```
**********************************************************
    3176 Sat Sep 19 01:15:20 2015
new/usr/src/cmd/oamuser/user/Makefile
293 useradd/del/mod should be ZFS-aware
**********************************************************
#
# 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) 1990, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2013 RackTop Systems.
# Copyright (c) 2013 Gary Mills
#
# cmd/oamuser/user/Makefile
#

DEFAULTFILES= useradd.dfl

include ../../Makefile.cmd

GREP=           grep

USERADD=        useradd
USERDEL=        userdel
USERMOD=        usermod
ROLEADD=        roleadd
ROLEDEL=        roledel
ROLEMOD=        rolemod

SBINPROG=       $(USERADD)   $(USERDEL)   $(USERMOD)
#
#       Removing sysadm: deleted $(SYSADMPROG) from this target.
#
PROG=           $(SBINPROG)
PRODUCT=        $(PROG)

ADD_OBJ=        useradd.o     homedir.o       groups.o     call_pass.o \
                userdefs.o    messages.o      val_lgrp.o   funcs.o \
                val_lprj.o    proj.o

DEL_OBJ=        userdel.o     call_pass.o     homedir.o    isbusy.o \
DEL_OBJ=        userdel.o     call_pass.o     rmfiles.o    isbusy.o \
                groups.o      messages.o      funcs.o      proj.o

MOD_OBJ=        usermod.o     movedir.o       groups.o     homedir.o \
                call_pass.o   isbusy.o        userdefs.o \
MOD_OBJ=        usermod.o     movedir.o       groups.o     rmfiles.o \
                call_pass.o   isbusy.o        homedir.o    userdefs.o \
                messages.o    val_lgrp.o      funcs.o      val_lprj.o \
```



```
                proj.o

OBJECTS=        $(ADD_OBJ)     $(DEL_OBJ)    $(MOD_OBJ)

SRCS=           $(OBJECTS:.o=.c)

LIBDIR=         ../lib
LIBUSRGRP=      $(LIBDIR)/lib.a
LIBADM=         -ladm
LOCAL=          ../inc
HERE=           .
LINTFLAGS=      -u

ROOTSKEL=       $(ROOTETC)/skel
INSSBINPROG=    $(SBINPROG:%=$(ROOTUSRSBIN)/%)
INSSKELFILE=    $(SKELFILE:%=$(ROOTSKEL)/%)

CPPFLAGS=       -I$(HERE) -I$(LOCAL) $(CPPFLAGS.master)
CERRWARN +=     -_gcc=-Wno-implicit-function-declaration

$(INSSBINPROG)  := FILEMODE = 0555
$(INSSYSADMPROG):= FILEMODE = 0500
$(INSSKELFILE)  := FILEMODE = 0644

$(USERADD) :=   OBJS = $(ADD_OBJ)
$(USERADD) :=   LIBS = $(LIBUSRGRP)
$(USERADD) :=   LDLIBS += -lcmdutils

$(USERDEL) :=   OBJS = $(DEL_OBJ)
$(USERDEL) :=   LIBS = $(LIBUSRGRP)

$(USERMOD) :=   OBJS = $(MOD_OBJ)
$(USERMOD) :=   LIBS = $(LIBUSRGRP)

LDLIBS +=       -lbsm -lnsl -lsecdb -lproject -lzfs -ltsol
LDLIBS +=       -lbsm -lnsl -lsecdb -lproject -ltsol

.PARALLEL: $(OBJECTS)

all:            $(PRODUCT)

$(PROG):        $$(OBJS) $$(LIBS)
        $(LINK.c) $(OBJS) -o $@ $(LIBS) $(LDLIBS)
        $(POST_PROCESS)

$(USERADD):     $(ADD_OBJ)
$(USERMOD):     $(MOD_OBJ)
$(USERDEL):     $(DEL_OBJ)

install:        all $(ROOTETCDEFAULTFILES) .WAIT \
                        $(ROOTSKEL) $(INSSBINPROG) $(INSSKELFILE)
                $(RM) $(ROOTUSRSBIN)/$(ROLEADD)
                $(LN) $(ROOTUSRSBIN)/$(USERADD) $(ROOTUSRSBIN)/$(ROLEADD)
                $(RM) $(ROOTUSRSBIN)/$(ROLEDEL)
                $(LN) $(ROOTUSRSBIN)/$(USERDEL) $(ROOTUSRSBIN)/$(ROLEDEL)
                $(RM) $(ROOTUSRSBIN)/$(ROLEMOD)
                $(LN) $(ROOTUSRSBIN)/$(USERMOD) $(ROOTUSRSBIN)/$(ROLEMOD)

clean:
        $(RM) $(OBJECTS)

lint:   lint_SRCS

include ../../Makefile.targ
```

```
**********************************************************
    1517 Sat Sep 19 01:15:21 2015
new/usr/src/cmd/oamuser/user/funcs.h
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
/*
 * 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 1999,2002-2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

#ifndef _FUNCS_H
#define _FUNCS_H

#ifdef  __cplusplus
extern "C" {
#endif

#define CMD_PREFIX_USER "user"

#define AUTH_SEP        ","
#define PROF_SEP        ","
#define ROLE_SEP        ","

#define MAX_TYPE_LENGTH 64

char *getusertype(char *cmdname);

int is_role(char *usertype);

void change_key(const char *, char *);
void addkey_args(char **, int *);
char *getsetdefval(const char *, char *);

extern int nkeys;

/* create_home() or rm_files() flags */
#define CHANGE_ZFS_FS_OPT       "CHANGE_ZFS_FS="
#define CHANGE_ZFS_FS   1
#endif /* ! codereview */

#ifdef  __cplusplus
}
#endif
```



```
#endif  /* _FUNCS_H */
```

```
**********************************************************
    6696 Sat Sep 19 01:15:21 2015
new/usr/src/cmd/oamuser/user/homedir.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
/*
 * 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 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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


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

#include <sys/types.h>
#include <sys/stat.h>
#endif /* ! codereview */
#include <stdio.h>
#include <userdefs.h>
#include <errno.h>
#include <strings.h>
#include <sys/mntent.h>
#include <sys/mnttab.h>
#include <libzfs.h>
#include <libgen.h>
#include <limits.h>
#include <deflt.h>

#include "funcs.h"
#endif /* ! codereview */
#include "messages.h"

#define SBUFSZ  256

#define DEFAULT_USERADD "/etc/default/useradd"

static int rm_homedir();
static char *get_mnt_special();
extern int rm_homedir();

static char cmdbuf[ SBUFSZ ];   /* buffer for system call */
static char dhome[ PATH_MAX + 1 ]; /* buffer for dirname */
```



```
static char bhome[ PATH_MAX + 1 ]; /* buffer for basename */
static char pdir[ PATH_MAX + 1 ]; /* parent directory */
static libzfs_handle_t *g_zfs = NULL;
#endif /* ! codereview */

/*
 * Create a home directory and populate with files from skeleton
 * directory.
 */
        Create a home directory and populate with files from skeleton
        directory.
*/
int
create_home(char *homedir, char *skeldir, uid_t uid, gid_t gid, int flags)
create_home(char *homedir, char *skeldir, uid_t uid, gid_t gid)
                /* home directory to create */
                /* skel directory to copy if indicated */
                /* uid of new user */
                /* group id of new user */
                /* miscellaneous flags */
#endif /* ! codereview */
{
        struct stat stbuf;
        char *dataset;
        char *dname, *bname;

        if (g_zfs == NULL)
                g_zfs = libzfs_init();

        (void) strcpy(dhome, homedir);
        (void) strcpy(bhome, homedir);
        dname = dirname(dhome);
        bname = basename(bhome);
        (void) strcpy(pdir, dname);

        if ((stat(pdir, &stbuf) != 0) || !S_ISDIR(stbuf.st_mode)) {
                errmsg(M_OOPS, "access the parent directory", strerror(errno));
                if (g_zfs) {
                        libzfs_fini(g_zfs);
                        g_zfs = NULL;
                }
                return (EX_HOMEDIR);
        }

        if ((strcmp(stbuf.st_fstype, MNTTYPE_ZFS) == 0) &&
            (g_zfs != NULL) && (flags & CHANGE_ZFS_FS) &&
            ((dataset = get_mnt_special(pdir, stbuf.st_fstype)) != NULL)) {
                char nm[ZFS_MAXNAMELEN];
                zfs_handle_t *zhp;

                (void) snprintf(nm, ZFS_MAXNAMELEN, "%s/%s", dataset, bname);

                if ((zfs_create(g_zfs, nm, ZFS_TYPE_FILESYSTEM, NULL) != 0) ||
                    ((zhp = zfs_open(g_zfs, nm, ZFS_TYPE_FILESYSTEM)) ==
                    NULL)) {
                        errmsg(M_OOPS, "create the home directory",
                            libzfs_error_description(g_zfs));
                        libzfs_fini(g_zfs);
                        g_zfs = NULL;
                        return (EX_HOMEDIR);
                }

                if (zfs_mount(zhp, NULL, 0) != 0) {
                        errmsg(M_OOPS, "mount the home directory",
                            libzfs_error_description(g_zfs));
                        (void) zfs_destroy(zhp, B_FALSE);
```

```
121                         zfs_close(zhp);
122                         libzfs_fini(g_zfs);
123                         g_zfs = NULL;
124                         return (EX_HOMEDIR);
125                 }

127                 zfs_close(zhp);

129                 if (chmod(homedir, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH)
130                         != 0) {
131                         errmsg(M_OOPS, "change permissions of home directory",
132                             strerror(errno));
133                         libzfs_fini(g_zfs);
134                         g_zfs = NULL;
135                         return (EX_HOMEDIR);
136                 }
137         } else {
138                 if (mkdir(homedir, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH)
139                         != 0) {
140                         errmsg(M_OOPS, "create the home directory",
141                             strerror(errno));
142                         if (g_zfs) {
143                                 libzfs_fini(g_zfs);
144                                 g_zfs = NULL;
145                         }
146                         return (EX_HOMEDIR);
147                 }
 47         if( mkdir(homedir, 0775) != 0 ) {
 48                 errmsg(M_OOPS, "create the home directory", strerror(errno));
 49                 return( EX_HOMEDIR );
148         }

150         if (chown(homedir, uid, gid) != 0) {
 52         if( chown(homedir, uid, gid) != 0 ) {
151                 errmsg(M_OOPS, "change ownership of home directory",
152                     strerror(errno));
153                 if (g_zfs) {
154                         libzfs_fini(g_zfs);
155                         g_zfs = NULL;
156                 }
157                 return (EX_HOMEDIR);
 55                 return( EX_HOMEDIR );
158         }

160         if (skeldir) {
 58         if(skeldir) {
161                 /* copy the skel_dir into the home directory */
162                 (void) sprintf(cmdbuf, "cd %s && find . -print | cpio -pd %s",
 60                 (void) sprintf( cmdbuf, "cd %s && find . -print | cpio -pd %s",
163                         skeldir, homedir);

165                 if (system(cmdbuf) != 0) {
 63                 if( system( cmdbuf ) != 0 ) {
166                         errmsg(M_OOPS, "copy skeleton directory into home "
167                             "directory", strerror(errno));
168                         (void) rm_homedir(homedir, flags);
169                         if (g_zfs) {
170                                 libzfs_fini(g_zfs);
171                                 g_zfs = NULL;
172                         }
173                         return (EX_HOMEDIR);
 66                         (void) rm_homedir( homedir );
 67                         return( EX_HOMEDIR );
174                 }

176                 /* make sure contents in the home dirctory have correct owner */
```



```
177                 (void) sprintf(cmdbuf,
178                     "cd %s && find . -exec chown %ld:%ld {} \\;",
179                     homedir, uid, gid);
180                 if (system(cmdbuf) != 0) {
181                         errmsg(M_OOPS,
182                             "change owner and group of files home directory",
 71                 (void) sprintf( cmdbuf,"cd %s && find . -exec chown %ld {} \\;",
 72                         homedir, uid );
 73                 if( system( cmdbuf ) != 0) {
 74                         errmsg(M_OOPS, "change owner of files home directory",
183                             strerror(errno));
184                         (void) rm_homedir(homedir, flags);
185                         if (g_zfs) {
186                                 libzfs_fini(g_zfs);
187                                 g_zfs = NULL;
188                         }
189                         return (EX_HOMEDIR);
190                 }
191 #endif /* ! codereview */

 76                         (void) rm_homedir( homedir );
 77                         return( EX_HOMEDIR );
193         }
194         if (g_zfs) {
195                 libzfs_fini(g_zfs);
196                 g_zfs = NULL;
197         }
198         return (EX_SUCCESS);
199 }

201 /* Remove a home directory structure */
202 int
203 rm_homedir(char *dir, int flags)
204 {
205         struct stat stbuf;
206         char *nm;

208         if ((stat(dir, &stbuf) != 0) || !S_ISDIR(stbuf.st_mode))
209                 return (0);
210 #endif /* ! codereview */

212         if (g_zfs == NULL)
213                 g_zfs = libzfs_init();

215         if ((strcmp(stbuf.st_fstype, MNTTYPE_ZFS) == 0) &&
216             (flags & CHANGE_ZFS_FS) &&
217             (g_zfs != NULL) &&
218             ((nm = get_mnt_special(dir, stbuf.st_fstype)) != NULL)) {
219                 zfs_handle_t *zhp;

221                 if ((zhp = zfs_open(g_zfs, nm, ZFS_TYPE_FILESYSTEM)) != NULL) {
222                         if ((zfs_unmount(zhp, NULL, 0) == 0) &&
223                             (zfs_destroy(zhp, B_FALSE) == 0)) {
224                                 zfs_close(zhp);
225                                 return (0);
226                         }

228                         (void) zfs_mount(zhp, NULL, 0);
229                         zfs_close(zhp);
230                 }
 79                 /* and group..... */
 80                 (void) sprintf( cmdbuf, "cd %s && find . -exec chgrp %ld {} \\;"
 81                         homedir, gid );
 82                 if( system( cmdbuf ) != 0) {
 83                         errmsg(M_OOPS, "change group of files home directory",
 84                             strerror(errno));
```

```
 85                         (void) rm_homedir( homedir );
 86                         return( EX_HOMEDIR );
231         }

233         (void) sprintf(cmdbuf, "rm -rf %s", dir);

235         return (system(cmdbuf));
236 }

238 int
239 rm_files(char *homedir, char *user, int flags)
240 {
241         if (rm_homedir(homedir, flags) != 0) {
242                 errmsg(M_RMFILES);
243                 return (EX_HOMEDIR);
244 #endif /* ! codereview */
245         }

247         return (EX_SUCCESS);
 88         return( EX_SUCCESS );
248 }

 91 /* Remove a home directory structure */
250 int
251 get_default_zfs_flags()
252 {
253         int flags = 0;

255         if (defopen(DEFAULT_USERADD) == 0) {
256                 char *defptr;

258                 if ((defptr = defread(CHANGE_ZFS_FS_OPT)) != NULL) {
259                         char let = tolower(*defptr);

261                         switch (let) {
262                                 case 'y':       /* yes */
263                                         flags |= CHANGE_ZFS_FS;
264                                 case 'n':       /* no */
265                                         break;
266                         }
267                 }
268                 (void) defopen((char *)NULL);
269         }
270         return (flags);
271 }

273 /* Get the name of a mounted filesytem */
274 char *
275 get_mnt_special(char *mountp, char *fstype)
 93 rm_homedir(char *dir)
276 {
277         struct mnttab entry, search;
278         char *special = NULL;
279         FILE *fp;

281         search.mnt_special = search.mnt_mntopts = search.mnt_time = NULL;
282         search.mnt_mountp = mountp;
283         search.mnt_fstype = fstype;

285         if ((fp = fopen(MNTTAB, "r")) != NULL) {
286                 if (getmntany(fp, &entry, &search) == 0)
287                         special = entry.mnt_special;

289                 (void) fclose(fp);
290         }
 95         (void) sprintf( cmdbuf, "rm -rf %s", dir );
```



```
292         return (special);
 97         return( system( cmdbuf ) );
293 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    4824 Sat Sep 19 01:15:21 2015
new/usr/src/cmd/oamuser/user/messages.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
/*
 * 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.
 */

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 [-z|Z] [-k skel_dir] |"
            "-f inactive |\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 [-z|Z]] login\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 [-z|Z]] | -s shell | -c comment |\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_
```

```
**********************************************************
    2487 Sat Sep 19 01:15:21 2015
new/usr/src/cmd/oamuser/user/movedir.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
/*
 * 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 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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


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

#include <sys/types.h>
#include <stdio.h>
#include <userdefs.h>
#include "messages.h"
#include <unistd.h>
#include <sys/stat.h>
#include <utime.h>

#define SBUFSZ  256

extern int access(), rm_files();

static char cmdbuf[SBUFSZ];     /* buffer for system call */

/*
 *      Move directory contents from one place to another
 */
int
move_dir(char *from, char *to, char *login, int flags)
move_dir(char *from, char *to, char *login)
                /* directory to move files from */
                /* dirctory to move files to */
                /* login id of owner */
                /* miscellaneous flags */
#endif /* ! codereview */
{
        size_t len = 0;
        int rc = EX_SUCCESS;
```



```
        struct stat statbuf;
        struct utimbuf times;
        /*
         * ***** THIS IS WHERE SUFFICIENT SPACE CHECK GOES
         */

        if (access(from, F_OK) == 0) {  /* home dir exists */
                /* move all files */
                (void) sprintf(cmdbuf,
                        "cd %s && find . -print | cpio -m -pd %s",
                        from, to);

                if (system(cmdbuf) != 0) {
                        errmsg(M_NOSPACE, from, to);
                        return (EX_NOSPACE);
                }

                /*
                 * Check that to dir is not a subdirectory of from
                 */
                len = strlen(from);
                if (strncmp(from, to, len) == 0 &&
                    strncmp(to+len, "/", 1) == 0) {
                        errmsg(M_RMFILES);
                        return (EX_HOMEDIR);
                }
                /* Retain the original permission and modification time */
                if (stat(from, &statbuf) == 0) {
                        chmod(to, statbuf.st_mode);
                        times.actime = statbuf.st_atime;
                        times.modtime = statbuf.st_mtime;
                        (void) utime(to, &times);

                }

                /* Remove the files in the old place */
                rc = rm_files(from, login, flags);
                rc = rm_files(from, login);

        }

        return (rc);
}
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   18152 Sat Sep 19 01:15:21 2015
new/usr/src/cmd/oamuser/user/useradd.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
/*
 * 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   */

/*
 * Copyright (c) 2013 RackTop Systems.
 */

#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        <libcmdutils.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 [-z|Z]]
 *  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 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 *);
extern int get_default_zfs_flags();
#endif /* ! codereview */

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 */
```

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

129 typedef enum {
130         BASEDIR = 0,
131         SKELDIR,
132         SHELL
133 } path_opt_t;


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

138 int
139 main(argc, argv)
140 int argc;
141 char *argv[];
142 {
143         int ch, ret, mflag = 0, oflag = 0, Dflag = 0;
144         int zflag = 0, Zflag = 0, **gidlist = NULL;
 99         int ch, ret, mflag = 0, oflag = 0, Dflag = 0, **gidlist = NULL;
145         projid_t **projlist = NULL;
146         char *ptr;                      /* loc in a str, may be set by strtol */
147         struct group *g_ptr;
148         struct project p_ptr;
149         char mybuf[PROJECT_BUFSZ];
150         struct stat statbuf;            /* status buffer for stat */
151         int warning;
152         int busy = 0;
153         char **nargv;                   /* arguments for execvp of passmgmt */
154         int argindex;                   /* argument index into nargv */
155         int zfs_flags = 0;                      /* create_home flags */
156 #endif /* ! codereview */

158         cmdname = argv[0];

160         if (geteuid() != 0) {
161                 errmsg(M_PERM_DENIED);
162                 exit(EX_NO_PERM);
163         }

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

168         change_key(USERATTR_TYPE_KW, usertype);

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

177                 case 'c':
178                         comment = optarg;
179                         break;

181                 case 'D':
182                         Dflag++;
183                         break;

185                 case 'd':
186                         dir = optarg;
187                         break;

189                 case 'e':
190                         expirestr = optarg;
```



```
191                         break;

193                 case 'f':
194                         inactstr = optarg;
195                         break;

197                 case 'G':
198                         grps = optarg;
199                         break;

201                 case 'g':
202                         group = optarg;
203                         break;

205                 case 'k':
206                         skel_dir = optarg;
207                         break;

209                 case 'm':
210                         mflag++;
211                         break;

213                 case 'o':
214                         oflag++;
215                         break;

217                 case 'p':
218                         projects = optarg;
219                         break;

221                 case 's':
222                         shell = optarg;
223                         break;

225                 case 'u':
226                         uidstr = optarg;
227                         break;

229                 case 'Z':
230                         Zflag++;
231                         break;

233                 case 'z':
234                         zflag++;
235                         break;

237 #endif /* ! codereview */
238                 case 'A':
239                         change_key(USERATTR_AUTHS_KW, optarg);
240                         break;

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

246                 case 'R':
247                         if (is_role(usertype)) {
248                                 errmsg(M_ARUSAGE);
249                                 exit(EX_SYNTAX);
250                         }
251                         change_key(USERATTR_ROLES_KW, optarg);
252                         break;

254                 case 'K':
255                         change_key(NULL, optarg);
256                         break;
```

```
258                 default:
259                 case '?':
260                         if (is_role(usertype))
261                                 errmsg(M_ARUSAGE);
262                         else
263                                 errmsg(M_AUSAGE);
264                         exit(EX_SYNTAX);
265                 }

267         if (((!mflag) && (zflag || Zflag)) || (zflag && Zflag) ||
268             (mflag > 1 && (zflag || Zflag))) {
269                 if (is_role(usertype))
270                         errmsg(M_ARUSAGE);
271                 else
272                         errmsg(M_AUSAGE);
273                 exit(EX_SYNTAX);
274         }


277 #endif /* ! codereview */
278         /* get defaults for adding new users */
279         usrdefs = getusrdef(usertype);

281         if (Dflag) {
282                 /* DISPLAY mode */

284                 /* check syntax */
285                 if (optind != argc) {
286                         if (is_role(usertype))
287                                 errmsg(M_ARUSAGE);
288                         else
289                                 errmsg(M_AUSAGE);
290                         exit(EX_SYNTAX);
291                 }

293                 if (uidstr != NULL || oflag || grps != NULL ||
294                     dir != NULL || mflag || comment != NULL) {
295                         if (is_role(usertype))
296                                 errmsg(M_ARUSAGE);
297                         else
298                                 errmsg(M_AUSAGE);
299                         exit(EX_SYNTAX);
300                 }

302                 /* Group must be an existing group */
303                 if (group != NULL) {
304                         switch (valid_group(group, &g_ptr, &warning)) {
305                         case INVALID:
306                                 errmsg(M_INVALID, group, "group id");
307                                 exit(EX_BADARG);
308                                 /*NOTREACHED*/
309                         case TOOBIG:
310                                 errmsg(M_TOOBIG, "gid", group);
311                                 exit(EX_BADARG);
312                                 /*NOTREACHED*/
313                         case RESERVED:
314                         case UNIQUE:
315                                 errmsg(M_GRP_NOTUSED, group);
316                                 exit(EX_NAME_NOT_EXIST);
317                         }
318                         if (warning)
319                                 warningmsg(warning, group);

321                         usrdefs->defgroup = g_ptr->gr_gid;
322                         usrdefs->defgname = g_ptr->gr_name;
```



```
324                 }

326                 /* project must be an existing project */
327                 if (projects != NULL) {
328                         switch (valid_project(projects, &p_ptr, mybuf,
329                             sizeof (mybuf), &warning)) {
330                         case INVALID:
331                                 errmsg(M_INVALID, projects, "project id");
332                                 exit(EX_BADARG);
333                                 /*NOTREACHED*/
334                         case TOOBIG:
335                                 errmsg(M_TOOBIG, "projid", projects);
336                                 exit(EX_BADARG);
337                                 /*NOTREACHED*/
338                         case UNIQUE:
339                                 errmsg(M_PROJ_NOTUSED, projects);
340                                 exit(EX_NAME_NOT_EXIST);
341                         }
342                         if (warning)
343                                 warningmsg(warning, projects);

345                         usrdefs->defproj = p_ptr.pj_projid;
346                         usrdefs->defprojname = p_ptr.pj_name;
347                 }

349                 /* base_dir must be an existing directory */
350                 if (base_dir != NULL) {
351                         valid_input(BASEDIR, base_dir);
352                         usrdefs->defparent = base_dir;
353                 }

355                 /* inactivity period is an integer */
356                 if (inactstr != NULL) {
357                         /* convert inactstr to integer */
358                         inact = strtol(inactstr, &ptr, 10);
359                         if (*ptr || inact < 0) {
360                                 errmsg(M_INVALID, inactstr,
361                                     "inactivity period");
362                                 exit(EX_BADARG);
363                         }

365                         usrdefs->definact = inact;
366                 }

368                 /* expiration string is a date, newer than today */
369                 if (expirestr != NULL) {
370                         if (*expirestr) {
371                                 if (valid_expire(expirestr, (time_t *)0)
372                                     == INVALID) {
373                                         errmsg(M_INVALID, expirestr,
374                                             "expiration date");
375                                         exit(EX_BADARG);
376                                 }
377                                 usrdefs->defexpire = expirestr;
378                         } else
379                                 /* Unset the expiration date */
380                                 usrdefs->defexpire = "";
381                 }

383                 if (shell != NULL) {
384                         valid_input(SHELL, shell);
385                         usrdefs->defshell = shell;
386                 }
387                 if (skel_dir != NULL) {
388                         valid_input(SKELDIR, skel_dir);
```

```
389                         usrdefs->defskel = skel_dir;
390                 }
391                 update_def(usrdefs);

393                 /* change defaults for useradd */
394                 if (putusrdef(usrdefs, usertype) < 0) {
395                         errmsg(M_UPDATE, "created");
396                         exit(EX_UPDATE);
397                 }

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

403         }

405         /* ADD mode */

407         /* check syntax */
408         if (optind != argc - 1 || (skel_dir != NULL && !mflag)) {
409                 if (is_role(usertype))
410                         errmsg(M_ARUSAGE);
411                 else
412                         errmsg(M_AUSAGE);
413                 exit(EX_SYNTAX);
414         }

416         logname = argv[optind];
417         switch (valid_login(logname, (struct passwd **)NULL, &warning)) {
418         case INVALID:
419                 errmsg(M_INVALID, logname, "login name");
420                 exit(EX_BADARG);
421                 /*NOTREACHED*/

423         case NOTUNIQUE:
424                 errmsg(M_USED, logname);
425                 exit(EX_NAME_EXISTS);
426                 /*NOTREACHED*/

428         case LONGNAME:
429                 errmsg(M_TOO_LONG, logname);
430                 exit(EX_BADARG);
431                 /*NOTREACHED*/
432         }

434         if (warning)
435                 warningmsg(warning, logname);
436         if (uidstr != NULL) {
437                 /* convert uidstr to integer */
438                 errno = 0;
439                 uid = (uid_t)strtol(uidstr, &ptr, (int)10);
440                 if (*ptr || errno == ERANGE) {
441                         errmsg(M_INVALID, uidstr, "user id");
442                         exit(EX_BADARG);
443                 }

445                 switch (valid_uid(uid, NULL)) {
446                 case NOTUNIQUE:
447                         if (!oflag) {
448                                 /* override not specified */
449                                 errmsg(M_UID_USED, uid);
450                                 exit(EX_ID_EXISTS);
451                         }
452                         break;
453                 case RESERVED:
454                         errmsg(M_RESERVED, uid);
```



```
455                         break;
456                 case TOOBIG:
457                         errmsg(M_TOOBIG, "uid", uid);
458                         exit(EX_BADARG);
459                         break;
460                 }

462         } else {

464                 if (findnextuid(DEFRID+1, MAXUID, &uid) != 0) {
465                         errmsg(M_INVALID, "default id", "user id");
466                         exit(EX_ID_EXISTS);
467                 }
468         }

470         if (group != NULL) {
471                 switch (valid_group(group, &g_ptr, &warning)) {
472                 case INVALID:
473                         errmsg(M_INVALID, group, "group id");
474                         exit(EX_BADARG);
475                         /*NOTREACHED*/
476                 case TOOBIG:
477                         errmsg(M_TOOBIG, "gid", group);
478                         exit(EX_BADARG);
479                         /*NOTREACHED*/
480                 case RESERVED:
481                 case UNIQUE:
482                         errmsg(M_GRP_NOTUSED, group);
483                         exit(EX_NAME_NOT_EXIST);
484                         /*NOTREACHED*/
485                 }

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

491         } else gid = usrdefs->defgroup;

493         if (grps != NULL) {
494                 if (!*grps)
495                         /* ignore -G "" */
496                         grps = (char *)0;
497                 else if (!(gidlist = valid_lgroup(grps, gid)))
498                         exit(EX_BADARG);
499         }

501         if (projects != NULL) {
502                 if (! *projects)
503                         projects = (char *)0;
504                 else if (! (projlist = valid_lproject(projects)))
505                         exit(EX_BADARG);
506         }

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

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

518         } else if (REL_PATH(dir)) {
519                 errmsg(M_RELPATH, dir);
520                 exit(EX_BADARG);
```

```
522         } else
523                 (void) strcpy(homedir, dir);
525         if (mflag) {
526                 /* Does home dir. already exist? */
527                 if (stat(homedir, &statbuf) == 0) {
528                         /* directory exists - don't try to create */
529                         mflag = 0;
531                         if (check_perm(statbuf, uid, gid, S_IXOTH) != 0)
532                                 errmsg(M_NO_PERM, logname, homedir);
533                 }
534         }
535         /*
536          * if shell, skel_dir are provided, check their validity.
537          * Otherwise default.
538          */
539         if (shell != NULL)
540                 valid_input(SHELL, shell);
541         else
542                 shell = usrdefs->defshell;
544         if (skel_dir != NULL)
545                 valid_input(SKELDIR, skel_dir);
546         else
547                 skel_dir = usrdefs->defskel;
549         if (inactstr != NULL) {
550                 /* convert inactstr to integer */
551                 inact = strtol(inactstr, &ptr, 10);
552                 if (*ptr || inact < 0) {
553                         errmsg(M_INVALID, inactstr, "inactivity period");
554                         exit(EX_BADARG);
555                 }
556         } else inact = usrdefs->definact;
558         /* expiration string is a date, newer than today */
559         if (expirestr != NULL) {
560                 if (*expirestr) {
561                         if (valid_expire(expirestr, (time_t *)0) == INVALID) {
562                                 errmsg(M_INVALID, expirestr, "expiration date");
563                                 exit(EX_BADARG);
564                         }
565                         usrdefs->defexpire = expirestr;
566                 } else
567                         /* Unset the expiration date */
568                         expirestr = (char *)0;
570         } else expirestr = usrdefs->defexpire;
572         import_def(usrdefs);
574         /* must now call passmgmt */
576         /* set up arguments to  passmgmt in nargv array */
577         nargv = malloc((30 + nkeys * 2) * sizeof (char *));
578         argindex = 0;
579         nargv[argindex++] = PASSMGMT;
580         nargv[argindex++] = "-a";       /* add */
582         if (comment != NULL) {
583                 /* comment */
584                 nargv[argindex++] = "-c";
585                 nargv[argindex++] = comment;
586         }
```



```
588         /* flags for home directory */
589         nargv[argindex++] = "-h";
590         nargv[argindex++] = homedir;
592         /* set gid flag */
593         nargv[argindex++] = "-g";
594         (void) sprintf(gidstring, "%u", gid);
595         nargv[argindex++] = gidstring;
597         /* shell */
598         nargv[argindex++] = "-s";
599         nargv[argindex++] = shell;
601         /* set inactive */
602         nargv[argindex++] = "-f";
603         (void) sprintf(inactstring, "%ld", inact);
604         nargv[argindex++] = inactstring;
606         /* set expiration date */
607         if (expirestr != NULL) {
608                 nargv[argindex++] = "-e";
609                 nargv[argindex++] = expirestr;
610         }
612         /* set uid flag */
613         nargv[argindex++] = "-u";
614         (void) sprintf(uidstring, "%u", uid);
615         nargv[argindex++] = uidstring;
617         if (oflag) nargv[argindex++] = "-o";
619         if (nkeys > 1)
620                 addkey_args(nargv, &argindex);
622         /* finally - login name */
623         nargv[argindex++] = logname;
625         /* set the last to null */
626         nargv[argindex++] = NULL;
628         /* now call passmgmt */
629         ret = PEX_FAILED;
630         /*
631          * If call_passmgmt fails for any reason other than PEX_BADUID, exit
632          * is invoked with an appropriate error message. If PEX_BADUID is
633          * returned, then if the user specified the ID, exit is invoked
634          * with an appropriate error message. Otherwise we try to pick a
635          * different ID and try again. If we run out of IDs, i.e. no more
636          * users can be created, then -1 is returned and we terminate via exit.
637          * If PEX_BUSY is returned we increment a count, since we will stop
638          * trying if PEX_BUSY reaches 3. For PEX_SUCCESS we immediately
639          * terminate the loop.
640          */
641         while (busy < 3 && ret != PEX_SUCCESS) {
642                 switch (ret = call_passmgmt(nargv)) {
643                 case PEX_SUCCESS:
644                         break;
645                 case PEX_BUSY:
646                         busy++;
647                         break;
648                 case PEX_HOSED_FILES:
649                         errmsg(M_HOSED_FILES);
650                         exit(EX_INCONSISTENT);
651                         break;
```

```
653                 case PEX_SYNTAX:
654                 case PEX_BADARG:
655                         /* should NEVER occur that passmgmt usage is wrong */
656                         if (is_role(usertype))
657                                 errmsg(M_ARUSAGE);
658                         else
659                                 errmsg(M_AUSAGE);
660                         exit(EX_SYNTAX);
661                         break;

663                 case PEX_BADUID:
664                         /*
665                          * The uid has been taken. If it was specified by a
666                          * user, then we must fail. Otherwise, keep trying
667                          * to get a good uid until we run out of IDs.
668                          */
669                         if (uidstr != NULL) {
670                                 errmsg(M_UID_USED, uid);
671                                 exit(EX_ID_EXISTS);
672                         } else {
673                                 if (findnextuid(DEFRID+1, MAXUID, &uid) != 0) {
674                                         errmsg(M_INVALID, "default id",
675                                             "user id");
676                                         exit(EX_ID_EXISTS);
677                                 }
678                                 (void) sprintf(uidstring, "%u", uid);
679                         }
680                         break;

682                 case PEX_BADNAME:
683                         /* invalid loname */
684                         errmsg(M_USED, logname);
685                         exit(EX_NAME_EXISTS);
686                         break;

688                 default:
689                         errmsg(M_UPDATE, "created");
690                         exit(ret);
691                         break;
692                 }
693         }
694         if (busy == 3) {
695                 errmsg(M_UPDATE, "created");
696                 exit(ret);
697         }

699         /* add group entry */
700         if ((grps != NULL) && edit_group(logname, (char *)0, gidlist, 0)) {
701                 errmsg(M_UPDATE, "created");
702                 cleanup(logname);
703                 exit(EX_UPDATE);
704         }

706         /* update project database */
707         if ((projects != NULL) &&
708             edit_project(logname, (char *)NULL, projlist, 0)) {
709                 errmsg(M_UPDATE, "created");
710                 cleanup(logname);
711                 exit(EX_UPDATE);
712         }

714         /* create home directory */
715         if (mflag) {
716                 zfs_flags = get_default_zfs_flags();

718                 if (zflag || mflag > 1)
```



```
719                         zfs_flags |= CHANGE_ZFS_FS;
720                 else if (Zflag)
721                         zfs_flags &= ~CHANGE_ZFS_FS;
722                 ret = create_home(homedir, skel_dir, uid, gid, zfs_flags);
723         }
724         if (ret != EX_SUCCESS) {
725                 (void) edit_project(logname, (char *)NULL, (projid_t **)NULL,
726                     0);
168         if (mflag &&
169             (create_home(homedir, skel_dir, uid, gid) != EX_SUCCESS)) {
727                 (void) edit_group(logname, (char *)0, (int **)0, 1);
728                 cleanup(logname);
729                 exit(EX_HOMEDIR);
730         }

732         return (ret);
733 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    1442 Sat Sep 19 01:15:22 2015
new/usr/src/cmd/oamuser/user/useradd.dfl
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
#
# 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

# The EXCEED_TRAD indicates the action when the traditional login name
# length limit of eight characters is exceeded.  The value "warning"
# means to issue a warning message and continue.  This is the default.
# The value "error" means to issue an error message and terminate.
# The value "silent" means to continue without issuing any message.
#
EXCEED_TRAD=warning
#EXCEED_TRAD=error
#EXCEED_TRAD=silent

# The CHANGE_ZFS_FS indicates if ZFS create/destroy operations
# should be performed by default when user passes -m flag to
# userdel/usermod or -r flag to userdel
CHANGE_ZFS_FS=no
#CHANGE_ZFS_FS=yes
#endif /* ! codereview */
```

```
**********************************************************
    5630 Sat Sep 19 01:15:22 2015
new/usr/src/cmd/oamuser/user/userdel.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

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


25 /*
26  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
27  * Use is subject to license terms.
28  */

30 #include <sys/types.h>
31 #include <sys/stat.h>
32 #include <stdio.h>
33 #include <ctype.h>
34 #include <limits.h>
35 #include <pwd.h>
36 #include <project.h>
37 #include <string.h>
38 #include <sys/types.h>
39 #include <sys/stat.h>
40 #include <userdefs.h>
41 #include <stdlib.h>
42 #include <errno.h>
43 #include <unistd.h>
44 #include <strings.h>
45 #include "users.h"
46 #include "messages.h"
47 #include "funcs.h"

49 /*
50  *  userdel [-r [-z|Z]] login
49 /*****************************************************************************
50  *  userdel [-r] login
51  *
52  *      This command deletes user logins.  Arguments are:
53  *
54  *      -r - when given, this option removes home directory & its contents
55  *
56  *      login - a string of printable chars except colon (:)
57  */
57  *****************************************************************************/
```



```
59 extern int check_perm(), isbusy(), get_default_zfs_flags();
59 extern int check_perm(), isbusy();
60 extern int rm_files(), call_passmgmt(), edit_group();

62 static char *logname;                  /* login name to delete */
63 static char *nargv[20];         /* arguments for execvp of passmgmt */

65 char *cmdname;

67 int
68 main(int argc, char **argv)
69 {
70         int ch, ret = 0, rflag = 0, zflag = 0, Zflag = 0;
71         int zfs_flags = 0, argindex, tries;
70         int ch, ret = 0, rflag = 0, argindex, tries;
72         struct passwd *pstruct;
73         struct stat statbuf;
74 #ifndef att
75         FILE *pwf;              /* fille ptr for opened passwd file */
76 #endif
77         char *usertype = NULL;
78         int rc;

80         cmdname = argv[0];

82         if (geteuid() != 0) {
83                 errmsg(M_PERM_DENIED);
84                 exit(EX_NO_PERM);
81         if( geteuid() != 0 ) {
82                 errmsg( M_PERM_DENIED );
83                 exit( EX_NO_PERM );
85         }

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

90         while ((ch = getopt(argc, argv, "rzZ")) != EOF) {
91                 switch (ch) {
89         while( (ch = getopt(argc, argv, "r")) != EOF ) {
90                 switch(ch) {
92                         case 'r':
93                                 rflag++;
94                                 break;
95                         case 'Z':
96                                 Zflag++;
97                                 break;
98                         case 'z':
99                                 zflag++;
100                                break;
101 #endif /* ! codereview */
102                         case '?':
103                                 if (is_role(usertype))
104                                         errmsg(M_DRUSAGE);
 94                                         errmsg( M_DRUSAGE );
105                                 else
106                                         errmsg(M_DUSAGE);
107                                 exit(EX_SYNTAX);
 96                                         errmsg( M_DUSAGE );
 97                                 exit( EX_SYNTAX );
108                 }
109         }

111         if (optind != argc - 1) {
101         if( optind != argc - 1 ) {
112                 if (is_role(usertype))
113                         errmsg(M_DRUSAGE);
```

```
                        errmsg( M_DRUSAGE );
                else
                        errmsg(M_DUSAGE);
                exit(EX_SYNTAX);
        }

        if (((!rflag) && (Zflag || zflag)) || (Zflag && zflag)) {
                if (is_role(usertype))
                        errmsg(M_DRUSAGE);
                else
                        errmsg(M_DUSAGE);
                exit(EX_SYNTAX);
                        errmsg( M_DUSAGE );
                exit( EX_SYNTAX );
        }

        logname = argv[optind];

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

        fclose(pwf);
#endif

        if (pstruct == NULL) {
                errmsg(M_EXIST, logname);
                exit(EX_NAME_NOT_EXIST);
                errmsg( M_EXIST, logname );
                exit( EX_NAME_NOT_EXIST );
        }

        if (isbusy(logname)) {
                errmsg(M_BUSY, logname, "remove");
                exit(EX_BUSY);
        if( isbusy(logname) ) {
                errmsg( M_BUSY, logname, "remove" );
                exit( EX_BUSY );
        }

        /* that's it for validations - now do the work */
        /* set up arguments to  passmgmt in nargv array */
        nargv[0] = PASSMGMT;
        nargv[1] = "-d";        /* delete */
        argindex = 2;           /* next argument */

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

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

        /* remove home directory */
        if (rflag) {
```



```
        if( rflag ) {
                /* Check Permissions */
                if (stat(pstruct->pw_dir, &statbuf)) {
                if( stat( pstruct->pw_dir, &statbuf ) ) {
                        errmsg(M_OOPS, "find status about home directory",
                            strerror(errno));
                        exit(EX_HOMEDIR);
                        exit( EX_HOMEDIR );
                }

                if (check_perm(statbuf, pstruct->pw_uid, pstruct->pw_gid,
                    S_IWOTH|S_IXOTH) != 0) {
                        errmsg(M_NO_PERM, logname, pstruct->pw_dir);
                        exit(EX_HOMEDIR);
                }
                zfs_flags = get_default_zfs_flags();
                if (zflag)
                        zfs_flags |= CHANGE_ZFS_FS;
                else if (Zflag)
                        zfs_flags &= ~CHANGE_ZFS_FS;
                if( check_perm( statbuf, pstruct->pw_uid, pstruct->pw_gid,
                    S_IWOTH|S_IXOTH ) != 0 ) {
                        errmsg( M_NO_PERM, logname, pstruct->pw_dir );
                        exit( EX_HOMEDIR );
                }

                if (rm_files(pstruct->pw_dir, logname, zfs_flags) != EX_SUCCESS)
                        exit(EX_HOMEDIR);
                if( rm_files(pstruct->pw_dir, logname) != EX_SUCCESS )
                        exit( EX_HOMEDIR );
        }

        /* now call passmgmt */
        ret = PEX_FAILED;
        for (tries = 3; ret != PEX_SUCCESS && tries--; ) {
                switch (ret = call_passmgmt(nargv)) {
        for( tries = 3; ret != PEX_SUCCESS && tries--; ) {
                switch( ret = call_passmgmt( nargv ) ) {
                case PEX_SUCCESS:
                        ret = edit_group(logname, (char *)0, (int **)0, 1);
                        if (ret != EX_SUCCESS)
                                errmsg(M_UPDATE, "deleted");
                        ret = edit_group( logname, (char *)0, (int **)0, 1 );
                        if( ret != EX_SUCCESS )
                                errmsg( M_UPDATE, "deleted" );
                        break;

                case PEX_BUSY:
                        break;

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

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

```
                                errmsg( M_DUSAGE );
                        exit( EX_SYNTAX );
                        break;

                case PEX_BADUID:
                /*
                 * uid is used - shouldn't happen but print message anyway
                 */
                        errmsg(M_UID_USED, pstruct->pw_uid);
                        exit(EX_ID_EXISTS);
                        /* uid is used - shouldn't happen but print message anyw
                        errmsg( M_UID_USED, pstruct->pw_uid );
                        exit( EX_ID_EXISTS );
                        break;

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

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

/*
 * Now, remove this user from all project entries
 */

        rc = edit_project(logname, (char *)0, (projid_t **)0, 1);
        if (rc != EX_SUCCESS) {
                errmsg(M_UPDATE, "modified");
                exit(rc);
        }

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

unchanged_portion_omitted_

```
**********************************************************
   16251 Sat Sep 19 01:15:22 2015
new/usr/src/cmd/oamuser/user/usermod.c
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
/*
 * 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   */

/*
 * Copyright (c) 2013 RackTop Systems.
 */

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

        cmdname = argv[0];

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

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

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



```
                        break;
                case 'm':
                        mflag++;
                        flag++;
                        fail_if_busy = B_TRUE;
                        break;
                case 'o':
                        oflag++;
                        flag++;
                        fail_if_busy = B_TRUE;
                        break;
                case 'p':
                        projects = optarg;
                        flag++;
                        break;
                case 's':
                        shell = optarg;
                        flag++;
                        break;
                case 'u':
                        uidstr = optarg;
                        flag++;
                        fail_if_busy = B_TRUE;
                        break;
                case 'Z':
                        Zflag++;
                        break;
                case 'z':
                        zflag++;
                        break;
#endif /* ! codereview */
                case 'A':
                        change_key(USERATTR_AUTHS_KW, optarg);
                        flag++;
                        break;
                case 'P':
                        change_key(USERATTR_PROFILES_KW, optarg);
                        flag++;
                        break;
                case 'R':
                        change_key(USERATTR_ROLES_KW, optarg);
                        flag++;
                        break;
                case 'K':
                        change_key(NULL, optarg);
                        flag++;
                        break;
                default:
                case '?':
                        if (is_role(usertype))
                                errmsg(M_MRUSAGE);
                        else
                                errmsg(M_MUSAGE);
                        exit(EX_SYNTAX);
                }

        if (((!mflag) && (zflag || Zflag)) || (zflag && Zflag) ||
            (mflag > 1 && (zflag || Zflag))) {
                if (is_role(usertype))
                        errmsg(M_ARUSAGE);
                else
                        errmsg(M_AUSAGE);
                exit(EX_SYNTAX);
        }
```

```
#endif /* ! codereview */
        if (optind != argc - 1 || flag == 0) {
                if (is_role(usertype))
                        errmsg(M_MRUSAGE);
                else
                        errmsg(M_MUSAGE);
                exit(EX_SYNTAX);
        }

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

        logname = argv[optind];

        /* Determine whether the account is a role or not */
        if ((ua = getusernam(logname)) == NULL ||
            (val = kva_match(ua->attr, USERATTR_TYPE_KW)) == NULL ||
            strcmp(val, USERATTR_TYPE_NONADMIN_KW) != 0)
                isrole = 0;
        else
                isrole = 1;

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

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

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

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

        fclose(pwf);
```



```
#endif

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

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

        pstruct = passwd_cpmalloc(pw);

        /*
         * We can't modify a logged in user if any of the following
         * are being changed:
         * uid (-u & -o), group (-g), home dir (-m), loginname (-l).
         * If none of those are specified it is okay to go ahead
         * some types of changes only take effect on next login, some
         * like authorisations and profiles take effect instantly.
         * One might think that -K type=role should require that the
         * user not be logged in, however this would make it very
         * difficult to make the root account a role using this command.
         */
        if (isbusy(logname)) {
                if (fail_if_busy) {
                        errmsg(M_BUSY, logname, "change");
                        exit(EX_BUSY);
                }
                warningmsg(WARN_LOGGED_IN, logname);
        }

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

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

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

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

        if (uidstr) {
                /* convert uidstr to integer */
```

```
410                 errno = 0;
411                 uid = (uid_t)strtol(uidstr, &ptr, (int)10);
412                 if (*ptr || errno == ERANGE) {
413                         errmsg(M_INVALID, uidstr, "user id");
414                         exit(EX_BADARG);
415                 }

417                 if (uid != pstruct->pw_uid) {
418                         switch (valid_uid(uid, NULL)) {
419                         case NOTUNIQUE:
420                                 if (!oflag) {
421                                         /* override not specified */
422                                         errmsg(M_UID_USED, uid);
423                                         exit(EX_ID_EXISTS);
424                                 }
425                                 break;
426                         case RESERVED:
427                                 errmsg(M_RESERVED, uid);
428                                 break;
429                         case TOOBIG:
430                                 errmsg(M_TOOBIG, "uid", uid);
431                                 exit(EX_BADARG);
432                                 break;
433                         }

435                         call_pass = 1;

437                 } else {
438                         /* uid's the same, so don't change anything */
439                         uidstr = NULL;
440                         oflag = 0;
441                 }

443         } else uid = pstruct->pw_uid;

445         if (group) {
446                 switch (valid_group(group, &g_ptr, &warning)) {
447                 case INVALID:
448                         errmsg(M_INVALID, group, "group id");
449                         exit(EX_BADARG);
450                         /*NOTREACHED*/
451                 case TOOBIG:
452                         errmsg(M_TOOBIG, "gid", group);
453                         exit(EX_BADARG);
454                         /*NOTREACHED*/
455                 case UNIQUE:
456                         errmsg(M_GRP_NOTUSED, group);
457                         exit(EX_NAME_NOT_EXIST);
458                         /*NOTREACHED*/
459                 case RESERVED:
460                         gid = (gid_t)strtol(group, &ptr, (int)10);
461                         errmsg(M_RESERVED_GID, gid);
462                         break;
463                 }
464                 if (warning)
465                         warningmsg(warning, group);

467                 if (g_ptr != NULL)
468                         gid = g_ptr->gr_gid;
469                 else
470                         gid = pstruct->pw_gid;

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



```
477         } else gid = pstruct->pw_gid;

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

485         if (projects && *projects) {
486                 if (! (projlist = valid_lproject(projects)))
487                         exit(EX_BADARG);
488         } else
489                 projlist = (projid_t **)0;

491         if (dir) {
492                 if (REL_PATH(dir)) {
493                         errmsg(M_RELPATH, dir);
494                         exit(EX_BADARG);
495                 }
496                 if (strcmp(pstruct->pw_dir, dir) == 0) {
497                         /* home directory is the same so ignore dflag & mflag */
498                         dir = NULL;
499                         mflag = 0;
500                 } else call_pass = 1;
501         }

503         if (mflag) {
504                 if (stat(dir, &statbuf) == 0) {
505                         /* Home directory exists */
506                         if (check_perm(statbuf, pstruct->pw_uid,
507                             pstruct->pw_gid, S_IWOTH|S_IXOTH) != 0) {
508                                 errmsg(M_NO_PERM, logname, dir);
509                                 exit(EX_NO_PERM);
510                         }

512                 } else {
513                         zfs_flags = get_default_zfs_flags();
514                         if (zflag || mflag > 1)
515                                 zfs_flags |= CHANGE_ZFS_FS;
516                         else if (Zflag)
517                                 zfs_flags &= ~CHANGE_ZFS_FS;
518                         ret = create_home(dir, NULL, uid, gid, zfs_flags);
519                 }
235                 } else ret = create_home(dir, NULL, uid, gid);

521                 if (ret == EX_SUCCESS)
522                         ret = move_dir(pstruct->pw_dir, dir,
523                             logname, zfs_flags);
238                         ret = move_dir(pstruct->pw_dir, dir, logname);

525                 if (ret != EX_SUCCESS)
526                         exit(ret);
527         }

529         if (shell) {
530                 if (REL_PATH(shell)) {
531                         errmsg(M_RELPATH, shell);
532                         exit(EX_BADARG);
533                 }
534                 if (strcmp(pstruct->pw_shell, shell) == 0) {
535                         /* ignore s option if shell is not different */
536                         shell = NULL;
537                 } else {
538                         if (stat(shell, &statbuf) < 0 ||
539                             (statbuf.st_mode & S_IFMT) != S_IFREG ||
```

```
540                             (statbuf.st_mode & 0555) != 0555) {

542                                 errmsg(M_INVALID, shell, "shell");
543                                 exit(EX_BADARG);
544                         }

546                         call_pass = 1;
547                 }
548         }

550         if (comment) {
551                 /* ignore comment if comment is not changed */
552                 if (strcmp(pstruct->pw_comment, comment))
553                         call_pass = 1;
554                 else
555                         comment = NULL;
556         }

558         /* inactive string is a positive integer */
559         if (inactstr) {
560                 /* convert inactstr to integer */
561                 inact = (int)strtol(inactstr, &ptr, 10);
562                 if (*ptr || inact < 0) {
563                         errmsg(M_INVALID, inactstr, "inactivity period");
564                         exit(EX_BADARG);
565                 }
566                 call_pass = 1;
567         }

569         /* expiration string is a date, newer than today */
570         if (expire) {
571                 if (*expire &&
572                     valid_expire(expire, (time_t *)0) == INVALID) {
573                         errmsg(M_INVALID, expire, "expiration date");
574                         exit(EX_BADARG);
575                 }
576                 call_pass = 1;
577         }

579         if (nkeys > 0)
580                 call_pass = 1;

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

584         if (grps) {
585                 /* redefine login's supplentary group memberships */
586                 ret = edit_group(logname, new_logname, gidlist, 1);
587                 if (ret != EX_SUCCESS) {
588                         errmsg(M_UPDATE, "modified");
589                         exit(ret);
590                 }
591         }
592         if (projects) {
593                 ret = edit_project(logname, (char *)NULL, projlist, 0);
594                 if (ret != EX_SUCCESS) {
595                         errmsg(M_UPDATE, "modified");
596                         exit(ret);
597                 }
598         }


601         if (!call_pass) exit(ret);

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



```
607         argindex = 0;
608         nargv[argindex++] = PASSMGMT;
609         nargv[argindex++] = "-m";       /* modify */

611         if (comment) {  /* comment */
612                 nargv[argindex++] = "-c";
613                 nargv[argindex++] = comment;
614         }

616         if (dir) {
617                 /* flags for home directory */
618                 nargv[argindex++] = "-h";
619                 nargv[argindex++] = dir;
620         }

622         if (group) {
623                 /* set gid flag */
624                 nargv[argindex++] = "-g";
625                 (void) sprintf(gidstring, "%u", gid);
626                 nargv[argindex++] = gidstring;
627         }

629         if (shell) {    /* shell */
630                 nargv[argindex++] = "-s";
631                 nargv[argindex++] = shell;
632         }

634         if (inactstr) {
635                 nargv[argindex++] = "-f";
636                 nargv[argindex++] = inactstr;
637         }

639         if (expire) {
640                 nargv[argindex++] = "-e";
641                 nargv[argindex++] = expire;
642         }

644         if (uidstr) {   /* set uid flag */
645                 nargv[argindex++] = "-u";
646                 (void) sprintf(uidstring, "%u", uid);
647                 nargv[argindex++] = uidstring;
648         }

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

652         if (new_logname) {      /* redefine login name */
653                 nargv[argindex++] = "-l";
654                 nargv[argindex++] = new_logname;
655         }

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

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

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

666         /* now call passmgmt */
667         ret = PEX_FAILED;
668         for (tries = 3; ret != PEX_SUCCESS && tries--; ) {
669                 switch (ret = call_passmgmt(nargv)) {
670                 case PEX_SUCCESS:
671                 case PEX_BUSY:
```

```
672                         break;

674                 case PEX_HOSED_FILES:
675                         errmsg(M_HOSED_FILES);
676                         exit(EX_INCONSISTENT);
677                         break;

679                 case PEX_SYNTAX:
680                 case PEX_BADARG:
681                         /* should NEVER occur that passmgmt usage is wrong */
682                         if (is_role(usertype))
683                                 errmsg(M_MRUSAGE);
684                         else
685                                 errmsg(M_MUSAGE);
686                         exit(EX_SYNTAX);
687                         break;

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

695                 case PEX_BADNAME:
696                         /* invalid loname */
697                         errmsg(M_USED, logname);
698                         exit(EX_NAME_EXISTS);
699                         break;

701                 default:
702                         errmsg(M_UPDATE, "modified");
703                         exit(ret);
704                         break;
705                 }
706         }
707         if (tries == 0) {
708                 errmsg(M_UPDATE, "modified");
709         }

711         exit(ret);
712         /*NOTREACHED*/
713 }
```

unchanged_portion_omitted_

```
**********************************************************
   13350 Sat Sep 19 01:15:22 2015
new/usr/src/man/man1m/useradd.1m
293 useradd/del/mod should be ZFS-aware
**********************************************************
'\" te
.\" Copyright (c) 2013 Gary Mills
.\" Copyright (c) 2008 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 USERADD 1M "Apr 16, 2013"
.SH NAME
useradd \- administer a new user login on the system
.SH SYNOPSIS
.LP
.nf
\fBuseradd\fR [\fB-A\fR \fIauthorization\fR [,\fIauthorization...\fR]]
     [\fB-b\fR \fIbase_dir\fR] [\fB-c\fR \fIcomment\fR] [\fB-d\fR \fIdir\fR] [\f
     [\fB-f\fR \fIinactive\fR] [\fB-g\fR \fIgroup\fR] [\fB-G\fR \fIgroup\fR [,\f
     [\fB-K\fR \fIkey=value\fR] [\fB-m\fR [\fB-z|-Z\fR] [\fB-k\fR \fIskel_dir\fR
     [\fB-K\fR \fIkey=value\fR] [\fB-m\fR [\fB-k\fR \fIskel_dir\fR]] [\fB-p\fR \
     [\fB-P\fR \fIprofile\fR [,\fIprofile...\fR]] [\fB-R\fR \fIrole\fR [,\fIrole
     [\fB-s\fR \fIshell\fR] [\fB-u\fR \fIuid\fR [\fB-o\fR]] \fIlogin\fR
.fi

.LP
.nf
\fBuseradd\fR \fB-D\fR [\fB-A\fR \fIauthorization\fR [,\fIauthorization...\fR]]
     [\fB-b\fR \fIbase_dir\fR] [\fB-s\fR \fIshell\fR [\fB-k\fR \fIskel_dir\fR]]
     [\fB-f\fR \fIinactive\fR] [\fB-g\fR \fIgroup\fR] [\fB-K\fR \fIkey=value\fR]
     [\fB-P\fR \fIprofile\fR [,\fIprofile...\fR]] [\fB-R\fR \fIrole\fR [,\fIrole
.fi

.SH DESCRIPTION
.sp
.LP
\fBuseradd\fR adds a new user to the \fB/etc/passwd\fR and \fB/etc/shadow\fR
and \fB/etc/user_attr\fR files. The \fB-A\fR and \fB-P\fR options respectively
assign authorizations and profiles to the user. The \fB-R\fR option assigns
roles to a user. The \fB-p\fR option associates a project with a user. The
\fB-K\fR option adds a \fIkey=value\fR pair to \fB/etc/user_attr\fR for the
user. Multiple \fIkey=value\fR pairs may be added with multiple \fB-K\fR
options.
.sp
.LP
\fBuseradd\fR also creates supplementary group memberships for the user
(\fB-G\fR option) and creates the home directory (\fB-m\fR option) for the user
if requested. The new login remains locked until the \fBpasswd\fR(1) command is
executed.
.sp
.LP
Specifying \fBuseradd\fR \fB-D\fR with the \fB-s\fR, \fB-k\fR,\fB-g\fR,
\fB-b\fR, \fB-f\fR, \fB-e\fR, \fB-A\fR, \fB-P\fR, \fB-p\fR, \fB-R\fR, or
\fB-K\fR option (or any combination of these options) sets the default values
for the respective fields. See the \fB-D\fR option, below. Subsequent
\fBuseradd\fR commands without the \fB-D\fR option use these arguments.
.sp
.LP
The system file entries created with this command have a limit of 2048
characters per line. Specifying long arguments to several options can exceed
this limit.
.sp
.LP
\fBuseradd\fR requires that usernames be in the format described in
```



```
\fBpasswd\fR(4). A warning message is displayed if these restrictions are not
met. See \fBpasswd\fR(4) for the requirements for usernames.
.LP
To change the action of \fBuseradd\fR when the traditional login name
length limit of eight characters is exceeded, edit the file
\fB/etc/default/useradd\fR by removing the \fB#\fR (pound sign) before the
appropriate \fBEXCEED_TRAD=\fR entry, and adding it before the others.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-A\fR \fIauthorization\fR\fR
.ad
.sp .6
.RS 4n
One or more comma separated authorizations defined in \fBauth_attr\fR(4). Only
a user or role who has \fBgrant\fR rights to the authorization can assign it to
an account.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR \fIbase_dir\fR\fR
.ad
.sp .6
.RS 4n
The base directory for new login home directories (see the \fB-d\fR option
below. When a new user account is being created, \fIbase_dir\fR must already
exist unless the \fB-m\fR option or the \fB-d\fR option is also specified.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIcomment\fR\fR
.ad
.sp .6
.RS 4n
Any text string. It is generally a short description of the login, and is
currently used as the field for the user's full name. This information is
stored in the user's \fB/etc/passwd\fR entry.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR \fIdir\fR\fR
.ad
.sp .6
.RS 4n
The home directory of the new user. It defaults to
\fIbase_dir\fR/\fIaccount_name\fR, where \fIbase_dir\fR is the base directory
for new login home directories and \fIaccount_name\fR is the new login name.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR\fR
.ad
.sp .6
.RS 4n
```

```
Display the default values for \fBgroup\fR, \fBbase_dir\fR, \fBskel_dir\fR,
\fBshell\fR, \fBinactive\fR, \fBexpire\fR, \fBproj\fR, \fBprojname\fR and
\fBkey=value\fR pairs. When used with the \fB-g\fR, \fB-b\fR, \fB-f\fR,
\fB-e\fR, \fB-A\fR, \fB-P\fR, \fB-p\fR, \fB-R\fR, or \fB-K\fR options, the
\fB-D\fR option sets the default values for the specified fields. The default
values are:
.sp
.ne 2
.na
\fBgroup\fR
.ad
.sp .6
.RS 4n
\fBother\fR (\fBGID\fR of 1)
.RE

.sp
.ne 2
.na
\fBbase_dir\fR
.ad
.sp .6
.RS 4n
\fB/home\fR
.RE

.sp
.ne 2
.na
\fBskel_dir\fR
.ad
.sp .6
.RS 4n
\fB/etc/skel\fR
.RE

.sp
.ne 2
.na
\fBshell\fR
.ad
.sp .6
.RS 4n
\fB/bin/sh\fR
.RE

.sp
.ne 2
.na
\fBinactive\fR
.ad
.sp .6
.RS 4n
\fB0\fR
.RE

.sp
.ne 2
.na
\fBexpire\fR
.ad
.sp .6
.RS 4n
null
.RE
```



```
.sp
.ne 2
.na
\fBauths\fR
.ad
.sp .6
.RS 4n
null
.RE

.sp
.ne 2
.na
\fBprofiles\fR
.ad
.sp .6
.RS 4n
null
.RE

.sp
.ne 2
.na
\fBproj\fR
.ad
.sp .6
.RS 4n
\fB3\fR
.RE

.sp
.ne 2
.na
\fBprojname\fR
.ad
.sp .6
.RS 4n
\fBdefault\fR
.RE

.sp
.ne 2
.na
\fBkey=value (pairs defined in \fBuser_attr\fR(4)\fR
.ad
.sp .6
.RS 4n
not present
.RE

.sp
.ne 2
.na
\fBroles\fR
.ad
.sp .6
.RS 4n
null
.RE

.RE

.sp
.ne 2
.na
\fB\fB-e\fR \fIexpire\fR\fR
```

```
.ad
.sp .6
.RS 4n
Specify the expiration date for a login. After this date, no user will be able
to access this login. The expire option argument is a date entered using one of
the date formats included in the template file \fB/etc/datemsk\fR. See
\fBgetdate\fR(3C).
.sp
If the date format that you choose includes spaces, it must be quoted. For
example, you can enter \fB10/6/90\fR or \fBOctober 6, 1990\fR. A null value
(\fB"" \fR) defeats the status of the expired date. This option is useful for
creating temporary logins.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR \fIinactive\fR\fR
.ad
.sp .6
.RS 4n
The maximum number of days allowed between uses of a login ID before that
\fBID\fR is declared invalid. Normal values are positive integers. A value of
\fB0\fR defeats the status.
.RE

.sp
.ne 2
.na
\fB\fB-g\fR \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
An existing group's integer \fBID\fR or character-string name. Without the
\fB-D\fR option, it defines the new user's primary group membership and
defaults to the default group. You can reset this default value by invoking
\fBuseradd\fR \fB-D\fR \fB-g\fR \fIgroup\fR. GIDs 0-99 are reserved for
allocation by the Solaris Operating System.
.RE

.sp
.ne 2
.na
\fB\fB-G\fR \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
An existing group's integer \fBID\fR or character-string name. It defines the
new user's supplementary group membership. Duplicates between \fIgroup\fR with
the \fB-g\fR and \fB-G\fR options are ignored. No more than \fBNGROUPS_MAX\fR
groups can be specified. GIDs 0-99 are reserved for allocation by the Solaris
Operating System.
.RE

.sp
.ne 2
.na
\fB\fB-K\fR \fIkey=value\fR\fR
.ad
.sp .6
.RS 4n
A \fIkey=value\fR pair to add to the user's attributes. Multiple \fB-K\fR
options may be used to add multiple \fIkey=value\fR pairs. The generic \fB-K\fR
option with the appropriate key may be used instead of the specific implied key
options (\fB-A\fR, \fB-P\fR, \fB-R\fR, \fB-p\fR). See \fBuser_attr\fR(4) for a
list of valid \fIkey=value\fR pairs. The "type" key is not a valid key for this
```



```
option. Keys may not be repeated.
.RE

.sp
.ne 2
.na
\fB\fB-k\fR \fIskel_dir\fR\fR
.ad
.sp .6
.RS 4n
A directory that contains skeleton information (such as \fB\&.profile\fR) that
can be copied into a new user's home directory. This directory must already
exist. The system provides the \fB/etc/skel\fR directory that can be used for
this purpose.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR [\fB-z|-Z\fR]
\fB\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Create the new user's home directory if it does not already exist. If the
directory already exists, it must have read, write, and execute permissions by
\fIgroup\fR, where \fIgroup\fR is the user's primary group.
CHANGE_ZFS_FS option in /etc/default/useradd file determines if ZFS filesystem
will be created for new user. If this option is set to yes and parent directory
of user's home directory is ZFS filesystem mount point, a new ZFS filesystem is
created. \fB-z\fR and \fB-Z\fR options allow overwrite default behavior.
If \fB-z\fR option is specified, \fBuseradd\fR tries to create new file system
for user. If \fB-Z\fR option is specified, new file system is not created.
#endif /* ! codereview */
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.sp .6
.RS 4n
This option allows a \fBUID\fR to be duplicated (non-unique).
.RE

.sp
.ne 2
.na
\fB\fB-P\fR \fIprofile\fR\fR
.ad
.sp .6
.RS 4n
One or more comma-separated execution profiles defined in \fBprof_attr\fR(4).
.RE

.sp
.ne 2
.na
\fB\fB-p\fR \fIprojname\fR\fR
.ad
.sp .6
.RS 4n
Name of the project with which the added user is associated. See the
\fIprojname\fR field as defined in \fBproject\fR(4).
.RE
```

```
.sp
.ne 2
.na
\fB\fB-R\fR \fIrole\fR\fR
.ad
.sp .6
.RS 4n
One or more comma-separated execution profiles defined in \fBuser_attr\fR(4).
Roles cannot be assigned to other roles.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIshell\fR\fR
.ad
.sp .6
.RS 4n
Full pathname of the program used as the user's shell on login. It defaults to
an empty field causing the system to use \fB/bin/sh\fR as the default. The
value of \fIshell\fR must be a valid executable file.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR \fIuid\fR\fR
.ad
.sp .6
.RS 4n
The \fBUID\fR of the new user. This \fBUID\fR must be a non-negative decimal
integer below \fBMAXUID\fR as defined in \fB<sys/param.h>\fR\&. The \fBUID\fR
defaults to the next available (unique) number above the highest number
currently assigned. For example, if \fBUID\fRs 100, 105, and 200 are assigned,
the next default \fBUID\fR number will be 201. \fBUID\fRs \fB0\fR-\fB99\fR are
reserved for allocation by the Solaris Operating System.
.RE

.SH FILES
.sp
.LP
\fB/etc/default/useradd\fR
.sp
.LP
\fB/etc/datemsk\fR
.sp
.LP
\fB/etc/passwd\fR
.sp
.LP
\fB/etc/shadow\fR
.sp
.LP
\fB/etc/group\fR
.sp
.LP
\fB/etc/skel\fR
.sp
.LP
\fB/usr/include/limits.h\fR
.sp
.LP
\fB/etc/user_attr\fR
.SH ATTRIBUTES
.sp
```



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

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
.TE

.SH SEE ALSO
.sp
.LP
\fBpasswd\fR(1), \fBprofiles\fR(1), \fBroles\fR(1), \fBusers\fR(1B),
\fBgroupadd\fR(1M), \fBgroupdel\fR(1M), \fBgroupmod\fR(1M), \fBgrpck\fR(1M),
\fBlogins\fR(1M), \fBpwck\fR(1M), \fBuserdel\fR(1M), \fBusermod\fR(1M),
\fBgetdate\fR(3C), \fBauth_attr\fR(4), \fBpasswd\fR(4), \fBprof_attr\fR(4),
\fBproject\fR(4), \fBuser_attr\fR(4), \fBattributes\fR(5)
.SH DIAGNOSTICS
.sp
.LP
In case of an error, \fBuseradd\fR prints an error message and exits with a
non-zero status.
.sp
.LP
The following indicates that \fBlogin\fR specified is already in use:
.sp
.in +2
.nf
UX: useradd: ERROR: login is already in use. Choose another.
.fi
.in -2
.sp

.sp
.LP
The following indicates that the \fIuid\fR specified with the \fB-u\fR option
is not unique:
.sp
.in +2
.nf
UX: useradd: ERROR: uid \fIuid\fR is already in use. Choose another.
.fi
.in -2
.sp

.sp
.LP
The following indicates that the \fIgroup\fR specified with the \fB-g\fR option
is already in use:
.sp
.in +2
.nf
UX: useradd: ERROR: group \fIgroup\fR does not exist. Choose another.
.fi
.in -2
.sp

.sp
.LP
The following indicates that the \fIuid\fR specified with the \fB-u\fR option
is in the range of reserved \fBUID\fRs (from \fB0\fR-\fB99\fR):
```

```
 516 .sp
 517 .in +2
 518 .nf
 519 UX: useradd: WARNING: uid \fIuid\fR is reserved.
 520 .fi
 521 .in -2
 522 .sp

 524 .sp
 525 .LP
 526 The following indicates that the \fIuid\fR specified with the \fB-u\fR option
 527 exceeds \fBMAXUID\fR as defined in \fB<sys/param.h>\fR:
 528 .sp
 529 .in +2
 530 .nf
 531 UX: useradd: ERROR: uid \fIuid\fR is too big. Choose another.
 532 .fi
 533 .in -2
 534 .sp

 536 .sp
 537 .LP
 538 The following indicates that the \fB/etc/passwd\fR or \fB/etc/shadow\fR files
 539 do not exist:
 540 .sp
 541 .in +2
 542 .nf
 543 UX: useradd: ERROR: Cannot update system files - login cannot be created.
 544 .fi
 545 .in -2
 546 .sp

 548 .SH NOTES
 476 .sp
 549 .LP
 550 The \fBuseradd\fR utility adds definitions to only the local \fB/etc/group\fR,
 551 \fBetc/passwd\fR, \fB/etc/passwd\fR, \fB/etc/shadow\fR, \fB/etc/project\fR, and
 552 \fB/etc/user_attr\fR files. If a network name service such as \fBNIS\fR or
 553 \fBNIS+\fR is being used to supplement the local \fB/etc/passwd\fR file with
 554 additional entries, \fBuseradd\fR cannot change information supplied by the
 555 network name service. However \fBuseradd\fR will verify the uniqueness of the
 556 user name (or role) and user id and the existence of any group names specified
 557 against the external name service.
```

```
**********************************************************
    4066 Sat Sep 19 01:15:22 2015
new/usr/src/man/man1m/userdel.1m
293 useradd/del/mod should be ZFS-aware
**********************************************************
```

```
   1 '\" te
   2 .\"  Copyright 1989 AT&T  Copyright (c) 1999,
   3 .\" Sun Microsystems, Inc.  All Rights Reserved
   4 .\" The contents of this file are subject to the terms of the Common Development
   5 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   6 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   7 .TH USERDEL 1M "Sep 8, 1999"
   8 .SH NAME
   9 userdel \- delete a user's login from the system
  10 .SH SYNOPSIS
  11 .LP
  12 .nf
  13 \fBuserdel\fR [\fB-r\fR [\fB-z|-Z\fR]] \fIlogin\fR
  13 \fBuserdel\fR [\fB-r\fR] \fIlogin\fR
  14 .fi

  16 .SH DESCRIPTION
  17 .sp
  17 .LP
  18 The \fBuserdel\fR utility deletes a user account from the system and makes the
  19 appropriate account-related changes to the system file and file system.
  20 .SH OPTIONS
  22 .sp
  21 .LP
  22 The following options are supported:
  23 .sp
  24 .ne 2
  25 .na
  26 \fB\fB-r\fR\fR [\fB-z|-Z\fR]
  28 \fB\fB-r\fR\fR
  27 .ad
  28 .RS 6n
  29 Remove the user's home directory from the system. This directory must exist.
  30 The files and directories under the home directory will no longer be accessible
  31 following successful execution of the command.
  32 CHANGE_ZFS_FS option in /etc/default/useradd file determines if ZFS filesystem
  33 underlying user home directory will be destroyed. If user's home directory is a
  34 ZFS file system and CHANGE_ZFS_FS option is set to yes, the filesystem
  35 will be destroyed. \fB-z\fR and \fB-Z\fR options allow overwrite default behavio
  36 If \fB-z\fR option is specified, \fBuserdel\fR tries to destroy file system.
  37 If \fB-Z\fR option is specified, file system is not destroyed.
  38 #endif /* ! codereview */
  39 .RE

  41 .SH OPERANDS
  34 .sp
  42 .LP
  43 The following operands are supported:
  44 .sp
  45 .ne 2
  46 .na
  47 \fB\fBlogin\fR\fR\fR
  48 .ad
  49 .RS 9n
  50 An existing login name to be deleted.
  51 .RE

  53 .SH EXIT STATUS
  47 .sp
  54 .LP
  55 The following exit values are returned:
```



```
  56 .sp
  57 .ne 2
  58 .na
  59 \fB\fB0\fR\fR
  60 .ad
  61 .RS 6n
  62 Successful completion.
  63 .RE

  65 .sp
  66 .ne 2
  67 .na
  68 \fB\fB2\fR\fR
  69 .ad
  70 .RS 6n
  71 Invalid command syntax. A usage message for the \fBuserdel\fR command is
  72 displayed.
  73 .RE

  75 .sp
  76 .ne 2
  77 .na
  78 \fB\fB6\fR\fR
  79 .ad
  80 .RS 6n
  81 The account to be removed does not exist.
  82 .RE

  84 .sp
  85 .ne 2
  86 .na
  87 \fB\fB8\fR\fR
  88 .ad
  89 .RS 6n
  90 The account to be removed is in use.
  91 .RE

  93 .sp
  94 .ne 2
  95 .na
  96 \fB\fB10\fR\fR
  97 .ad
  98 .RS 6n
  99 Cannot update the \fB/etc/group\fR or \fB/etc/user_attr\fR file but the login
 100 is removed from the \fB/etc/passwd\fR file.
 101 .RE

 103 .sp
 104 .ne 2
 105 .na
 106 \fB\fB12\fR\fR
 107 .ad
 108 .RS 6n
 109 Cannot remove or otherwise modify the home directory.
 110 .RE

 112 .SH FILES
 113 .ne 2
 114 .na
 115 \fB\fB/etc/default/useradd\fR\fR
 116 .ad
 117 .RS 18n
 118 useradd, usermod and userdel configuration file
 119 .RE

 121 #endif /* ! codereview */
```

```
.sp
.ne 2
.na
\fB\fB/etc/passwd\fR\fR
.ad
.RS 18n
system password file
.RE

.sp
.ne 2
.na
\fB\fB/etc/shadow\fR\fR
.ad
.RS 18n
system file contain users' encrypted passwords and related information
.RE

.sp
.ne 2
.na
\fB\fB/etc/group\fR\fR
.ad
.RS 18n
system file containing group definitions
.RE

.sp
.ne 2
.na
\fB\fB/etc/user_attr\fR\fR
.ad
.RS 18n
system file containing additional user attributes
.RE

.SH SEE ALSO
.sp
.LP
\fBauths\fR(1), \fBpasswd\fR(1), \fBprofiles\fR(1), \fBroles\fR(1),
\fBusers\fR(1B), \fBgroupadd\fR(1M), \fBgroupdel\fR(1M), \fBgroupmod\fR(1M),
\fBlogins\fR(1M), \fBroleadd\fR(1M), \fBrolemod\fR(1M), \fBuseradd\fR(1M),
\fBuserdel\fR(1M), \fBusermod\fR(1M), \fBpasswd\fR(4), \fBprof_attr\fR(4),
\fBuser_attr\fR(4), \fBattributes\fR(5)
.SH NOTES
.sp
.LP
The \fBuserdel\fR utility only deletes an account definition that is in the
local \fB/etc/group\fR, \fB/etc/passwd\fR, \fB/etc/shadow\fR, and
\fB/etc/user_attr\fR file. file. If a network name service such as \fBNIS\fR or
\fBNIS+\fR is being used to supplement the local \fB/etc/passwd\fR file with
additional entries, \fBuserdel\fR cannot change information supplied by the
network name service.
```

```
**********************************************************
   13107 Sat Sep 19 01:15:23 2015
new/usr/src/man/man1m/usermod.1m
293 useradd/del/mod should be ZFS-aware
**********************************************************
   1 '\" te
   2 .\" Copyright 1989 AT&T Copyright (c) 2004, 2009, Sun Microsystems, Inc. All Rig
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH USERMOD 1M "Feb 22, 2008"
   7 .SH NAME
   8 usermod \- modify a user's login information on the system
   9 .SH SYNOPSIS
  10 .LP
  11 .nf
  12 \fBusermod\fR [\fB-u\fR \fIuid\fR [\fB-o\fR]] [\fB-g\fR \fIgroup\fR] [\fB-G\fR \
  13      [\fB-d\fR \fIdir\fR [\fB-m\fR [\fB-z|-Z\fR]]] [\fB-s\fR \fIshell\fR] [\fB-c
  13      [\fB-d\fR \fIdir\fR [\fB-m\fR]] [\fB-s\fR \fIshell\fR] [\fB-c\fR \fIcomment
  14      [\fB-f\fR \fIinactive\fR] [\fB-e\fR \fIexpire\fR]
  15      [\fB-A\fR \fIauthorization\fR [, \fIauthorization\fR]]
  16      [\fB-P\fR \fIprofile\fR [, \fIprofile\fR]] [\fB-R\fR \fIrole\fR [, \fIrole\
  17      [\fB-K\fR \fIkey=value\fR] \fIlogin\fR
  18 .fi

  20 .SH DESCRIPTION
  21 .sp
  21 .LP
  22 The \fBusermod\fR utility modifies a user's login definition on the system. It
  23 changes the definition of the specified login and makes the appropriate
  24 login-related system file and file system changes.
  25 .sp
  26 .LP
  27 The system file entries created with this command have a limit of 512
  28 characters per line. Specifying long arguments to several options might exceed
  29 this limit.
  30 .SH OPTIONS
  32 .sp
  31 .LP
  32 The following options are supported:
  33 .sp
  34 .ne 2
  35 .na
  36 \fB\fB-A\fR \fIauthorization\fR\fR
  37 .ad
  38 .sp .6
  39 .RS 4n
  40 One or more comma separated authorizations as defined in \fBauth_attr\fR(4).
  41 Only a user or role who has \fBgrant\fR rights to the \fBauthorization\fR can
  42 assign it to an account. This replaces any existing authorization setting. If
  43 no authorization list is specified, the existing setting is removed.
  44 .RE

  46 .sp
  47 .ne 2
  48 .na
  49 \fB\fB-c\fR \fIcomment\fR\fR
  50 .ad
  51 .sp .6
  52 .RS 4n
  53 Specify a comment string. \fIcomment\fR can be any text string. It is generally
  54 a short description of the login, and is currently used as the field for the
  55 user's full name. This information is stored in the user's \fB/etc/passwd\fR
  56 entry.
  57 .RE
```



```
  59 .sp
  60 .ne 2
  61 .na
  62 \fB\fB-d\fR \fIdir\fR\fR
  63 .ad
  64 .sp .6
  65 .RS 4n
  66 Specify the new home directory of the user. It defaults to
  67 \fIbase_dir/login\fR, where \fIbase_dir\fR is the base directory for new login
  68 home directories, and \fBlogin\fR is the new login.
  69 .RE

  71 .sp
  72 .ne 2
  73 .na
  74 \fB\fB-e\fR \fIexpire\fR\fR
  75 .ad
  76 .sp .6
  77 .RS 4n
  78 Specify the expiration date for a login. After this date, no user will be able
  79 to access this login. The expire option argument is a date entered using one of
  80 the date formats included in the template file \fB/etc/datemsk\fR. See
  81 \fBgetdate\fR(3C).
  82 .sp
  83 For example, you may enter \fB10/6/90\fR or \fBOctober 6, 1990\fR. A value of
  84 \fB'' ''\fR defeats the status of the expired date.
  85 .RE

  87 .sp
  88 .ne 2
  89 .na
  90 \fB\fB-f\fR \fIinactive\fR\fR
  91 .ad
  92 .sp .6
  93 .RS 4n
  94 Specify the maximum number of days allowed between uses of a login \fBID\fR
  95 before that login \fBID\fR is declared invalid. Normal values are positive
  96 integers. A value of \fB0\fR defeats the status.
  97 .RE

  99 .sp
 100 .ne 2
 101 .na
 102 \fB\fB-g\fR \fIgroup\fR\fR
 103 .ad
 104 .sp .6
 105 .RS 4n
 106 Specify an existing group's integer \fBID\fR or character-string name. It
 107 redefines the user's primary group membership.
 108 .RE

 110 .sp
 111 .ne 2
 112 .na
 113 \fB\fB-G\fR \fIgroup\fR\fR
 114 .ad
 115 .sp .6
 116 .RS 4n
 117 Specify an existing group's integer "ID" "," or character string name. It
 118 redefines the user's supplementary group membership. Duplicates between
 119 \fIgroup\fR with the \fB-g\fR and \fB-G\fR options are ignored. No more than
 120 \fBNGROUPS_UMAX\fR groups may be specified as defined in \fB<param.h>\fR\&.
 121 .RE

 123 .sp
 124 .ne 2
```

```
.na
\fB\fB-K\fR \fIkey=value\fR\fR
.ad
.sp .6
.RS 4n
Replace existing or add to a user's \fIkey=value\fR pair attributes. Multiple
\fB-K\fR options can be used to replace or add multiple \fIkey=value\fR pairs.
However, keys must not be repeated. The generic \fB-K\fR option with the
appropriate key can be used instead of the specific implied key options
(\fB-A\fR, \fB-P\fR, \fB-R\fR, \fB-p\fR). See \fBuser_attr\fR(4) for a list of
valid \fIkey\fRs. Values for these keys are usually found in man pages or other
sources related to those keys. For example, see \fBproject\fR(4) for guidance
on values for the \fBproject\fR key. Use the command \fBppriv\fR(1) with the
\fB-v\fR and \fB-l\fR options for a list of values for the keys
\fBdefaultpriv\fR and \fBlimitpriv\fR.
.sp
The keyword \fBtype\fR can be specified with the value \fBrole\fR or the value
\fBnormal\fR.  When using the value \fBrole\fR, the account changes from a
normal user to a role; using the value \fBnormal\fR keeps the account a normal
user.
.sp
As a \fBrole\fR account, no roles (\fB-R\fR or \fIroles=value\fR) can be
present.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR \fInew_logname\fR\fR
.ad
.sp .6
.RS 4n
Specify the new login name for the user. See \fBpasswd\fR(4) for the
requirements for usernames.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR [\fB-z|-Z\fR]
\fI\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Move the user's home directory to the new directory specified with the \fB-d\fR
option. If the directory already exists, it must have permissions
read/write/execute by \fIgroup\fR, where \fIgroup\fR is the user's primary
group.
CHANGE_ZFS_FS option in /etc/default/useradd file determines if ZFS filesystem
will be created or destroyed during this action. If this option is set to yes
and parent directory of user's home directory is ZFS filesystem mount point, a
new ZFS filesystem is created. If old user's home directory is a ZFS file system
and CHANGE_ZFS_FS is set to yes, the filesystem will be destroyed.
\fB-z\fB and \fB-Z\fR options allow overwrite default behavior. If \fB-z\fR
option is specified, \fBusermod\fR tries to create new file system and destroy t
old one. If \fB-Z\fR option is specified, new filesystem is not created and the
one is not destroyed.
#endif /* ! codereview */
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.sp .6
```



```
.RS 4n
This option allows the specified \fBUID\fR to be duplicated (non-unique).
.RE

.sp
.ne 2
.na
\fB\fB-P\fR \fIprofile\fR\fR
.ad
.sp .6
.RS 4n
One or more comma-separated rights profiles defined in \fBprof_attr\fR(4). This
replaces any existing profile setting in \fBuser_attr\fR(4). If an empty
profile list is specified, the existing setting is removed.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR \fIrole\fR\fR
.ad
.sp .6
.RS 4n
One or more comma-separated roles (see \fBroleadd\fR(1M)). This replaces any
existing role setting. If no role list is specified, the existing setting is
removed.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIshell\fR\fR
.ad
.sp .6
.RS 4n
Specify the full pathname of the program that is used as the user's shell on
login. The value of \fIshell\fR must be a valid executable file.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR \fIuid\fR\fR
.ad
.sp .6
.RS 4n
Specify a new \fBUID\fR for the user. It must be a non-negative decimal integer
less than \fBMAXUID\fR as defined in \fB<param.h>\fR\&. The \fBUID\fR
associated with the user's home directory is not modified with this option; a
user will not have access to their home directory until the \fBUID\fR is
manually reassigned using \fBchown\fR(1).
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fBlogin\fR\fR
.ad
.sp .6
.RS 4n
An existing login name to be modified.
.RE
```

```
.SH EXAMPLES
.LP
\fBExample 1 \fRAssigning Privileges to a User
.sp
.LP
The following command adds the privilege that affects high resolution times to
a user’s initial, inheritable set of privileges.

.sp
.in +2
.nf
# \fBusermod -K defaultpriv=basic,proc_clock_highres jdoe\fR
.fi
.in -2
.sp

.sp
.LP
This command results in the following entry in \fBuser_attr\fR:

.sp
.in +2
.nf
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
.fi
.in -2

.LP
\fBExample 2 \fRRemoving a Privilege from a User’s Limit Set
.sp
.LP
The following command removes the privilege that allows the specified user to
create hard links to directories and to unlink directories.

.sp
.in +2
.nf
# \fBusermod -K limitpriv=all,!sys_linkdir jdoe\fR
.fi
.in -2
.sp

.sp
.LP
This command results in the following entry in \fBuser_attr\fR:

.sp
.in +2
.nf
jdoe::::type=normal;defaultpriv=basic,limitpriv=all,!sys_linkdir
.fi
.in -2

.LP
\fBExample 3 \fRRemoving a Privilege from a User’s Basic Set
.sp
.LP
The following command removes the privilege that allows the specified user to
examine processes outside the user’s session.

.sp
.in +2
.nf
# \fBusermod -K defaultpriv=basic,!proc_session jdoe\fR
.fi
```



```
.in -2
.sp

.sp
.LP
This command results in the following entry in \fBuser_attr\fR:

.sp
.in +2
.nf
jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
.fi
.in -2

.LP
\fBExample 4 \fRAssigning a Role to a User
.sp
.LP
The following command assigns a role to a user. The role must have been created
prior to this command, either through use of the Solaris Management Console GUI
or through \fBroleadd\fR(1M).

.sp
.in +2
.nf
# \fBusermod -R mailadm jdoe\fR
.fi
.in -2
.sp

.sp
.LP
This command results in the following entry in \fBuser_attr\fR:

.sp
.in +2
.nf
jdoe::::type=normal;roles=mailadm;defaultpriv=basic;limitpriv=all
.fi
.in -2

.LP
\fBExample 5 \fRRemoving All Profiles from a User
.sp
.LP
The following command removes all profiles that were granted to a user
directly. The user will still have any rights profiles that are granted by
means of the \fBPROFS_GRANTED\fR key in \fBpolicy.conf\fR(4).

.sp
.in +2
.nf
# \fBusermod -P "" jdoe\fR
.fi
.in -2
.sp

.SH EXIT STATUS
.sp
.LP
In case of an error, \fBusermod\fR prints an error message and exits with one
of the following values:
.sp
.ne 2
.na
\fB\fB2\fR\fR
```

```
.ad
.sp .6
.RS 4n
The command syntax was invalid. A usage message for the \fBusermod\fR command
is displayed.
.RE

.sp
.ne 2
.na
\fB\fB3\fR\fR
.ad
.sp .6
.RS 4n
An invalid argument was provided to an option.
.RE

.sp
.ne 2
.na
\fB\fB4\fR\fR
.ad
.sp .6
.RS 4n
The \fIuid\fR given with the \fB-u\fR option is already in use.
.RE

.sp
.ne 2
.na
\fB\fB5\fR\fR
.ad
.sp .6
.RS 4n
The password files contain an error. \fBpwconv\fR(1M) can be used to correct
possible errors. See \fBpasswd\fR(4).
.RE

.sp
.ne 2
.na
\fB\fB6\fR\fR
.ad
.sp .6
.RS 4n
The login to be modified does not exist, the \fIgroup\fR does not exist, or the
login shell does not exist.
.RE

.sp
.ne 2
.na
\fB\fB8\fR\fR
.ad
.sp .6
.RS 4n
The login to be modified is in use.
.RE

.sp
.ne 2
.na
\fB\fB9\fR\fR
.ad
.sp .6
.RS 4n
```



```
The \fInew_logname\fR is already in use.
.RE

.sp
.ne 2
.na
\fB\fB10\fR\fR
.ad
.sp .6
.RS 4n
Cannot update the \fB/etc/group\fR or \fB/etc/user_attr\fR file. Other update
requests will be implemented.
.RE

.sp
.ne 2
.na
\fB\fB11\fR\fR
.ad
.sp .6
.RS 4n
Insufficient space to move the home directory (\fB-m\fR option). Other update
requests will be implemented.
.RE

.sp
.ne 2
.na
\fB\fB12\fR\fR
.ad
.sp .6
.RS 4n
Unable to complete the move of the home directory to the new home directory.
.RE

.SH FILES
.ne 2
.na
\fB\fB/etc/default/useradd\fR\fR
.ad
.sp .6
.RS 4n
useradd, usermod and userdel configuration file
.RE

#endif /* ! codereview */
.sp
.ne 2
.na
\fB\fB/etc/group\fR\fR
.ad
.sp .6
.RS 4n
system file containing group definitions
.RE

.sp
.ne 2
.na
\fB\fB/etc/datemsk\fR\fR
.ad
.sp .6
.RS 4n
system file of date formats
.RE
```

```
.sp
.ne 2
.na
\fB\fB/etc/passwd\fR\fR
.ad
.sp .6
.RS 4n
system password file
.RE

.sp
.ne 2
.na
\fB\fB/etc/shadow\fR\fR
.ad
.sp .6
.RS 4n
system file containing users' encrypted passwords and related information
.RE

.sp
.ne 2
.na
\fB\fB/etc/user_attr\fR\fR
.ad
.sp .6
.RS 4n
system file containing additional user and role attributes
.RE

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

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
.TE

.SH SEE ALSO
.sp
.LP
\fBchown\fR(1), \fBpasswd\fR(1), \fBusers\fR(1B), \fBgroupadd\fR(1M),
\fBgroupdel\fR(1M), \fBgroupmod\fR(1M), \fBlogins\fR(1M), \fBpwconv\fR(1M),
\fBroleadd\fR(1M), \fBroledel\fR(1M), \fBrolemod\fR(1M), \fBuseradd\fR(1M),
\fBuserdel\fR(1M), \fBgetdate\fR(3C), \fBauth_attr\fR(4), \fBpasswd\fR(4),
\fBpolicy.conf\fR(4), \fBprof_attr\fR(4), \fBuser_attr\fR(4),
\fBattributes\fR(5)
.SH NOTES
.sp
.LP
The \fBusermod\fR utility modifies \fBpasswd\fR definitions only in the local
\fB/etc/passwd\fR and \fB/etc/shadow\fR files. If a network nameservice such as
\fBNIS\fR or \fBNIS+\fR is being used to supplement the local files with
additional entries, \fBusermod\fR cannot change information supplied by the
network nameservice. However \fBusermod\fR will verify the uniqueness of user
name and user \fBID\fR against the external nameservice.
.sp
.LP
```



```
The \fBusermod\fR utility uses the \fB/etc/datemsk\fR file, available with
SUNWaccr, for date formatting.
```