_clname,
523                                     "SYS") == 0 ||
524                                     lwp->li_info.pr_lwp.pr_sname == 'Z')
525                                         (void) strcpy(pnice, "  -");
526                                 else
527                                         Format_num(pnice,
528                                             lwp->li_info.pr_lwp.pr_nice - NZERO,
529                                             4);
530                                 Format_num(ppri, lwp->li_info.pr_lwp.pr_pri, 4);
531                                 Format_pct(pcpu,
532                                     FRC2PCT(lwp->li_info.pr_lwp.pr_pctcpu), 4);
533                                 if (opts.o_outpmode & OPT_LWPS)
534                                         Format_time(ptime,
535                                             lwp->li_info.pr_lwp.pr_time.tv_sec,
536                                             10);
537                                 else
538                                         Format_time(ptime,
539                                             lwp->li_info.pr_time.tv_sec, 10);
540                                 if (opts.o_outpmode & OPT_TTY)
541                                         (void) putchar('\r');
542                                 stripfname(lwp->li_info.pr_fname);
```

```
543                                 if (opts.o_outpmode & OPT_LGRP) {
544                                         (void) printf(PSINFO_LINE_LGRP,
545                                             (int)lwp->li_info.pr_pid, pname,
546                                             psize, prssize, pstate,
547                                             ppri, pnice, ptime, pcpu,
525                                             psize, prssize, pstate, ppri, pnice,
526                                             ptime, pcpu,
548                                             (int)lwp->li_info.pr_lwp.pr_lgrp,
549                                             lwp->li_info.pr_fname, lwpid);
550                                 } else {
551                                         (void) printf(PSINFO_LINE,
552                                             (int)lwp->li_info.pr_pid, pname,
553                                             psize, prssize,
554                                             pstate, ppri, pnice,
532                                             psize, prssize, pstate, ppri, pnice,
555                                             ptime, pcpu,
556                                             lwp->li_info.pr_fname, lwpid);
557                                 }
558                                 (void) putp(t_eol);
559                                 (void) putchar('\n');
560                         }
561                         if (opts.o_outpmode & OPT_MSACCT) {
562                                 Format_pct(usr, lwp->li_usr, 4);
563                                 Format_pct(sys, lwp->li_sys, 4);
564                                 Format_pct(slp, lwp->li_slp, 4);
565                                 Format_num(vcx, lwp->li_vcx, 4);
566                                 Format_num(icx, lwp->li_icx, 4);
567                                 Format_num(scl, lwp->li_scl, 4);
568                                 Format_num(sig, lwp->li_sig, 4);
569                                 Format_pct(trp, lwp->li_trp, 4);
570                                 Format_pct(tfl, lwp->li_tfl, 4);
571                                 Format_pct(dfl, lwp->li_dfl, 4);
572                                 Format_pct(lck, lwp->li_lck, 4);
573                                 Format_pct(lat, lwp->li_lat, 4);
574                                 if (opts.o_outpmode & OPT_TTY)
575                                         (void) putchar('\r');
576                                 stripfname(lwp->li_info.pr_fname);
577                                 (void) printf(USAGE_LINE,
578                                     (int)lwp->li_info.pr_pid, pname,
579                                     usr, sys, trp, tfl, dfl, lck,
580                                     slp, lat, vcx, icx, scl, sig,
581                                     lwp->li_info.pr_fname, lwpid);
582                                 (void) putp(t_eol);
583                                 (void) putchar('\n');
584                         }
585                         break;
586                 }
587         }

589         if (opts.o_outpmode & OPT_TTY)
590                 (void) putchar('\r');
591         if (opts.o_outpmode & OPT_TERMCAP) {
592                 switch (list->l_type) {
593                 case LT_PROJECTS:
594                 case LT_USERS:
595                 case LT_TASKS:
596                 case LT_ZONES:
597                         while (i++ < opts.o_nbottom) {
598                                 (void) putp(t_eol);
599                                 (void) putchar('\n');
600                         }
601                         break;
602                 case LT_LWPS:
603                         while (i++ < opts.o_ntop) {
604                                 (void) putp(t_eol);
605                                 (void) putchar('\n');
```



```
606                         }
607                 }
608         }

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

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

616         (void) printf(TOTAL_LINE, total_procs, total_lwps,
617             loadavg[LOADAVG_1MIN], loadavg[LOADAVG_5MIN],
618             loadavg[LOADAVG_15MIN]);
619         (void) putp(t_eol);
620         (void) putchar('\n');
621         if (opts.o_outpmode & OPT_TTY)
622                 (void) putchar('\r');
623         (void) putp(t_eol);
624         (void) fflush(stdout);
625 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    6709 Mon Mar 25 12:53:26 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 Mon Mar 25 12:53:26 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 Mon Mar 25 12:53:26 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 Mon Mar 25 12:53:26 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_
```

```
**********************************************************
   58128 Mon Mar 25 12:53:26 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

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

/* 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_
```

```
**********************************************************
    5514 Mon Mar 25 12:53:26 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>

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

#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] == '-')  {
                        /*
```

```
125                          * Should set flag here to allow special case checking
126                          * in the rest of the code,
127                          * but for now, we'll just ignore this entry.
128                          */
129                         continue;
130                 }

132                 /* Check number of fields */

134                 for (i = 0; buf[i] != NULL; i++)
135                         if (buf[i] == ':') {
136                                 delim[colons] = i;
137                                 ++colons;
138                         }

140                 if (colons != 6) {
141                         error(ERROR1);
142                         continue;
143                 }
144                 delim[6] = i - 1;
145                 delim[7] = NULL;

147                 /*
148                  * Check the first char is alpha; the rest alphanumeric;
149                  * and that the name does not consist solely of uppercase
150                  * alpha chars
151                  */
152                 if (buf[0] == ':')
153                         error(ERROR2b);
154                 else if (!isalpha(buf[0]))
155                         error(ERROR2a);

157                 for (i = 0; buf[i] != ':'; i++) {
158                         if (!isalnum(buf[i]) &&
159                                 buf[i] != '_' &&
160                                 buf[i] != '-' &&
161                                 buf[i] != '.')
162                                 badc++;
163                         else if (islower(buf[i]))
164                                 lc++;
165                 }
166                 if (lc == 0)
167                         error(ERROR2c);
168                 if (badc > 0)
169                         error(ERROR2);

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

173                 if (i <= 0 || i > _LOGNAME_MAX)
166                 if (i <= 0 || i > 8)
174                         error(ERROR3);

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

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

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

187                 errno = 0;
188                 str = &buf[delim[2] + 1];
189                 gid = strtol(str, &lastc, 10);
```



```
190                 if (lastc != str + (delim[3] - delim[2]) - 1 ||
191                     gid > MAXUID || errno == ERANGE)
192                         error(ERROR5);

194                 /* Check initial working directory */

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

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

205                 /* Check program to use as shell  */

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

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

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

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

219                         for (j = 0; j < 512; j++)
220                                 logbuf[j] = NULL;
221                 }
222         }
223         (void) fclose(fptr);
224         return (code);
225 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   57376 Mon Mar 25 12:53:27 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>

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

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
```

```
 128 #define SUPATH  "/usr/bin/su"
 129 #define FAILSAFESHELL   "/sbin/sh"
 130 #define DEFAULTSHELL    "/sbin/sh"
 131 #define DEF_PATH        "/usr/sbin:/usr/bin"

 133 #define CLUSTER_BRAND_NAME      "cluster"

 135 /*
 136  * The ZLOGIN_BUFSIZ is larger than PIPE_BUF so we can be sure we're clearing
 137  * out the pipe when the child is exiting.  The ZLOGIN_RDBUFSIZ must be less
 138  * than ZLOGIN_BUFSIZ (because we share the buffer in doio).  This value is
 139  * also chosen in conjunction with the HI_WATER setting to make sure we
 140  * don't fill up the pipe.  We can write FIFOHIWAT (16k) into the pipe before
 141  * blocking.  By having ZLOGIN_RDBUFSIZ set to 1k and HI_WATER set to 8k, we
 142  * know we can always write a ZLOGIN_RDBUFSIZ chunk into the pipe when there
 143  * is less than HI_WATER data already in the pipe.
 144  */
 145 #define ZLOGIN_BUFSIZ   8192
 146 #define ZLOGIN_RDBUFSIZ 1024
 147 #define HI_WATER        8192

 149 /*
 150  * See canonify() below.  CANONIFY_LEN is the maximum length that a
 151  * "canonical" sequence will expand to (backslash, three octal digits, NUL).
 152  */
 153 #define CANONIFY_LEN 5

 155 static void
 156 usage(void)
 157 {
 158         (void) fprintf(stderr, gettext("usage: %s [ -CES ] [ -e cmdchar ] "
 159             "[-l user] zonename [command [args ...] ]\n"), pname);
 160         exit(2);
 161 }
_____unchanged_portion_omitted_

1235 /*
1236  * Finish the preparation of the envp array for exec'd non-interactive
1237  * zlogins.  This is called in the child process *after* we zone_enter(), since
1238  * it derives things we can only know within the zone, such as $HOME, $SHELL,
1239  * etc.  We need only do this in the non-interactive, mode, since otherwise
1240  * login(1) will do it.  We don't do this in failsafe mode, since it presents
1241  * additional ways in which the command could fail, and we'd prefer to avoid
1242  * that.
1243  */
1244 static char **
1245 prep_env_noninteractive(const char *user_cmd, char **env)
1246 {
1247         size_t size;
1248         char **new_env;
1249         int e, i;
1250         char *estr;
1251         char varmail[_LOGNAME_MAX + 11]; /* strlen(/var/mail/) = 10, NUL */
1243         char varmail[LOGNAME_MAX + 11]; /* strlen(/var/mail/) = 10, NUL */
1252         char pwbuf[NSS_BUFLEN_PASSWD + 1];
1253         struct passwd pwent;
1254         struct passwd *pw = NULL;

1256         assert(env != NULL);
1257         assert(failsafe == 0);

1259         /*
1260          * Exec the "user_cmd" brand hook to get a pwent for the
1261          * login user.  If this fails, HOME will be set to "/", SHELL
1262          * will be set to $DEFAULTSHELL, and we will continue to exec
```



```
1263          * SUPATH <login> -c <cmd>.
1264          */
1265         pw = zone_get_user_pw(user_cmd, &pwent, pwbuf, sizeof (pwbuf));

1267         /*
1268          * Get existing envp size.
1269          */
1270         for (size = 0; env[size] != NULL; size++)
1271                 ;

1273         e = size;

1275         /*
1276          * Finish filling out the environment; we duplicate the environment
1277          * setup described in login(1), for lack of a better precedent.
1278          */
1279         if (pw != NULL)
1280                 size += 3;      /* LOGNAME, HOME, MAIL */
1281         else
1282                 size += 1;      /* HOME */

1284         size++; /* always fill in SHELL */
1285         size++; /* terminating NULL */

1287         if ((new_env = malloc(sizeof (char *) * size)) == NULL)
1288                 goto malloc_fail;

1290         /*
1291          * Copy existing elements of env into new_env.
1292          */
1293         for (i = 0; env[i] != NULL; i++) {
1294                 if ((new_env[i] = strdup(env[i])) == NULL)
1295                         goto malloc_fail;
1296         }
1297         assert(e == i);

1299         if (pw != NULL) {
1300                 if ((estr = add_env("LOGNAME", pw->pw_name)) == NULL)
1301                         goto malloc_fail;
1302                 new_env[e++] = estr;

1304                 if ((estr = add_env("HOME", pw->pw_dir)) == NULL)
1305                         goto malloc_fail;
1306                 new_env[e++] = estr;

1308                 if (chdir(pw->pw_dir) != 0)
1309                         zerror(gettext("Could not chdir to home directory "
1310                             "%s: %s"), pw->pw_dir, strerror(errno));

1312                 (void) snprintf(varmail, sizeof (varmail), "/var/mail/%s",
1313                     pw->pw_name);
1314                 if ((estr = add_env("MAIL", varmail)) == NULL)
1315                         goto malloc_fail;
1316                 new_env[e++] = estr;
1317         } else {
1318                 if ((estr = add_env("HOME", "/")) == NULL)
1319                         goto malloc_fail;
1320                 new_env[e++] = estr;
1321         }

1323         if (pw != NULL && strlen(pw->pw_shell) > 0) {
1324                 if ((estr = add_env("SHELL", pw->pw_shell)) == NULL)
1325                         goto malloc_fail;
1326                 new_env[e++] = estr;
1327         } else {
1328                 if ((estr = add_env("SHELL", DEFAULTSHELL)) == NULL)
```

```
1329                        goto malloc_fail;
1330                new_env[e++] = estr;
1331        }

1333        new_env[e++] = NULL;    /* add terminating NULL */

1335        assert(e == size);
1336        return (new_env);

1338 malloc_fail:
1339        zperror(gettext("failed to allocate memory for process environment"));
1340        return (NULL);
1341 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   10497 Mon Mar 25 12:53:27 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
```

```
125 #define _POSIX_PATH_MAX           255
126 #endif
127 #define _POSIX_PIPE_BUF           512
128 #define _POSIX_RTSIG_MAX            8
129 #define _POSIX_SEM_NSEMS_MAX      256
130 #define _POSIX_SEM_VALUE_MAX    32767
131 #define _POSIX_SIGQUEUE_MAX        32
132 #define _POSIX_SSIZE_MAX        32767
133 #define _POSIX_STREAM_MAX           8
134 #define _POSIX_TIMER_MAX           32
135 #ifdef _XPG6
136 #define _POSIX_TZNAME_MAX           6
137 #else
138 #define _POSIX_TZNAME_MAX           3   /* POSIX.1-1990 default */
139 #endif
140 /* POSIX.1c conformant */
141 #define _POSIX_LOGIN_NAME_MAX                   9
142 #define _POSIX_THREAD_DESTRUCTOR_ITERATIONS     4
143 #define _POSIX_THREAD_KEYS_MAX                128
144 #define _POSIX_THREAD_THREADS_MAX              64
145 #define _POSIX_TTY_NAME_MAX                    9
146 /* UNIX 03 conformant */
147 #define _POSIX_HOST_NAME_MAX                   255
148 #define _POSIX_RE_DUP_MAX                      255
149 #define _POSIX_SYMLINK_MAX                     255
150 #define _POSIX_SYMLOOP_MAX                     8

152 /*
153  * POSIX.2 and XPG4-XSH4 conformant definitions
154  */

156 #define _POSIX2_BC_BASE_MAX              99
157 #define _POSIX2_BC_DIM_MAX             2048
158 #define _POSIX2_BC_SCALE_MAX             99
159 #define _POSIX2_BC_STRING_MAX          1000
160 #define _POSIX2_COLL_WEIGHTS_MAX          2
161 #define _POSIX2_EXPR_NEST_MAX            32
162 #define _POSIX2_LINE_MAX               2048
163 #define _POSIX2_RE_DUP_MAX              255
164 /* UNIX 03 conformant */
165 #define _POSIX2_CHARCLASS_NAME_MAX       14

167 #define BC_BASE_MAX             _POSIX2_BC_BASE_MAX
168 #define BC_DIM_MAX              _POSIX2_BC_DIM_MAX
169 #define BC_SCALE_MAX            _POSIX2_BC_SCALE_MAX
170 #define BC_STRING_MAX           _POSIX2_BC_STRING_MAX
171 #define COLL_WEIGHTS_MAX        10
172 #define EXPR_NEST_MAX           _POSIX2_EXPR_NEST_MAX
173 #define LINE_MAX                _POSIX2_LINE_MAX
174 #if !defined(_XPG6)
175 #define RE_DUP_MAX              _POSIX2_RE_DUP_MAX
176 #else
177 #define RE_DUP_MAX              _POSIX_RE_DUP_MAX
178 #endif /* !defined(_XPG6) */

180 #endif /* defined(__EXTENSIONS__) || !defined(_STRICT_STDC) ... */

182 #if defined(__EXTENSIONS__) || \
183         (!defined(_STRICT_STDC) && !defined(_POSIX_C_SOURCE)) || \
184         defined(_XOPEN_SOURCE)

186 /*
187  * For dual definitions for PASS_MAX and sysconf.c
188  */
189 #define _PASS_MAX_XPG   8       /* old standards PASS_MAX */
190 #define _PASS_MAX       256     /* modern Solaris PASS_MAX */
```



```
192 #if defined(_XPG3) && !defined(_XPG6)
193 #define PASS_MAX        _PASS_MAX_XPG   /* max # of characters in a password */
194 #else   /* XPG6 or just Solaris */
195 #define PASS_MAX        _PASS_MAX       /* max # of characters in a password */
196 #endif  /* defined(_XPG3) && !defined(_XPG6) */

198 #define CHARCLASS_NAME_MAX      _POSIX2_CHARCLASS_NAME_MAX

200 #define NL_ARGMAX       9       /* max value of "digit" in calls to the */
201                                 /* NLS printf() and scanf() */
202 #define NL_LANGMAX      14      /* max # of bytes in a LANG name */
203 #define NL_MSGMAX       32767   /* max message number */
204 #define NL_NMAX         1       /* max # bytes in N-to-1 mapping characters */
205 #define NL_SETMAX       255     /* max set number */
206 #define NL_TEXTMAX      2048    /* max set number */
207 #define NZERO           20      /* default process priority */

209 #define WORD_BIT        32      /* # of bits in a "word" or "int" */
210 #if defined(_LP64)
211 #define LONG_BIT        64      /* # of bits in a "long" */
212 #else   /* _ILP32 */
213 #define LONG_BIT        32      /* # of bits in a "long" */
214 #endif

216 /* Marked as LEGACY in SUSv2 and removed in UNIX 03 */
217 #ifndef _XPG6
218 #define DBL_DIG         15      /* digits of precision of a "double" */
219 #define DBL_MAX         1.79769313486231570814527E+308   /* max decimal value */
220                                                          /* of a double */
221 #define FLT_DIG         6               /* digits of precision of a "float" */
222 #define FLT_MAX         3.40282346638528859811704E+38F   /* max decimal value */
223                                                          /* of a "float" */
224 #endif

226 /* Marked as LEGACY in SUSv1 and removed in SUSv2 */
227 #ifndef _XPG5
228 #define DBL_MIN         2.22507385850720138309023E-308   /* min decimal value */
229                                                          /* of a double */
230 #define FLT_MIN         1.17549435082228750796882E-38F   /* min decimal value */
231                                                          /* of a float */
232 #endif

234 #endif  /* defined(__EXTENSIONS__) || (!defined(_STRICT_STDC) ... */

236 #define _XOPEN_IOV_MAX  16      /* max # iovec/process with readv()/writev() */
237 #define _XOPEN_NAME_MAX 255     /* max # bytes in filename excluding null */
238 #define _XOPEN_PATH_MAX 1024    /* max # bytes in a pathname */

240 #define IOV_MAX         _XOPEN_IOV_MAX

242 #if defined(__EXTENSIONS__) || \
243         (!defined(_STRICT_STDC) && !defined(__XOPEN_OR_POSIX))

245 #define FCHR_MAX        1048576         /* max size of a file in bytes */
246 #define PID_MAX         999999          /* max value for a process ID */

248 /*
249  * POSIX 1003.1a, section 2.9.5, table 2-5 contains [NAME_MAX] and the
250  * related text states:
251  *
252  * A definition of one of the values from Table 2-5 shall be omitted from the
253  * <limits.h> on specific implementations where the corresponding value is
254  * equal to or greater than the stated minimum, but where the value can vary
255  * depending on the file to which it is applied. The actual value supported for
256  * a specific pathname shall be provided by the pathconf() (5.7.1) function.
```

```
 257  *
 258  * This is clear that any machine supporting multiple file system types
 259  * and/or a network can not include this define, regardless of protection
 260  * by the _POSIX_SOURCE and _POSIX_C_SOURCE flags.
 261  *
 262  * #define      NAME_MAX        14
 263  */

 265 #define CHILD_MAX       25      /* max # of processes per user id */
 266 #ifndef OPEN_MAX
 267 #define OPEN_MAX        256     /* max # of files a process can have open */
 268 #endif

 270 #define PIPE_MAX        5120    /* max # bytes written to a pipe in a write */

 272 #define STD_BLK         1024    /* # bytes in a physical I/O block */
 273 #define UID_MAX         2147483647      /* max value for a user or group ID */
 274 #define USI_MAX         4294967295      /* max decimal value of an "unsigned" */
 275 #define SYSPID_MAX      1       /* max pid of system processes */

 277 #ifndef SYS_NMLN                /* also defined in sys/utsname.h */
 278 #define SYS_NMLN        257     /* 4.0 size of utsname elements */
 279 #endif

 281 #ifndef CLK_TCK

 283 #if !defined(_CLOCK_T) || __cplusplus >= 199711L
 284 #define _CLOCK_T
 285 typedef long    clock_t;
 286 #endif  /* !_CLOCK_T */

 288 extern long _sysconf(int);      /* System Private interface to sysconf() */
 289 #define CLK_TCK ((clock_t)_sysconf(3))  /* 3 is _SC_CLK_TCK */

 291 #endif /* CLK_TCK */

 293 #define LOGNAME_MAX     8       /* max # of characters in a login name */
 294 #define LOGNAME_MAX_ILLUMOS     32      /* max # of characters in an */
 295                                         /* illumos login name */
 296 #define TTYNAME_MAX     128     /* max # of characters in a tty name */

 298 #endif  /* if defined(__EXTENSIONS__) || (!defined(_STRICT_STDC) ... */

 300 #if     defined(__EXTENSIONS__) || (_POSIX_C_SOURCE >= 199506L)
 301 #include <sys/unistd.h>

 303 #if !defined(_SIZE_T) || __cplusplus >= 199711L
 304 #define _SIZE_T
 305 #if defined(_LP64) || defined(_I32LPx)
 306 typedef unsigned long size_t;   /* size of something in bytes */
 307 #else
 308 typedef unsigned int  size_t;   /* (historical version) */
 309 #endif
 310 #endif  /* _SIZE_T */

 312 extern long _sysconf(int);      /* System Private interface to sysconf() */

 314 #define PTHREAD_STACK_MIN       ((size_t)_sysconf(_SC_THREAD_STACK_MIN))
 315 /* Added for UNIX98 conformance */
 316 #define PTHREAD_DESTRUCTOR_ITERATIONS   _POSIX_THREAD_DESTRUCTOR_ITERATIONS
 317 #define PTHREAD_KEYS_MAX                _POSIX_THREAD_KEYS_MAX
 318 #define PTHREAD_THREADS_MAX             _POSIX_THREAD_THREADS_MAX
 319 #endif  /* defined(__EXTENSIONS__) || (_POSIX_C_SOURCE >= 199506L) */

 321 #ifdef  __cplusplus
 322 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   29196 Mon Mar 25 12:53:27 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
```

```
127 #define NSS_DEFCONF_PASSWD_COMPAT       NSS_NS_ONLY

129 #define NSS_DEFCONF_ATTRDB      NSS_FILES_NS

131 #define NSS_DEFCONF_AUDITUSER   NSS_DEFCONF_PASSWD
132 #define NSS_DEFCONF_USERATTR    NSS_DEFCONF_PASSWD
133 #define NSS_DEFCONF_AUTHATTR    NSS_DEFCONF_ATTRDB
134 #define NSS_DEFCONF_PROFATTR    NSS_DEFCONF_ATTRDB
135 #define NSS_DEFCONF_EXECATTR    NSS_DEFCONF_PROFATTR

137 #define NSS_DEFCONF_TSOL_TP     NSS_TSOL_FALLBACK
138 #define NSS_DEFCONF_TSOL_RH     NSS_TSOL_FALLBACK
139 #define NSS_DEFCONF_TSOL_ZC     NSS_TSOL_FALLBACK

141 /*
142  * Line-lengths that the "files" and "compat" backends will try to support.
143  * It may be reasonable (even advisable) to use smaller values than these.
144  */

146 #define NSS_BUFSIZ              1024

148 #define NSS_LINELEN_GROUP       ((NSS_BUFSIZ) * 8)
149 #define NSS_LINELEN_HOSTS       ((NSS_BUFSIZ) * 8)
150 #define NSS_LINELEN_IPNODES     ((NSS_BUFSIZ) * 8)
151 #define NSS_LINELEN_NETMASKS    NSS_BUFSIZ
152 #define NSS_LINELEN_NETWORKS    NSS_BUFSIZ
153 #define NSS_LINELEN_PASSWD      NSS_BUFSIZ
154 #define NSS_LINELEN_PRINTERS    NSS_BUFSIZ
155 #define NSS_LINELEN_PROJECT     ((NSS_BUFSIZ) * 4)
156 #define NSS_LINELEN_PROTOCOLS   NSS_BUFSIZ
157 #define NSS_LINELEN_PUBLICKEY   NSS_BUFSIZ
158 #define NSS_LINELEN_RPC         NSS_BUFSIZ
159 #define NSS_LINELEN_SERVICES    NSS_BUFSIZ
160 #define NSS_LINELEN_SHADOW      NSS_BUFSIZ
161 #define NSS_LINELEN_ETHERS      NSS_BUFSIZ
162 #define NSS_LINELEN_BOOTPARAMS  NSS_BUFSIZ

164 #define NSS_LINELEN_ATTRDB      NSS_BUFSIZ

166 #define NSS_LINELEN_AUDITUSER   NSS_LINELEN_ATTRDB
167 #define NSS_LINELEN_AUTHATTR    NSS_LINELEN_ATTRDB
168 #define NSS_LINELEN_EXECATTR    NSS_LINELEN_ATTRDB
169 #define NSS_LINELEN_PROFATTR    NSS_LINELEN_ATTRDB
170 #define NSS_LINELEN_USERATTR    NSS_LINELEN_ATTRDB

172 #define NSS_MMAPLEN_EXECATTR    NSS_LINELEN_EXECATTR * 8

174 #define NSS_LINELEN_TSOL        NSS_BUFSIZ

176 #define NSS_LINELEN_TSOL_TP     NSS_LINELEN_TSOL
177 #define NSS_LINELEN_TSOL_RH     NSS_LINELEN_TSOL
178 #define NSS_LINELEN_TSOL_ZC     NSS_LINELEN_TSOL

180 /*
181  * Reasonable defaults for 'buflen' values passed to _r functions.  The BSD
182  * and SunOS 4.x implementations of the getXXXbyYYY() functions used hard-
183  * coded array sizes;  the values here are meant to handle anything that
184  * those implementations handled.
185  * === These might more reasonably go in <pwd.h>, <netdb.h> et al
186  */

188 #define NSS_BUFLEN_GROUP        NSS_LINELEN_GROUP
189 #define NSS_BUFLEN_HOSTS        \
190         (NSS_LINELEN_HOSTS + (MAXALIASES + MAXADDRS + 2) * sizeof (char *))
191 #define NSS_BUFLEN_IPNODES      \
192         (NSS_LINELEN_IPNODES + (MAXALIASES + MAXADDRS + 2) * sizeof (char *))
```



```
193 #ifdef  LOGNAME_MAX_ILLUMOS
194 #define NSS_BUFLEN_NETGROUP     (MAXHOSTNAMELEN * 2 + LOGNAME_MAX_ILLUMOS + 3)
195 #else /* LOGNAME_MAX_ILLUMOS */
196 #define NSS_BUFLEN_NETGROUP     (MAXHOSTNAMELEN * 2 + LOGNAME_MAX + 3)
197 #endif /* LOGNAME_MAX_ILLUMOS */
198 #define NSS_BUFLEN_NETWORKS     NSS_LINELEN_NETWORKS    /* === ?  + 35 * 4 */
199 #define NSS_BUFLEN_PASSWD       NSS_LINELEN_PASSWD
200 #define NSS_BUFLEN_PROJECT      (NSS_LINELEN_PROJECT + 800 * sizeof (char *))
201 #define NSS_BUFLEN_PROTOCOLS    NSS_LINELEN_PROTOCOLS   /* === ?  + 35 * 4 */
202 #define NSS_BUFLEN_PUBLICKEY    NSS_LINELEN_PUBLICKEY
203 #define NSS_BUFLEN_RPC          NSS_LINELEN_RPC         /* === ?  + 35 * 4 */
204 #define NSS_BUFLEN_SERVICES     NSS_LINELEN_SERVICES    /* === ?  + 35 * 4 */
205 #define NSS_BUFLEN_SHADOW       NSS_LINELEN_SHADOW
206 #define NSS_BUFLEN_ETHERS       NSS_LINELEN_ETHERS
207 #define NSS_BUFLEN_BOOTPARAMS   NSS_LINELEN_BOOTPARAMS

209 #define NSS_BUFLEN_ATTRDB       NSS_LINELEN_ATTRDB

211 #define NSS_BUFLEN_AUDITUSER    NSS_BUFLEN_ATTRDB
212 #define NSS_BUFLEN_AUTHATTR     NSS_BUFLEN_ATTRDB
213 #define NSS_BUFLEN_EXECATTR     NSS_BUFLEN_ATTRDB
214 #define NSS_BUFLEN_PROFATTR     NSS_BUFLEN_ATTRDB
215 #define NSS_BUFLEN_USERATTR     ((NSS_BUFLEN_ATTRDB) * 8)

217 #define NSS_BUFLEN_TSOL         NSS_LINELEN_TSOL
218 #define NSS_BUFLEN_TSOL_TP      NSS_BUFLEN_TSOL
219 #define NSS_BUFLEN_TSOL_RH      NSS_BUFLEN_TSOL
220 #define NSS_BUFLEN_TSOL_ZC      NSS_BUFLEN_TSOL

222 /*
223  * Default cache door buffer size (2x largest buffer)
224  */

226 #define NSS_BUFLEN_DOOR         ((NSS_BUFSIZ) * 16)

228 /*
229  * Arguments and results, passed between the frontends and backends for
230  * the well-known databases.  The getXbyY_r() and getXent_r() routines
231  * use a common format that is further described below;  other routines
232  * use their own formats.
233  */

235 /*
236  * The nss_str2ent_t routine is the data marshaller for the nsswitch.
237  * it converts 'native files' format into 'entry' format as part of the
238  * return processing for a getXbyY interface.
239  *
240  * The nss_groupstr_t routine does the real work for any backend
241  * that can supply a netgroup entry as a string in /etc/group format
242  */
243 #if defined(__STDC__)
244 typedef int             (*nss_str2ent_t)(const char *in, int inlen,
245                                 void *ent, char *buf, int buflen);

247 struct nss_groupsbymem;         /* forward definition */
248 typedef nss_status_t    (*nss_groupstr_t)(const char *instr, int inlen,
249                                 struct nss_groupsbymem *);
250 #else
251 typedef int             (*nss_str2ent_t)();
252 typedef nss_status_t    (*nss_groupstr_t)();
253 #endif

255 /*
256  * The initgroups() function [see initgroups(3c)] needs to find all the
257  *   groups to which a given user belongs.  To do this it calls
258  *   _getgroupsbymember(), which is part of the frontend for the "group"
```

```
 259  *   database.
 260  * We want the same effect as if we used getgrent_r() to enumerate the
 261  *   entire groups database (possibly from multiple sources), but getgrent_r()
 262  *   is too inefficient.  Most backends can do better if they know they're
 263  *   meant to scan all groups;  hence there's a separate backend operation,
 264  *   NSS_DBOP_GROUP_BYMEMBER, which uses the nss_groupsbymem struct.
 265  * Note that the normal return-value from such a backend, even when it
 266  *   successfully finds matching group entries, is NSS_NOTFOUND, because
 267  *   this tells the switch engine to keep searching in any more sources.
 268  *   In fact, the backends only return NSS_SUCCESS if they find enough
 269  *   matching entries that the gid_array is completely filled, in which
 270  *   case the switch engine should stop searching.
 271  * If the force_slow_way field is set, the backend should eschew any cached
 272  *   information (e.g. the YP netid.byname map or the NIS+ cred.org_dir table)
 273  *   and should instead grind its way through the group map/table/whatever.
 274  */

 276 struct nss_groupsbymem {                        /* For _getgroupsbymember() */
 277 /* in: */
 278         const char      *username;
 279         gid_t           *gid_array;
 280         int             maxgids;
 281         int             force_slow_way;
 282         nss_str2ent_t   str2ent;
 283         nss_groupstr_t  process_cstr;

 285 /* in_out: */
 286         int             numgids;
 287 };
```

unchanged_portion_omitted_

```
**********************************************************
    6618 Mon Mar 25 12:53:27 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)

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



```
#endif /* LOGNAME_MAX_ILLUMOS */
static char             luser[LOGNAME_MAX + 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);
        (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);
        (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);
        (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);
        (void) strncpy(luser, uname, LOGNAME_MAX);
        generate_record(luser, 0, "");
}
______unchanged_portion_omitted_
```

```
*********************************************************
   23535 Mon Mar 25 12:53:27 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
#ifdef  LOGNAME_MAX_ILLUMOS
#define MAX_TRIPLE_LEN          (MAXHOSTNAMELEN + LOGNAME_MAX_ILLUMOS + \
                                        MAX_DOMAIN_LEN + 5)
#else /* LOGNAME_MAX_ILLUMOS */
#define MAX_TRIPLE_LEN          (MAXHOSTNAMELEN + LOGNAME_MAX + \
                                        MAX_DOMAIN_LEN + 5)
#endif /* LOGNAME_MAX_ILLUMOS */

#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 Mon Mar 25 12:53:27 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.
```

```
391 .RE

393 .SH OUTPUT
394 .sp
395 .LP
396 The following list defines the column headings and the meanings of a
397 \fBprstat\fR report:
398 .sp
399 .ne 2
400 .na
401 \fBPID\fR
402 .ad
403 .sp .6
404 .RS 4n
405 The process \fBID\fR of the process.
406 .RE

408 .sp
409 .ne 2
410 .na
411 \fBUSERNAME\fR
412 .ad
413 .sp .6
414 .RS 4n
415 The real user (login) name or real user \fBID\fR.
416 A trailing asterisk marks a long name that has been truncated
417 to fit the column.
418 .RE

420 .sp
421 .ne 2
422 .na
423 \fBSWAP\fR
424 .ad
425 .sp .6
426 .RS 4n
427 The total virtual memory size of the process, including all mapped files and
428 devices, in kilobytes (\fBK\fR), megabytes (\fBM\fR), or gigabytes (\fBG\fR).
429 .RE

431 .sp
432 .ne 2
433 .na
434 \fBRSS\fR
435 .ad
436 .sp .6
437 .RS 4n
438 The resident set size of the process (\fBRSS\fR), in kilobytes (\fBK\fR),
439 megabytes (\fBM\fR), or gigabytes (\fBG\fR). The RSS value is an estimate
440 provided by \fBproc\fR(4) that might underestimate the actual resident set
441 size. Users who want to get more accurate usage information for capacity
442 planning should use the \fB-x\fR option to \fBpmap\fR(1) instead.
443 .RE

445 .sp
446 .ne 2
447 .na
448 \fBSTATE\fR
449 .ad
450 .sp .6
451 .RS 4n
452 The state of the process:
453 .sp
454 .ne 2
455 .na
456 \fBcpu\fIN\fR\fR
```



```
457 .ad
458 .sp .6
459 .RS 4n
460 Process is running on \fBCPU\fR \fIN\fR.
461 .RE

463 .sp
464 .ne 2
465 .na
466 \fBsleep\fR
467 .ad
468 .sp .6
469 .RS 4n
470 Sleeping: process is waiting for an event to complete.
471 .RE

473 .sp
474 .ne 2
475 .na
476 \fBwait\fR
477 .ad
478 .sp .6
479 .RS 4n
480 Waiting: process is waiting for CPU usage to drop to the CPU-caps enforced
481 limits. See the description of \fBCPU-caps\fR in \fBresource_controls\fR(5).
482 .RE

484 .sp
485 .ne 2
486 .na
487 \fBrun\fR
488 .ad
489 .sp .6
490 .RS 4n
491 Runnable: process in on run queue.
492 .RE

494 .sp
495 .ne 2
496 .na
497 \fBzombie\fR
498 .ad
499 .sp .6
500 .RS 4n
501 Zombie state: process terminated and parent not waiting.
502 .RE

504 .sp
505 .ne 2
506 .na
507 \fBstop\fR
508 .ad
509 .sp .6
510 .RS 4n
511 Process is stopped.
512 .RE

514 .RE

516 .sp
517 .ne 2
518 .na
519 \fBPRI\fR
520 .ad
521 .sp .6
522 .RS 4n
```

```
523 The priority of the process. Larger numbers mean higher priority.
524 .RE

526 .sp
527 .ne 2
528 .na
529 \fBNICE\fR
530 .ad
531 .sp .6
532 .RS 4n
533 Nice value used in priority computation. Only processes in certain scheduling
534 classes have a nice value.
535 .RE

537 .sp
538 .ne 2
539 .na
540 \fBTIME\fR
541 .ad
542 .sp .6
543 .RS 4n
544 The cumulative execution time for the process.
545 .RE

547 .sp
548 .ne 2
549 .na
550 \fBCPU\fR
551 .ad
552 .sp .6
553 .RS 4n
554 The percentage of recent \fBCPU\fR time used by the process. If executing in a
555 non-global \fBzone\fR and the pools facility is active, the percentage will be
556 that of the processors in the processor set in use by the pool to which the
557 \fBzone\fR is bound.
558 .RE

560 .sp
561 .ne 2
562 .na
563 \fBPROCESS\fR
564 .ad
565 .sp .6
566 .RS 4n
567 The name of the process (name of executed file).
568 .RE

570 .sp
571 .ne 2
572 .na
573 \fBLWPID\fR
574 .ad
575 .sp .6
576 .RS 4n
577 The \fBlwp\fR \fBID\fR of the \fBlwp\fR being reported.
578 .RE

580 .sp
581 .ne 2
582 .na
583 \fBNLWP\fR
584 .ad
585 .sp .6
586 .RS 4n
587 The number of \fBlwp\fRs in the process.
588 .RE
```



```
590 .sp
591 .LP
592 With the some options, in addition to a number of the column headings shown
593 above, there are:
594 .sp
595 .ne 2
596 .na
597 \fBNPROC\fR
598 .ad
599 .sp .6
600 .RS 4n
601 Number of processes in a specified collection.
602 .RE

604 .sp
605 .ne 2
606 .na
607 \fBMEMORY\fR
608 .ad
609 .sp .6
610 .RS 4n
611 Percentage of memory used by a specified collection of processes.
612 .RE

614 .sp
615 .LP
616 The following columns are displayed when the \fB-v\fR or \fB-m\fR option is
617 specified
618 .sp
619 .ne 2
620 .na
621 \fBUSR\fR
622 .ad
623 .sp .6
624 .RS 4n
625 The percentage of time the process has spent in user mode.
626 .RE

628 .sp
629 .ne 2
630 .na
631 \fBSYS\fR
632 .ad
633 .sp .6
634 .RS 4n
635 The percentage of time the process has spent in system mode.
636 .RE

638 .sp
639 .ne 2
640 .na
641 \fBTRP\fR
642 .ad
643 .sp .6
644 .RS 4n
645 The percentage of time the process has spent in processing system traps.
646 .RE

648 .sp
649 .ne 2
650 .na
651 \fBTFL\fR
652 .ad
653 .sp .6
654 .RS 4n
```

```
655 The percentage of time the process has spent processing text page faults.
656 .RE

658 .sp
659 .ne 2
660 .na
661 \fBDFL\fR
662 .ad
663 .sp .6
664 .RS 4n
665 The percentage of time the process has spent processing data page faults.
666 .RE

668 .sp
669 .ne 2
670 .na
671 \fBLCK\fR
672 .ad
673 .sp .6
674 .RS 4n
675 The percentage of time the process has spent waiting for user locks.
676 .RE

678 .sp
679 .ne 2
680 .na
681 \fBSLP\fR
682 .ad
683 .sp .6
684 .RS 4n
685 The percentage of time the process has spent sleeping.
686 .RE

688 .sp
689 .ne 2
690 .na
691 \fBLAT\fR
692 .ad
693 .sp .6
694 .RS 4n
695 The percentage of time the process has spent waiting for CPU.
696 .RE

698 .sp
699 .ne 2
700 .na
701 \fBVCX\fR
702 .ad
703 .sp .6
704 .RS 4n
705 The number of voluntary context switches.
706 .RE

708 .sp
709 .ne 2
710 .na
711 \fBICX\fR
712 .ad
713 .sp .6
714 .RS 4n
715 The number of involuntary context switches.
716 .RE

718 .sp
719 .ne 2
720 .na
```



```
721 \fBSCL\fR
722 .ad
723 .sp .6
724 .RS 4n
725 The number of system calls.
726 .RE

728 .sp
729 .ne 2
730 .na
731 \fBSIG\fR
732 .ad
733 .sp .6
734 .RS 4n
735 The number of signals received.
736 .RE

738 .sp
739 .LP
740 Under the \fB-L\fR option, one line is printed for each \fBlwp\fR in the
741 process and some reporting fields show the values for the \fBlwp\fR, not the
742 process.
743 .sp
744 .LP
745 The following column is displayed when the \fB-H\fR option is specified:
746 .sp
747 .ne 2
748 .na
749 \fBLGRP\fR
750 .ad
751 .sp .6
752 .RS 4n
753 The home \fIlgroup\fR of the process or lwp.
754 .RE

756 .SH OPERANDS
757 .sp
758 .LP
759 The following operands are supported:
760 .sp
761 .ne 2
762 .na
763 \fB\fIcount\fR\fR
764 .ad
765 .sp .6
766 .RS 4n
767 Specifies the number of times that the statistics are repeated. By default,
768 \fBprstat\fR reports statistics until a termination signal is received.
769 .RE

771 .sp
772 .ne 2
773 .na
774 \fB\fIinterval\fR\fR
775 .ad
776 .sp .6
777 .RS 4n
778 Specifies the sampling interval in seconds; the default interval is \fB5\fR
779 seconds.
780 .RE

782 .SH EXAMPLES
783 .LP
784 \fBExample 1 \fRReporting the Five Most Active Super-User Processes
785 .sp
786 .LP
```

```
787 The following command reports the five most active super-user processes running
788 on \fBCPU1\fR and \fBCPU2\fR:

790 .sp
791 .in +2
792 .nf
793 example% prstat -u root -n 5 -P 1,2 1 1

795 PID   USERNAME  SWAP   RSS STATE  PRI  NICE      TIME  CPU PROCESS/LWP
796  306   root     3024K 1448K sleep   58    0   0:00.00 0.3% sendmail/1
797  102   root     1600K  592K sleep   59    0   0:00.00 0.1% in.rdisc/1
798  250   root     1000K  552K sleep   58    0   0:00.00 0.0% utmpd/1
799  288   root     1720K 1032K sleep   58    0   0:00.00 0.0% sac/1
800    1   root      744K  168K sleep   58    0   0:00.00 0.0% init/1
801 TOTAL:       25, load averages:  0.05, 0.08, 0.12
802 .fi
803 .in -2
804 .sp

806 .LP
807 \fBExample 2 \fRDisplaying Verbose Process Usage Information
808 .sp
809 .LP
810 The following command displays verbose process usage information about
811 processes with lowest resident set sizes owned by users \fBroot\fR and
812 \fBjohn\fR.

814 .sp
815 .in +2
816 .nf
817 example% prstat -S rss -n 5 -vc -u root,john

819  PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWP
820    1 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 init/1
821  102 root     0.0 0.0  -   -   -   -  100  -    0   0   3   0 in.rdisc/1
822  250 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 utmpd/1
823 1185 john    0.0 0.0  -   -   -   -  100  -    0   0   0   0 csh/1
824  240 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 powerd/4
825  TOTAL:       71, load averages:  0.02, 0.04, 0.08

827 .fi
828 .in -2
829 .sp

831 .SH EXIT STATUS
832 .sp
833 .LP
834 The following exit values are returned:
835 .sp
836 .ne 2
837 .na
838 \fB\fB0\fR\fR
839 .ad
840 .sp .6
841 .RS 4n
842 Successful completion.
843 .RE

845 .sp
846 .ne 2
847 .na
848 \fB\fB1\fR\fR
849 .ad
850 .sp .6
851 .RS 4n
852 An error occurred.
```



```
853 .RE

855 .SH SEE ALSO
856 .sp
857 .LP
858 \fBdate\fR(1), \fBlgrpinfo\fR(1), \fBplgrp\fR(1), \fBproc\fR(1), \fBps\fR(1),
859 \fBtime\fR(2), \fBpsrinfo\fR(1M), \fBpsrset\fR(1M), \fBsar\fR(1M),
860 \fBpset_getloadavg\fR(3C), \fBproc\fR(4), \fBproject\fR(4),
861 \fBattributes\fR(5), \fBresource_controls\fR(5), \fBzones\fR(5)
862 .SH NOTES
863 .sp
864 .LP
865 The snapshot of system usage displayed by \fBprstat\fR is true only for a
866 split-second, and it may not be accurate by the time it is displayed. When the
867 \fB-m\fR option is specified, \fBprstat\fR tries to turn on microstate
868 accounting for each process; the original state is restored when \fBprstat\fR
869 exits. See \fBproc\fR(4) for additional information about the microstate
870 accounting facility.
871 .sp
872 .LP
873 The total memory size reported in the SWAP and RSS columns for groups of
874 processes can sometimes overestimate the actual amount of memory used by
875 processes with shared memory segments.
```

```
**********************************************************
    9883 Mon Mar 25 12:53:27 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
```