```
**********************************************************
    1269 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/Makefile
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/projadd/Makefile
#

include ../Makefile.cmd

SUBDIRS=        projadd projmod projdel projtest
PROGS= projadd projmod projdel
USRSBINPROGS= $(PROGS:%=$(ROOTUSRSBIN)/%)
POFILES= $(PROGS:%=%.po)

all :=          TARGET = all
install :=      TARGET = install
clean :=        TARGET = clean
clobber :=      TARGET = clobber
lint :=         TARGET = lint
_msg :=         TARGET = _msg
# No msg catalog here.
POFILE=

CLOBBERFILES += $(PROGS)

.KEEP_STATE:

all install lint clean clobber _msg: $(SUBDIRS)
all: $(PROGS)

$(SUBDIRS): FRC
        @cd $@; pwd; $(MAKE) $(MFLAGS) $(TARGET)
install : all .WAIT $(USRSBINPROGS)

FRC:
clean lint:

_msg: $(MSGDOMAIN) $(POFILES)
        $(CP) $(POFILES) $(MSGDOMAIN)
```



```
$(MSGDOMAIN):
        $(INS.dir)

clobber: clean
        $(RM) $(PROG) $(CLOBBERFILES)

$(ROOTUSRSBIN)/% : %
        $(INS.file)
```

```
**********************************************************
    1105 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/Makefile.projadd
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/stat/Makefile.stat

PROJADD = $(SRC)/cmd/projadd
PROJADDCOMMONDIR = $(PROJADD)/common

COMMON_OBJS = projent.o attrib.o util.o resctl.o lst.o
COMMON_SRCS = $(COMMON_OBJS:%.o=$(PROJADDCOMMONDIR)/%.c)
```

```
**********************************************************
   33008 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/attrib.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <locale.h>
#include <stddef.h>
#include <limits.h>
#include <rctl.h>
#include <regex.h>
#include <ctype.h>

#include <sys/debug.h>

#include "attrib.h"
#include "resctl.h"
#include "util.h"

#define BOSTR_REG_EXP   "^"
#define EOSTR_REG_EXP   "$"
#define EQUAL_REG_EXP   "="
#define STRN0_REG_EXP   "(.*)"
#define IDENT_REG_EXP   "[[:alpha:]][[:alnum:]_.-]*"
#define INTNM_REG_EXP   "[[:digit:]]+"
#define SIGAC_REG_EXP   "sig(nal)?(=.*)?"
#define SIGHD_REG_EXP   "(signal|sig)"
#define SIGVL_REG_EXP   "((([[:digit:]]+)|((SIG)?([[:upper:]]+)([+-][123])?))"
#define SIGNL_REG_EXP   SIGHD_REG_EXP EQUAL_REG_EXP SIGVL_REG_EXP
#define STOCK_REG_EXP   "([[:upper:]]{1,5}(.[[:upper:]]{1,5})?,)?"
#define ATTRB_REG_EXP   "(" STOCK_REG_EXP IDENT_REG_EXP ")"
#define ATVAL_REG_EXP   ATTRB_REG_EXP EQUAL_REG_EXP STRN0_REG_EXP

#define TO_EXP(X)       BOSTR_REG_EXP X EOSTR_REG_EXP

#define POOLN_EXP       TO_EXP(IDENT_REG_EXP)
#define INTNM_EXP       TO_EXP(INTNM_REG_EXP)
#define SIGAC_EXP       TO_EXP(SIGAC_REG_EXP)
#define SIGNL_EXP       TO_EXP(SIGNL_REG_EXP)
#define ATTRB_EXP       TO_EXP(ATTRB_REG_EXP)
#define ATVAL_EXP       TO_EXP(ATVAL_REG_EXP)

#define MAX_OF(X, Y)    (((X) > (Y)) ? (X) : (Y))

#define SEQU(X, Y)              (strcmp((X), (Y)) == 0)
#define SIN1(X, S1)             ((SEQU((X), (S1))))
#define SIN2(X, S1, S2)         ((SEQU((X), (S1))) || (SIN1((X), (S2))))
#define SIN3(X, S1, S2, S3)     ((SEQU((X), (S1))) || (SIN2((X), (S2), (S3))))

#define ATT_VAL_TYPE_NULL       0
#define ATT_VAL_TYPE_VALUE      1
#define ATT_VAL_TYPE_LIST       2

#define ATT_ALLOC()             attrib_alloc()
#define ATT_VAL_ALLOC(T, V)     attrib_val_alloc((T), (V))
#define ATT_VAL_ALLOC_NULL()    ATT_VAL_ALLOC(ATT_VAL_TYPE_NULL, NULL)
#define ATT_VAL_ALLOC_VALUE(V)  ATT_VAL_ALLOC(ATT_VAL_TYPE_VALUE, (V))
#define ATT_VAL_ALLOC_LIST(L)   ATT_VAL_ALLOC(ATT_VAL_TYPE_LIST, (L))

typedef struct attrib_val_s {
```



```
        int att_val_type;
        /*LINTED*/
        union {
                char *att_val_value;
                lst_t *att_val_values;
        };
} attrib_val_t;

typedef struct attrib_s {
        char *att_name;
        attrib_val_t *att_value;
} attrib_t;


attrib_t *attrib_alloc();
attrib_val_t *attrib_val_alloc(int, void *);
char *attrib_val_tostring(attrib_val_t *, boolean_t);

int
attrib_validate_rctl(attrib_t *att, resctlrule_t *rule, lst_t *errlst)
{
        int ret = 0;
        char *atname = att->att_name;
        attrib_val_t *atv, *atval = att->att_value;
        attrib_val_t *priv, *val, *action;
        char *vpriv, *vval, *vaction, *sigstr;
        int atv_type = atval->att_val_type;
        char *str;
        int i, j, k;

        int nmatch;
        uint8_t rpriv, raction, sigval;
        uint64_t rmax;
        regex_t pintexp, sigacexp, signlexp;
        regmatch_t *mat;

        int nonecount, denycount, sigcount;

        if (regcomp(&pintexp, INTNM_EXP, REG_EXTENDED) != 0) {
                        util_add_errmsg(errlst, gettext(
                            "Failed to compile regex for pos. integer"));
                        return (1);
        }

        if (regcomp(&sigacexp, SIGAC_EXP, REG_EXTENDED) != 0) {
                        util_add_errmsg(errlst, gettext(
                            "Failed to compile regex for sigaction"));
                        regfree(&pintexp);
                        return (1);
        }

        if (regcomp(&signlexp, SIGNL_EXP, REG_EXTENDED) != 0) {
                        util_add_errmsg(errlst, gettext(
                            "Failed to compile regex for signal"));
                        regfree(&pintexp);
                        regfree(&sigacexp);
                        return (1);
        }

        nmatch = signlexp.re_nsub + 1;
        mat = util_safe_zmalloc(nmatch * sizeof (regmatch_t));

        if (atv_type != ATT_VAL_TYPE_LIST) {
                util_add_errmsg(errlst, gettext(
                    "rctl \"%s\" missing value"), atname);
                ret = 1;
```

```
128                 goto out;
129         }

131         for (i = 0; i < lst_size(atval->att_val_values); i++) {
132                 atv = lst_at(atval->att_val_values, i);
133                 if (atv->att_val_type != ATT_VAL_TYPE_LIST) {
134                         if ((str = attrib_val_tostring(atv, B_FALSE)) != NULL) {
135                                 util_add_errmsg(errlst, gettext(
136                                     "rctl \"%s\" value \"%s\" "
137                                     "should be in ()"), atname, str);
138                                 free(str);
139                         } else {
140                                 util_add_errmsg(errlst, gettext(
141                                     "rctl \"%s\" value "
142                                     "should be in ()"), atname);
143                         }
144                         ret = 1;
145                         continue;
146                 }
147                 /* Values should be in the form (priv, val, actions) */
148                 if (lst_size(atv->att_val_values) < 3) {
149                         util_add_errmsg(errlst, gettext(
150                             "rctl \"%s\" value should be in the form "
151                             "(priv, val, action[,...])[,....]"), atname);
152                         ret = 1;
153                         continue;
154                 }

156                 priv = lst_at(atv->att_val_values, 0);
157                 val = lst_at(atv->att_val_values, 1);
158                 /* actions = el[2], el[3], ... */

160                 vpriv = priv->att_val_value;
161                 rpriv = rule->resctl_privs;

163                 if (priv->att_val_type != ATT_VAL_TYPE_VALUE) {
164                         util_add_errmsg(errlst, gettext(
165                             "rctl \"%s\" invalid privilege"), atname);
166                         ret = 1;
167                 } else if (!SIN3(vpriv, "basic", "privileged", "priv")) {
168                         util_add_errmsg(errlst, gettext(
169                             "rctl \"%s\" unknown privilege \"%s\""),
170                             atname, vpriv);
171                         ret = 1;
172                 } else if (!(
173                     ((rpriv & RESCTL_PRIV_PRIVE) &&
174                     SEQU(vpriv, "priv")) ||
175                     ((rpriv & RESCTL_PRIV_PRIVD) &&
176                     SEQU(vpriv, "privileged")) ||
177                     ((rpriv & RESCTL_PRIV_BASIC) &&
178                     SEQU(vpriv, "basic")))) {
179                         util_add_errmsg(errlst, gettext(
180                             "rctl \"%s\" privilege not allowed \"%s\""),
181                             atname, vpriv);
182                         ret = 1;
183                 }

185                 vval = val->att_val_value;
186                 rmax = rule->resctl_max;

188                 if (val->att_val_type != ATT_VAL_TYPE_VALUE) {
189                         util_add_errmsg(errlst, gettext(
190                             "rctl \"%s\" invalid value"), atname);
191                         ret = 1;
192                 } else if (regexec(&pintexp, vval, 0, NULL, 0) != 0) {
193                         util_add_errmsg(errlst, gettext(
```



```
194                             "rctl \"%s\" value \"%s\" is not an integer"),
195                             atname, vval);
196                         ret = 1;
197                 } else if (strtoll(vval, NULL, 0) > rmax) {
198                         util_add_errmsg(errlst, gettext(
199                             "rctl \"%s\" value \"%s\" exceeds system limit"),
200                             atname, vval);
201                         ret = 1;
202                 }

204                 nonecount = 0;
205                 denycount = 0;
206                 sigcount = 0;

208                 for (j = 2; j < lst_size(atv->att_val_values); j++) {
209                         action = lst_at(atv->att_val_values, j);

211                         if (action->att_val_type != ATT_VAL_TYPE_VALUE) {
212                                 util_add_errmsg(errlst, gettext(
213                                     "rctl \"%s\" invalid action"), atname);
214                                 ret = 1;
215                                 continue;
216                         }

218                         vaction = action->att_val_value;

220                         if (regexec(&sigacexp, vaction, 0, NULL, 0) != 0 &&
221                             !SIN2(vaction, "none", "deny")) {
222                                 util_add_errmsg(errlst, gettext(
223                                     "rctl \"%s\" unknown action \"%s\""),
224                                     atname, vaction);
225                                 ret = 1;
226                                 continue;
227                         }

229                         raction = rule->resctl_action;
230                         if (!(((raction & RESCTL_ACTN_SIGN) &&
231                             regexec(&sigacexp, vaction, 0, NULL, 0) == 0) ||
232                             ((raction & RESCTL_ACTN_NONE) &&
233                             SEQU(vaction, "none")) ||
234                             ((raction & RESCTL_ACTN_DENY) &&
235                             SEQU(vaction, "deny")))) {
236                                 util_add_errmsg(errlst, gettext(
237                                     "rctl \"%s\" action not allowed \"%s\""),
238                                     atname, vaction);
239                                 ret = 1;
240                                 continue;
241                         }

243                         if (SEQU(vaction, "none")) {
244                                 if (nonecount >= 1) {
245                                         util_add_errmsg(errlst, gettext(
246                                             "rctl \"%s\" duplicate action "
247                                             "none"), atname);
248                                         ret = 1;
249                                 }
250                                 nonecount++;
251                                 continue;
252                         }

254                         if (SEQU(vaction, "deny")) {
255                                 if (denycount >= 1) {
256                                         util_add_errmsg(errlst, gettext(
257                                             "rctl \"%s\" duplicate action "
258                                             "deny"), atname);
259                                         ret = 1;
```

```
260                             }
261                             denycount++;
262                             continue;
263                     }

265                     /* At this point, the action must be signal. */
266                     if (sigcount >= 1) {
267                             util_add_errmsg(errlst, gettext(
268                                 "rctl \"%s\" duplicate action sig"),
269                                 atname);
270                             ret = 1;
271                     }
272                     sigcount++;

274                     /*
275                      * Make sure signal is correct format, on of:
276                      * sig=##
277                      * signal=##
278                      * sig=SIGXXX
279                      * signal=SIGXXX
280                      * sig=XXX
281                      * signal=XXX
282                      */

284                     if (regexec(&signlexp, vaction, nmatch, mat, 0) != 0 ||
285                         (sigstr = util_substr(&signlexp, mat, vaction, 2))
286                         == NULL) {
287                             util_add_errmsg(errlst, gettext(
288                                 "rctl \"%s\" invalid signal \"%s\""),
289                                 atname, vaction);
290                             ret = 1;
291                             continue;
292                     }

294                     /* Our version of sigstr =~ s/SIG// */
295                     if (strstr(sigstr, "SIG") != NULL)
296                             sigstr = strstr(sigstr, "SIG") + 3;

298                     sigval = 0;
299                     for (k = 0; k < SIGS_CNT; k++) {
300                             if (SEQU(sigs[k].sig, sigstr))
301                                     sigval = sigs[k].mask;
302                     }
303                     free(sigstr);

305                     if (sigval == 0) {
306                             util_add_errmsg(errlst, gettext(
307                                 "rctl \"%s\" invalid signal \"%s\""),
308                                 atname, vaction);
309                             ret = 1;
310                             continue;
311                     }

313                     if (!(sigval & rule->resctl_sigs)) {
314                             util_add_errmsg(errlst, gettext(
315                                 "rctl \"%s\" signal not allowed \"%s\""),
316                                 atname, vaction);
317                             ret = 1;
318                             continue;
319                     }
320             }
321             if (nonecount > 0 && (denycount > 0 || sigcount > 0)) {
322                     util_add_errmsg(errlst, gettext(
323                         "rctl \"%s\" action \"none\" specified with "
324                         "other actions"),
325                         atname);
```



```
326                             ret = 1;
327                     }
328             }

330     out:
331             free(mat);
332             regfree(&signlexp);
333             regfree(&sigacexp);
334             regfree(&pintexp);
335             return (ret);
336     }

338     int
339     attrib_validate(attrib_t *att, lst_t *errlst)
340     {
341             int ret = 0;
342             char *atname = att->att_name;
343             attrib_val_t *atv = att->att_value;
344             int atv_type = atv->att_val_type;
345             char *str, *eptr;
346             long long ll;

348             resctl_info_t rinfo;
349             resctlrule_t rrule;

351             regex_t poolnexp;
352             if (regcomp(&poolnexp, POOLN_EXP, REG_EXTENDED) != 0) {
353                     util_add_errmsg(errlst, gettext(
354                         "Failed to compile poolname regular expression:"));
355                     return (1);
356             }

358             if (SEQU(atname, "task.final")) {
359                     if (atv_type != ATT_VAL_TYPE_NULL) {
360                             util_add_errmsg(errlst, gettext(
361                                 "task.final should not have value"));
362                             ret = 1;
363                     }
364             } else if (SEQU(atname, "rcap.max-rss")) {
365                     if (atv_type == ATT_VAL_TYPE_NULL) {
366                             util_add_errmsg(errlst, gettext(
367                                 "rcap.max-rss missing value"));
368                             ret = 1;
369                     } else if (atv_type == ATT_VAL_TYPE_LIST) {
370                             util_add_errmsg(errlst, gettext(
371                                 "rcap.max-rss should have single value"));
372                             ret = 1;
373                     } else if (atv_type == ATT_VAL_TYPE_VALUE) {
374                             if ((str = attrib_val_tostring(atv, B_FALSE)) != NULL) {
375                                     ll = strtoll(str, &eptr, 0);
376                                     if (*eptr != '\0') {
377                                             util_add_errmsg(errlst, gettext(
378                                                 "rcap.max-rss is not an integer "
379                                                 "value: \"%s\""), str);
380                                             ret = 1;
381                                     } else if (ll == LLONG_MIN && errno == ERANGE) {
382                                             util_add_errmsg(errlst, gettext(
383                                                 "rcap.max-rss too small"));
384                                             ret = 1;
385                                     } else if (ll == LLONG_MAX && errno == ERANGE) {
386                                             util_add_errmsg(errlst, gettext(
387                                                 "rcap.max-rss too large"));
388                                             ret = 1;
389                                     } else if (ll < 0) {
390                                             util_add_errmsg(errlst, gettext(
391                                                 "rcap.max-rss should not have "
```

```
392                                         "negative value: \"%s\""), str);
393                                 ret = 1;
394                         }
395                         free(str);
396                 } else {
397                         util_add_errmsg(errlst, gettext(
398                             "rcap.max-rss has invalid value"));
399                         ret = 1;
400                 }
401         }
402 } else if (SEQU(atname, "project.pool")) {
403         if (atv_type == ATT_VAL_TYPE_NULL) {
404                 util_add_errmsg(errlst, gettext(
405                     "project.pool missing value"));
406                 ret = 1;
407         } else if (atv_type == ATT_VAL_TYPE_LIST) {
408                 util_add_errmsg(errlst, gettext(
409                     "project.pool should have single value"));
410                 ret = 1;
411         } else if (atv_type == ATT_VAL_TYPE_VALUE) {
412                 if ((str = attrib_val_tostring(atv, B_FALSE)) != NULL) {
413                         if (regexec(&poolnexp, str, 0, NULL, 0) != 0) {
414                                 util_add_errmsg(errlst, gettext(
415                                     "project.pool: invalid pool "
416                                     "name \"%s\""), str);
417                                 ret = 1;
418                         } else if (resctl_pool_exist(str) != 0) {
419                                 util_add_errmsg(errlst, gettext(
420                                     "project.pool: pools not enabled  "
421                                     "or pool does not exist: \"%s\""),
422                                     str);
423                                 ret = 1;
424                         }
425                         free(str);
426                 } else {
427                         util_add_errmsg(errlst, gettext(
428                             "project.pool has invalid value "));
429                         ret = 1;
430                 }
431         }
432 } else if (resctl_get_info(atname, &rinfo) == 0) {
433         resctl_get_rule(&rinfo, &rrule);
434         if (attrib_validate_rctl(att, &rrule, errlst) != 0) {
435                 ret = 1;
436         }
437 }

439 regfree(&poolnexp);
440 return (ret);
441 }

443 int
444 attrib_validate_lst(lst_t *attribs, lst_t *errlst)
445 {
446         int i, j;
447         attrib_t *att;
448         char **atnames, **atlast;
449         char *atname;
450         int ret = 0;

452         atlast = atnames = util_safe_zmalloc(
453             (lst_size(attribs) + 1) * sizeof (char *));
454         for (i = 0; i < lst_size(attribs); i++) {
455                 att = lst_at(attribs, i);

457                 /* Validate this attribute */
```



```
458                 if (attrib_validate(att, errlst) != 0)
459                         ret = 1;
460                 /* Make sure it is not duplicated */
461                 for (j = 0; (atname = atnames[j]) != NULL; j++) {
462                         if (strcmp(atname, att->att_name) == 0) {
463                                 util_add_errmsg(errlst, gettext(
464                                     "Duplicate attributes \"%s\""), atname);
465                                 ret = 1;
466                         }
467                 }
468                 /*
469                  * Add it to the attribute name to the
470                  * temporary list if not found
471                  */
472                 if (atname == NULL) {
473                         *atlast++ = att->att_name;
474                 }
475         }
476         free(atnames);
477         return (ret);
478 }

480 attrib_t *
481 attrib_alloc()
482 {
483         return (util_safe_zmalloc(sizeof (attrib_t)));
484 }

486 attrib_val_t *
487 attrib_val_alloc(int type, void *val)
488 {
489         attrib_val_t *ret;

491         ret = util_safe_malloc(sizeof (attrib_val_t));
492         ret->att_val_type = type;
493         ret->att_val_value = val;
494         return (ret);
495 }

497 char *
498 attrib_val_tostring(attrib_val_t *val, boolean_t innerlist)
499 {
500         char *ret = NULL;
501         char *vstring;
502         int i;
503         attrib_val_t *v;
504         switch (val->att_val_type) {
505                 case ATT_VAL_TYPE_NULL:
506                         return (util_safe_strdup(""));
507                 case ATT_VAL_TYPE_VALUE:
508                         return (util_safe_strdup(val->att_val_value));
509                 case ATT_VAL_TYPE_LIST:
510                         /* Only innerlists need to be betweeen ( and ) */
511                         if (innerlist)
512                                 ret = UTIL_STR_APPEND1(ret, "(");
513                         for (i = 0; i < lst_size(val->att_val_values);
514                             i++) {
515                                 v = lst_at(val->att_val_values, i);
516                                 if (i > 0) {
517                                         ret = UTIL_STR_APPEND1(ret, ",");
518                                 }
519                                 if ((vstring =
520                                     attrib_val_tostring(v, B_TRUE)) == NULL) {
521                                         UTIL_FREE_SNULL(ret);
522                                         goto out;
523                                 }
```

```
524                                 ret = UTIL_STR_APPEND1(ret, vstring);
525                                 free(vstring);
526                         }
527                         if (innerlist)
528                                 ret = UTIL_STR_APPEND1(ret, ")");
529                         return (ret);
530         }

532 out:
533         return (ret);
534 }

536 char *
537 attrib_tostring(void *at)
538 {
539         attrib_t *att;
540         char *ret = NULL, *vstring;

542         att = (attrib_t *)at;
543         ret = UTIL_STR_APPEND1(ret, att->att_name);
544         if ((vstring = attrib_val_tostring(att->att_value, B_FALSE)) != NULL) {
545                 if (strlen(vstring) > 0)
546                         ret = UTIL_STR_APPEND2(ret, "=", vstring);
547                 free(vstring);
548                 return (ret);
549         }
550         UTIL_FREE_SNULL(ret);
551         return (ret);
552 }

554 char *
555 attrib_lst_tostring(lst_t *attrs)
556 {
557         int i;
558         attrib_t *att;
559         char *ret = NULL;
560         char *str;

562         ret = UTIL_STR_APPEND1(ret, "");
563         for (i = 0; i < lst_size(attrs); i++) {
564                 att = lst_at(attrs, i);

566                 if ((str = attrib_tostring(att)) != NULL) {
567                         if (i > 0)
568                                 ret = UTIL_STR_APPEND1(ret, ";");

570                         ret = UTIL_STR_APPEND1(ret, str);
571                         free(str);
572                         continue;
573                 }

575                 free(ret);
576                 return (NULL);
577         }
578         return (ret);
579 }

581 void
582 attrib_val_free(attrib_val_t *atv)
583 {
584         attrib_val_t *val;

586         if (atv->att_val_type == ATT_VAL_TYPE_VALUE) {
587                 free(atv->att_val_value);
588         } else if (atv->att_val_type == ATT_VAL_TYPE_LIST) {
589                 while (!lst_is_empty(atv->att_val_values)) {
```



```
590                         val = lst_at(atv->att_val_values, 0);
591                         (void) lst_remove(atv->att_val_values, val);
592                         attrib_val_free(val);
593                         free(val);
594                 }
595                 free(atv->att_val_values);
596         }
597 }

599 void
600 attrib_free(attrib_t *att)
601 {
602         free(att->att_name);
603         if (att->att_value != NULL) {
604                 attrib_val_free(att->att_value);
605                 free(att->att_value);
606         }
607 }
608 void
609 attrib_free_lst(lst_t *attribs)
610 {
611         attrib_t *att;

613         if (attribs == NULL)
614                 return;

616         while (!lst_is_empty(attribs)) {
617                 att = lst_at(attribs, 0);
618                 (void) lst_remove(attribs, att);
619                 attrib_free(att);
620                 free(att);
621         }
622 }

624 void
625 attrib_sort_lst(lst_t *attribs)
626 {
627         int i, j, n;
628         attrib_t *atti, *attj;

630         if (attribs == NULL)
631                 return;

633         n = lst_size(attribs);
634         for (i = 0; i < n - 1; i++) {
635                 for (j = i + 1; j < n; j++) {
636                         atti = lst_at(attribs, i);
637                         attj = lst_at(attribs, j);
638                         if (strcmp(attj->att_name, atti->att_name) < 0) {
639                                 (void) lst_replace_at(attribs, i, attj);
640                                 (void) lst_replace_at(attribs, j, atti);
641                         }
642                 }
643         }
644 }

646 void
647 attrib_val_to_list(attrib_val_t *atv)
648 {
649         void *val;
650         int type;
651         attrib_val_t *mat;

653         if (atv->att_val_type == ATT_VAL_TYPE_LIST)
654                 return;
```

```
656         val = atv->att_val_value;
657         type = atv->att_val_type;

659         atv->att_val_type = ATT_VAL_TYPE_LIST;
660         atv->att_val_values = util_safe_malloc(sizeof (lst_t));
661         lst_create(atv->att_val_values);

663         if (type == ATT_VAL_TYPE_VALUE && val != NULL) {
664                 mat = ATT_VAL_ALLOC_VALUE(val);
665                 lst_insert_tail(atv->att_val_values, mat);
666         }
667 }

669 void
670 attrib_val_append(attrib_val_t *atv, char *token)
671 {
672         attrib_val_t *nat;
673         if (atv->att_val_type == ATT_VAL_TYPE_VALUE) {
674                 /* convert this to LIST attribute */
675                 attrib_val_to_list(atv);
676         }

678         if (atv->att_val_type == ATT_VAL_TYPE_NULL) {
679                 /* convert this to VALUE attribute */
680                 atv->att_val_type = ATT_VAL_TYPE_VALUE;
681                 atv->att_val_value = util_safe_strdup(token);
682         } else if (atv->att_val_type == ATT_VAL_TYPE_LIST) {
683                 /* append token to the list */
684                 nat = ATT_VAL_ALLOC_VALUE(util_safe_strdup(token));
685                 lst_insert_tail(atv->att_val_values, nat);
686         }
687 }

689 attrib_val_t *
690 attrib_val_parse(char *values, lst_t *errlst)
691 {
692         attrib_val_t *ret = NULL;
693         attrib_val_t *at;
694         attrib_val_t *nat;
695         lst_t stk;

697         char **tokens, *token, *usedtokens, *prev;
698         int i, error, parendepth;

700         error = parendepth = 0;
701         prev = "";

703         if ((tokens = util_tokenize(values, errlst)) == NULL) {
704                 goto out1;
705         }

707         lst_create(&stk);
708         usedtokens = UTIL_STR_APPEND1(NULL, "");

710         at = ret = ATT_VAL_ALLOC_NULL();

712         for (i = 0; (token = tokens[i]) != NULL; i++) {

714                 usedtokens = UTIL_STR_APPEND1(usedtokens, token);

716                 if (SEQU(token, ",")) {
717                         if (SIN3(prev, ",", "(", "")) {
718                                 attrib_val_append(at, "");
719                         }
720                         attrib_val_to_list(at);
721                         prev = ",";
```



```
722                 } else if (SEQU(token, "(")) {
723                         if (!(SIN3(prev, "(", ",", ""))) {
724                                 util_add_errmsg(errlst, gettext(
725                                     "\"%s\" <- \"(\" unexpected"), usedtokens);
726                                 error = 1;
727                                 goto out;
728                         }

730                         switch (at->att_val_type) {
731                                 case ATT_VAL_TYPE_VALUE:
732                                         util_add_errmsg(errlst, gettext(
733                                             "\"%s\" <- \"%s\" unexpected"),
734                                             usedtokens, token);
735                                         error = 1;
736                                         goto out;
737                                 case ATT_VAL_TYPE_NULL:
738                                         /* Make is a LIST attrib */
739                                         attrib_val_to_list(at);
740                                         /*FALLTHROUGH*/
741                                 case ATT_VAL_TYPE_LIST:
742                                         /* Allocate NULL node */
743                                         nat = ATT_VAL_ALLOC_NULL();
744                                         attrib_val_to_list(nat);
745                                         lst_insert_tail(
746                                             at->att_val_values, nat);
747                                         /* push at down one level */
748                                         lst_insert_head(&stk, at);
749                                         at = nat;
750                                 break;
751                         }
752                         parendepth++;
753                         prev = "(";
754                 } else if (SEQU(token, ")")) {
755                         if (parendepth <= 0) {
756                                 util_add_errmsg(errlst, gettext(
757                                     "\"%s\" <- \")\" unexpected"), usedtokens);
758                                 error = 1;
759                                 goto out;
760                         }
761                         if (SIN2(prev, ",", "(")) {
762                                 attrib_val_append(at, "");
763                         }

765                         if (!lst_is_empty(&stk)) {
766                                 at = lst_at(&stk, 0);
767                                 (void) lst_remove(&stk, at);
768                         }
769                         parendepth--;
770                         prev = ")";
771                 } else {
772                         if (!(SIN3(prev, ",", "(", ""))) {
773                                 util_add_errmsg(errlst, gettext(
774                                     "\"%s\" <- \"%s\" unexpected"),
775                                     usedtokens, token);
776                                 error = 1;
777                                 goto out;
778                         }

780                         attrib_val_append(at, token);
781                         prev = token;
782                 }
783         }

785         if (parendepth != 0) {
786                 util_add_errmsg(errlst, gettext(
787                     "\"%s\" <- \")\" missing"),
```

```
788                     usedtokens);
789                 error = 1;
790                 goto out;
791         }
792
793         if (SIN2(prev, ",", "")) {
794                 switch (at->att_val_type) {
795                         case ATT_VAL_TYPE_NULL:
796                                 util_add_errmsg(errlst, gettext(
797                                     "\"%s\" unexpected"),
798                                     usedtokens);
799                                 error = 1;
800                                 goto out;
801                         case ATT_VAL_TYPE_VALUE:
802                         case ATT_VAL_TYPE_LIST:
803                                 attrib_val_append(at, "");
804                         break;
805                 }
806         }
807
808 out:
809         while (!lst_is_empty(&stk)) {
810                 at = lst_at(&stk, 0);
811                 (void) lst_remove(&stk, at);
812         }
813
814         util_free_tokens(tokens);
815         free(tokens);
816         free(usedtokens);
817         if (error) {
818                 attrib_val_free(ret);
819                 UTIL_FREE_SNULL(ret);
820         }
821 out1:
822         return (ret);
823 }
824
825 attrib_t *
826 attrib_parse(regex_t *attrbexp, regex_t *atvalexp, char *att, int flags,
827     lst_t *errlst)
828 {
829         int nmatch = MAX_OF(attrbexp->re_nsub, atvalexp->re_nsub) + 1;
830         attrib_t *ret = NULL;
831         attrib_val_t *retv, *atv, *atvl;
832         char *values = NULL;
833         int vidx, nidx, vlen;
834         int scale;
835
836         char *num, *mod, *unit;
837         int i;
838
839         resctl_info_t rinfo;
840         resctlrule_t rrule;
841
842         regmatch_t *mat = util_safe_malloc(nmatch * sizeof (regmatch_t));
843         ret = ATT_ALLOC();
844
845         if (regexec(attrbexp, att, attrbexp->re_nsub + 1, mat, 0) == 0) {
846                 ret->att_name = util_safe_strdup(att);
847                 ret->att_value = ATT_VAL_ALLOC_NULL();
848         } else if (regexec(atvalexp, att,
849             atvalexp->re_nsub + 1, mat, 0) == 0) {
850                 vidx = atvalexp->re_nsub;
851                 vlen = mat[vidx].rm_eo - mat[vidx].rm_so;
852                 nidx = atvalexp->re_nsub - 3;
853                 ret->att_name = util_substr(atvalexp, mat, att, nidx);
```



```
854
855                 if (vlen > 0) {
856                         values = util_substr(atvalexp, mat, att, vidx);
857                         ret->att_value = attrib_val_parse(values, errlst);
858                         free(values);
859                         if (ret->att_value == NULL) {
860                                 util_add_errmsg(errlst, gettext(
861                                     "Invalid value on attribute \"%s\""),
862                                     ret->att_name);
863                                 attrib_free(ret);
864                                 UTIL_FREE_SNULL(ret);
865                                 goto out;
866                         }
867                 } else {
868                         /* the value is an empty string */
869                         ret->att_value = ATT_VAL_ALLOC_NULL();
870                 }
871         } else {
872                 util_add_errmsg(errlst, gettext(
873                     "Invalid attribute \"%s\""), att);
874                 attrib_free(ret);
875                 UTIL_FREE_SNULL(ret);
876                 goto out;
877         }
878
879         if (!(flags & F_PAR_UNT))
880                 goto out;
881
882         if (SEQU(ret->att_name, "rcap.max-rss")) {
883                 values = attrib_val_tostring(ret->att_value, B_FALSE);
884                 if (util_val2num(values, BYTES_SCALE, errlst,
885                     &num, &mod, &unit) == 0) {
886                         attrib_val_free(ret->att_value);
887                         ret->att_value = ATT_VAL_ALLOC_VALUE(num);
888                         free(mod);
889                         free(unit);
890                 } else {
891                         attrib_free(ret);
892                         UTIL_FREE_SNULL(ret);
893                         goto out;
894                 }
895                 free(values);
896         }
897
898         if (resctl_get_info(ret->att_name, &rinfo) == 0) {
899                 resctl_get_rule(&rinfo, &rrule);
900                 retv = ret->att_value;
901
902                 switch (rrule.resctl_type) {
903                         case RESCTL_TYPE_BYTES:
904                                 scale = BYTES_SCALE;
905                                 break;
906                         case RESCTL_TYPE_SCNDS:
907                                 scale = SCNDS_SCALE;
908                                 break;
909                         case RESCTL_TYPE_COUNT:
910                                 scale = COUNT_SCALE;
911                                 break;
912                         default:
913                                 scale = UNKWN_SCALE;
914                                 break;
915                 }
916
917                 if (retv->att_val_type != ATT_VAL_TYPE_LIST)
918                         goto out;
```

```
921                 for (i = 0; i < lst_size(retv->att_val_values); i++) {
922                         atvl = atv = lst_at(retv->att_val_values, i);

924                         /*
925                          * Continue if not a list and the second value
926                          * is not a scaler value
927                          */
928                         if (atv->att_val_type != ATT_VAL_TYPE_LIST ||
929                             lst_size(atv->att_val_values) < 3 ||
930                             (atv = lst_at(atv->att_val_values, 1)) == NULL ||
931                             atv->att_val_type != ATT_VAL_TYPE_VALUE) {
932                                 continue;
933                         }
934                         values = attrib_val_tostring(atv, B_FALSE);
935                         if (util_val2num(values, scale, errlst,
936                             &num, &mod, &unit) == 0) {
937                                 attrib_val_free(atv);
938                                 atv = ATT_VAL_ALLOC_VALUE(num);
939                                 (void) lst_replace_at(atvl->att_val_values, 1,
940                                     atv);
941                                 free(mod);
942                                 free(unit);

944                         } else {
945                                 free(values);
946                                 attrib_free(ret);
947                                 UTIL_FREE_SNULL(ret);
948                                 goto out;
949                         }
950                         free(values);
951                 }
952         }

954 out:
955         free(mat);
956         return (ret);
957 }

959 lst_t *
960 attrib_parse_attributes(char *attribs, int flags, lst_t *errlst)
961 {
962         char *sattrs, *attrs, *att;
963         regex_t attrbexp, atvalexp;

965         attrib_t *natt = NULL;
966         lst_t *ret = NULL;

968         ret = util_safe_malloc(sizeof (lst_t));
969         lst_create(ret);

971         if (regcomp(&attrbexp, ATTRB_EXP, REG_EXTENDED) != 0)
972                 goto out1;
973         if (regcomp(&atvalexp, ATVAL_EXP, REG_EXTENDED) != 0)
974                 goto out2;

976         sattrs = attrs = util_safe_strdup(attribs);
977         while ((att = strsep(&attrs, ";")) != NULL) {
978                 if (*att == '\0')
979                         continue;
980                 if ((natt = attrib_parse(&attrbexp,
981                     &atvalexp, att, flags, errlst)) == NULL) {
982                         attrib_free_lst(ret);
983                         UTIL_FREE_SNULL(ret);
984                         break;
985                 }
```



```
987                 lst_insert_tail(ret, natt);
988         }

990         free(sattrs);
991         regfree(&atvalexp);
992 out2:
993         regfree(&attrbexp);
994 out1:
995         return (ret);
996 }

998 attrib_val_t *
999 attrib_val_duplicate(attrib_val_t *atv)
1000 {
1001         int i;
1002         lst_t *values;
1003         attrib_val_t *val;
1004         attrib_val_t *natv;

1006         switch (atv->att_val_type) {
1007                 case ATT_VAL_TYPE_NULL:
1008                         natv = ATT_VAL_ALLOC_NULL();
1009                 break;
1010                 case ATT_VAL_TYPE_VALUE:
1011                         natv = ATT_VAL_ALLOC_VALUE(
1012                             util_safe_strdup(atv->att_val_value));
1013                 break;
1014                 case ATT_VAL_TYPE_LIST:
1015                         values = util_safe_malloc(sizeof (lst_t));
1016                         lst_create(values);
1017                         for (i = 0; i < lst_size(atv->att_val_values);
1018                             i++) {
1019                                 val = lst_at(atv->att_val_values, i);
1020                                 lst_insert_tail(values,
1021                                     attrib_val_duplicate(val));
1022                         }
1023                         natv = ATT_VAL_ALLOC_LIST(values);
1024                 break;
1025         }

1027         return (natv);
1028 }

1030 attrib_t *
1031 attrib_duplicate(attrib_t *att)
1032 {
1033         attrib_t *natt = ATT_ALLOC();
1034         natt->att_name = util_safe_strdup(att->att_name);
1035         natt->att_value = attrib_val_duplicate(att->att_value);
1036         return (natt);
1037 }

1039 attrib_t *
1040 attrib_merge_add(attrib_t *eatt, attrib_t *natt)
1041 {
1042         int i;
1043         attrib_t *att;
1044         attrib_val_t *atv, *eatv, *natv;
1045         lst_t *values;

1047         eatv = eatt->att_value;
1048         natv = natt->att_value;
1049         att = ATT_ALLOC();
1050         att->att_name = util_safe_strdup(eatt->att_name);
```

```
1052         if (eatv->att_val_type == ATT_VAL_TYPE_NULL) {
1053
1054                 /* NULL + X -> X */
1055                 atv = attrib_val_duplicate(natv);
1056
1057         } else if (natv->att_val_type == ATT_VAL_TYPE_NULL) {
1058
1059                 /* X + NULL -> X */
1060                 atv = attrib_val_duplicate(eatv);
1061
1062         } else if (eatv->att_val_type == ATT_VAL_TYPE_VALUE &&
1063             natv->att_val_type == ATT_VAL_TYPE_VALUE) {
1064
1065                 /* VALUE + VALUE -> LIST */
1066                 values = util_safe_malloc(sizeof (lst_t));
1067                 lst_create(values);
1068                 lst_insert_tail(values, attrib_val_duplicate(eatv));
1069                 lst_insert_tail(values, attrib_val_duplicate(natv));
1070                 atv = ATT_VAL_ALLOC_LIST(values);
1071
1072         } else if (eatv->att_val_type == ATT_VAL_TYPE_VALUE &&
1073             natv->att_val_type == ATT_VAL_TYPE_LIST) {
1074
1075                 /* VALUE + LIST -> LIST */
1076                 atv = attrib_val_duplicate(natv);
1077                 lst_insert_head(atv->att_val_values,
1078                     attrib_val_duplicate(eatv));
1079
1080         } else if (eatv->att_val_type == ATT_VAL_TYPE_LIST &&
1081             natv->att_val_type == ATT_VAL_TYPE_VALUE) {
1082
1083                 /* LIST + VALUE -> LIST */
1084                 atv = attrib_val_duplicate(eatv);
1085                 lst_insert_tail(atv->att_val_values,
1086                     attrib_val_duplicate(natv));
1087
1088         } else if (eatv->att_val_type == ATT_VAL_TYPE_LIST &&
1089             natv->att_val_type == ATT_VAL_TYPE_LIST) {
1090
1091                 /* LIST + LIST -> LIST */
1092                 atv = attrib_val_duplicate(eatv);
1093                 for (i = 0; i < lst_size(natv->att_val_values); i++) {
1094                         lst_insert_tail(atv->att_val_values,
1095                             attrib_val_duplicate(
1096                                 lst_at(natv->att_val_values, i)));
1097                 }
1098         }
1099
1100         att->att_value = atv;
1101         return (att);
1102 }
1103
1104 int
1105 attrib_val_equal(attrib_val_t *xatv, attrib_val_t *yatv)
1106 {
1107         int i;
1108         attrib_val_t *xv, *yv;
1109
1110         if (xatv->att_val_type != yatv->att_val_type)
1111                 return (1);
1112
1113         switch (xatv->att_val_type) {
1114                 case ATT_VAL_TYPE_NULL:
1115                         return (0);
1116                 case ATT_VAL_TYPE_VALUE:
1117                         if (SEQU(xatv->att_val_value, yatv->att_val_value)) {
```



```
1118                                 return (0);
1119                         }
1120                         return (1);
1121                 case ATT_VAL_TYPE_LIST:
1122                         if (lst_size(xatv->att_val_values) !=
1123                             lst_size(yatv->att_val_values))
1124                                 return (1);
1125                         for (i = 0; i < lst_size(xatv->att_val_values);
1126                             i++) {
1127                                 xv = lst_at(xatv->att_val_values, i);
1128                                 yv = lst_at(yatv->att_val_values, i);
1129                                 if (attrib_val_equal(xv, yv) != 0) {
1130                                         return (1);
1131                                 }
1132                         }
1133                 break;
1134         }
1135         return (0);
1136 }
1137
1138 attrib_t *
1139 attrib_merge_remove(attrib_t *eatt, attrib_t *natt, lst_t *errlst)
1140 {
1141
1142         int i, j;
1143         attrib_t *att = NULL;
1144         attrib_val_t *eatv, *natv;
1145         attrib_val_t *ev, *nv1, *nv2;
1146         lst_t *values;
1147         boolean_t found;
1148
1149         eatv = eatt->att_value;
1150         natv = natt->att_value;
1151
1152         if (eatv->att_val_type == ATT_VAL_TYPE_NULL &&
1153             natv->att_val_type == ATT_VAL_TYPE_NULL) {
1154
1155                 /* NULL - NULL -> EMPTY */
1156                 att = attrib_duplicate(eatt);
1157                 (void) strcpy(att->att_name, "");
1158
1159         } else if (eatv->att_val_type == ATT_VAL_TYPE_NULL ||
1160             (eatv->att_val_type == ATT_VAL_TYPE_VALUE &&
1161             natv->att_val_type == ATT_VAL_TYPE_LIST) ||
1162             (eatv->att_val_type == ATT_VAL_TYPE_LIST &&
1163             natv->att_val_type == ATT_VAL_TYPE_VALUE)) {
1164
1165                 /* NULL - X -> ERR, VALUE - LIST -> ERR, LIST - VALUE -> ERR */
1166                 util_add_errmsg(errlst, gettext(
1167                     "Can not remove attribute \"%s\""),
1168                     eatt->att_name);
1169
1170         } else if (natv->att_val_type == ATT_VAL_TYPE_NULL) {
1171
1172                 /* X - NULL -> X */
1173                 att = attrib_duplicate(eatt);
1174
1175         } else if (eatv->att_val_type == ATT_VAL_TYPE_VALUE &&
1176             natv->att_val_type == ATT_VAL_TYPE_VALUE) {
1177
1178                 /* VALUE - VALUE -> {EMPTY | ERR} */
1179                 if (attrib_val_equal(eatv, natv) == 0) {
1180                         att = ATT_ALLOC();
1181                         att->att_name = util_safe_strdup("");
1182                         att->att_value = ATT_VAL_ALLOC_NULL();
1183                 } else {
```

```
1184                        util_add_errmsg(errlst, gettext(
1185                            "Can not remove attribute \"%s\""),
1186                            eatt->att_name);
1187                }

1189        } else if (eatv->att_val_type == ATT_VAL_TYPE_LIST &&
1190            natv->att_val_type == ATT_VAL_TYPE_LIST) {
1191                /* LIST - LIST -> {EMPTY | ERR | LIST} */
1192                if (attrib_val_equal(eatv, natv) == 0) {
1193                        att = ATT_ALLOC();
1194                        att->att_name = util_safe_strdup("");
1195                        att->att_value = ATT_VAL_ALLOC_NULL();
1196                        goto out;
1197                }

1199                for (i = 0; i < lst_size(natv->att_val_values); i++) {
1200                        nv1 = lst_at(natv->att_val_values, i);
1201                        for (j = 0; j < lst_size(natv->att_val_values);
1202                            j++) {
1203                                nv2 = lst_at(natv->att_val_values, j);
1204                                if (i != j && attrib_val_equal(nv1, nv2) == 0) {
1205                                        util_add_errmsg(errlst, gettext(
1206                                            "Duplicate values, can not remove"
1207                                            " attribute \"%s\""),
1208                                            eatt->att_name);
1209                                        goto out;
1210                                }
1211                        }

1213                        found = B_FALSE;
1214                        for (j = 0; j < lst_size(eatv->att_val_values);
1215                            j++) {
1216                                ev = lst_at(eatv->att_val_values, j);
1217                                if (attrib_val_equal(nv1, ev) == 0) {
1218                                        found = B_TRUE;
1219                                        break;
1220                                }
1221                        }

1223                        if (!found) {
1224                                util_add_errmsg(errlst, gettext(
1225                                    "Value not found, can not remove"
1226                                    " attribute \"%s\""),
1227                                    eatt->att_name);
1228                                goto out;
1229                        }
1230                }

1232                values = util_safe_malloc(sizeof (lst_t));
1233                lst_create(values);
1234                for (i = 0; i < lst_size(eatv->att_val_values); i++) {
1235                        ev = lst_at(eatv->att_val_values, i);
1236                        found = B_FALSE;
1237                        for (j = 0; j < lst_size(natv->att_val_values);
1238                            j++) {
1239                                nv1 = lst_at(natv->att_val_values, j);
1240                                if (attrib_val_equal(ev, nv1) == 0) {
1241                                        found = B_TRUE;
1242                                        break;
1243                                }
1244                        }

1246                        if (!found) {
1247                                lst_insert_tail(values,
1248                                    attrib_val_duplicate(ev));
1249                        }
```



```
1250                }
1251                att = ATT_ALLOC();
1252                att->att_name = util_safe_strdup(eatt->att_name);
1253                att->att_value = ATT_VAL_ALLOC_LIST(values);
1254        }

1256 out:
1257        return (att);
1258 }

1260 attrib_t *
1261 attrib_merge(attrib_t *eatt, attrib_t *natt, int flags, lst_t *errlst)
1262 {
1263        attrib_t *att = NULL;

1265        VERIFY(SEQU(eatt->att_name, natt->att_name));

1267        if (flags & F_MOD_ADD) {
1268                att = attrib_merge_add(eatt, natt);
1269        } else if (flags & F_MOD_REM) {
1270                att = attrib_merge_remove(eatt, natt, errlst);
1271        }

1273        return (att);
1274 }

1276 void
1277 attrib_merge_attrib_lst(lst_t **eattrs, lst_t *nattrs, int flags,
1278     lst_t *errlst)
1279 {

1281        lst_t *attrs = NULL;
1282        int i, j;
1283        attrib_t *att, *natt, *eatt;
1284        boolean_t found;

1286        if (flags & F_MOD_ADD) {
1287                attrs = util_safe_malloc(sizeof (lst_t));
1288                lst_create(attrs);

1290                for (i = 0; i < lst_size(*eattrs); i++) {
1291                        eatt = lst_at(*eattrs, i);
1292                        found = B_FALSE;
1293                        for (j = 0; j < lst_size(nattrs); j++) {
1294                                natt = lst_at(nattrs, j);
1295                                if (SEQU(eatt->att_name, natt->att_name)) {
1296                                        found = B_TRUE;
1297                                        break;
1298                                }
1299                        }

1301                        att = found ? attrib_merge(eatt, natt, flags, errlst) :
1302                            attrib_duplicate(eatt);
1303                        if (att == NULL) {
1304                                attrib_free_lst(attrs);
1305                                UTIL_FREE_SNULL(attrs);
1306                                goto out;
1307                        }
1308                        lst_insert_tail(attrs, att);
1309                }

1311                for (i = 0; i < lst_size(nattrs); i++) {
1312                        natt = lst_at(nattrs, i);
1313                        found = B_FALSE;
1314                        for (j = 0; j < lst_size(*eattrs); j++) {
1315                                eatt = lst_at(*eattrs, j);
```

```
1316                                if (SEQU(natt->att_name, eatt->att_name)) {
1317                                        found = B_TRUE;
1318                                        break;
1319                                }
1320                        }
1321                        if (found)
1322                                continue;
1323                        lst_insert_tail(attrs, attrib_duplicate(natt));
1324                }

1326        } else if (flags & (F_MOD_REM | F_MOD_SUB)) {

1328                for (i = 0; i < lst_size(nattrs); i++) {
1329                        natt = lst_at(nattrs, i);
1330                        for (j = 0; j < lst_size(nattrs); j++) {
1331                                att = lst_at(nattrs, j);
1332                                if (SEQU(natt->att_name, att->att_name) &&
1333                                    i != j) {
1334                                        util_add_errmsg(errlst, gettext(
1335                                            "Duplicate Attributes \"%s\""),
1336                                            natt->att_name);
1337                                        goto out;
1338                                }
1339                        }

1341                        found = B_FALSE;
1342                        for (j = 0; j < lst_size(*eattrs); j++) {
1343                                eatt = lst_at(*eattrs, j);
1344                                if (SEQU(eatt->att_name, natt->att_name)) {
1345                                        found = B_TRUE;
1346                                        break;
1347                                }
1348                        }
1349                        if (!found) {
1350                                util_add_errmsg(errlst, gettext(
1351                                    "Project does not contain \"%s\""),
1352                                    natt->att_name);
1353                                goto out;
1354                        }
1355                }

1357                attrs = util_safe_malloc(sizeof (lst_t));
1358                lst_create(attrs);

1360                for (i = 0; i < lst_size(*eattrs); i++) {
1361                        eatt = lst_at(*eattrs, i);
1362                        found = B_FALSE;
1363                        for (j = 0; j < lst_size(nattrs); j++) {
1364                                natt = lst_at(nattrs, j);
1365                                if (SEQU(eatt->att_name, natt->att_name)) {
1366                                        found = B_TRUE;
1367                                        break;
1368                                }
1369                        }

1371                        if (flags & F_MOD_REM) {
1372                                att = found ?
1373                                    attrib_merge(eatt, natt, flags, errlst) :
1374                                    attrib_duplicate(eatt);
1375                        } else if (flags & F_MOD_SUB) {
1376                                att = attrib_duplicate(found ? natt : eatt);
1377                        }

1379                        if (att == NULL) {
1380                                attrib_free_lst(attrs);
1381                                UTIL_FREE_SNULL(attrs);
```



```
1382                                goto out;
1383                        } else if (SEQU(att->att_name, "")) {
1384                                attrib_free(att);
1385                        } else {
1386                                lst_insert_tail(attrs, att);
1387                        }
1388                }

1390        } else if (flags & F_MOD_REP) {
1391                attrs = util_safe_malloc(sizeof (lst_t));
1392                lst_create(attrs);
1393                for (i = 0; i < lst_size(nattrs); i++) {
1394                        natt = lst_at(nattrs, i);
1395                        lst_insert_tail(attrs, attrib_duplicate(natt));
1396                }
1397        }
1398 out:
1399        if (attrs != NULL) {
1400                attrib_free_lst(*eattrs);
1401                free(*eattrs);
1402                *eattrs = attrs;
1403        }
1404 }
```

```
**********************************************************
     622 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/attrib.h
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#ifndef _PROJENT_ATTRIB_H
#define _PROJENT_ATTRIB_H


#include <sys/types.h>
#include <regex.h>
#include <project.h>
#include <sys/varargs.h>

#include "projent.h"
#include "lst.h"

#ifdef  __cplusplus
extern "C" {
#endif

extern char *attrib_lst_tostring(lst_t *);
extern lst_t *attrib_parse_attributes(char *, int, lst_t *);
extern void attrib_free_lst(lst_t *);
extern char *attrib_tostring(void *);
extern void attrib_sort_lst(lst_t *);
extern int attrib_validate_lst(lst_t *, lst_t *);
extern void attrib_merge_attrib_lst(lst_t **, lst_t *, int, lst_t *);

#ifdef  __cplusplus
}
#endif
#endif /* _PROJENT_ATTRIB_H */
```

```
**********************************************************
    1728 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/lst.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <libintl.h>
#include "lst.h"
#include "util.h"


void
lst_create(lst_t *plst)
{
        plst->csz = 0;
        plst->tsz = 0;
        plst->buf = NULL;
}

int
lst_is_empty(lst_t *plst)
{
        return (plst->csz == 0);
}

void
lst_insert_head(lst_t *plst, void *ndata)
{
        int i;
        if (plst->csz == plst->tsz) {
                plst->tsz = (plst->tsz == 0) ? 1 : plst->tsz * 2;
                plst->buf = util_safe_realloc(plst->buf,
                    plst->tsz * sizeof (void *));
        }

        for (i = plst->csz; i > 0; i--)
                plst->buf[i] = plst->buf[i - 1];

        plst->buf[0] = ndata;
        plst->csz++;
}


void
lst_insert_tail(lst_t *plst, void *ndata)
{
        if (plst->csz == plst->tsz) {
                plst->tsz = (plst->tsz == 0) ? 1 : plst->tsz * 2;
                plst->buf = util_safe_realloc(plst->buf,
                    plst->tsz * sizeof (void *));
        }

        plst->buf[plst->csz++] = ndata;
}

int
lst_remove(lst_t *plst, void *rdata)
{
        int i, idx = -1;
        for (i = 0; i < plst->csz; i++) {
                if (plst->buf[i] == rdata) {
                        idx = i;
                        break;
                }
        }
```



```
        if (idx >= 0) {
                for (i = idx; i < plst->csz - 1; i++)
                        plst->buf[i] = plst->buf[i + 1];

                if (--plst->csz == 0) {
                        plst->tsz = 0;
                        free(plst->buf);
                        plst->buf = NULL;
                }
                return (0);
        }
        return (-1);
}

void *
lst_at(lst_t *plst, int idx)
{
        if (idx < 0 || idx >= plst->csz) {
                (void) fprintf(stderr, gettext(
                    "error accessing element outside lst\n"));
                exit(1);
        }
        return (plst->buf[idx]);
}

void *
lst_replace_at(lst_t *plst, int idx, void *ndata)
{
        void *odata;

        if (idx < 0 || idx >= plst->csz) {
                (void) fprintf(stderr, gettext(
                    "error accessing element outside lst\n"));
                exit(1);
        }
        odata = plst->buf[idx];
        plst->buf[idx] = ndata;
        return (odata);
}

int
lst_size(lst_t *plst)
{
        return (plst->csz);
}
```

```
**********************************************************
     509 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/lst.h
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#ifndef _PROJENT_LST_H
#define _PROJENT_LST_H


/*
 * #include <stdlib.h>
 */

#ifdef  __cplusplus
extern "C" {
#endif



typedef struct lst_s {
        int csz;
        int tsz;
        void **buf;
} lst_t;


void lst_create(lst_t *);
int lst_is_empty(lst_t *);
void lst_insert_head(lst_t *, void *);
void lst_insert_tail(lst_t *, void *);

int lst_remove(lst_t *, void *);
void *lst_at(lst_t *, int);
void *lst_replace_at(lst_t *, int, void *);
int lst_size(lst_t *);



#ifdef  __cplusplus
}
#endif
#endif /* _PROJENT_LST_H */
```

```
**********************************************************
   18404 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/projent.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <locale.h>
#include <stddef.h>
#include <limits.h>
#include <pwd.h>
#include <grp.h>
#include <unistd.h>
#include <rctl.h>
#include <regex.h>
#include <ctype.h>

#include "projent.h"
#include "attrib.h"
#include "util.h"


#define BOSTR_REG_EXP   "^"
#define EOSTR_REG_EXP   "$"
#define IDENT_REG_EXP   "[[:alpha:]][[:alnum:]_.-]*"
#define PRJID_REG_EXP   "[[:digit:]]+"
#define USERN_REG_EXP   "!?[[:alpha:]][[:alnum:]_.-]*"
#define GRUPN_REG_EXP   "!?[[:alnum:]][[:alnum:]]*"

#define TO_EXP(X)       BOSTR_REG_EXP X EOSTR_REG_EXP

#define PROJN_EXP       TO_EXP(IDENT_REG_EXP)
#define PRJID_EXP       TO_EXP(PRJID_REG_EXP)
#define USERN_EXP       TO_EXP(USERN_REG_EXP)
#define GRUPN_EXP       TO_EXP(GRUPN_REG_EXP)

/*ARGSUSED*/
int
projent_validate_name(char *pname, lst_t *errlst)
{
        /* Do nothing, as any parse-able project name is valid */
        return (0);
}

/*ARGSUSED*/
int
projent_validate_comment(char *comment, lst_t *errlst)
{
        /* Do nothing, as any parse-able project name is valid */
        return (0);
}

int
projent_validate_users(char *users, lst_t *errlst)
{
        char *susrs, *usrs, *usr;
        char *u, **ulast, **ulist;
        int ret = 0;
        int i;

        susrs = usrs = util_safe_strdup(users);
```



```
        ulast = ulist = util_safe_zmalloc(
            (strlen(users) + 1) * sizeof (char *));
        while ((usr = strsep(&usrs, ",")) != NULL) {
                if (*usr == '!')
                        usr++;
                if ((*usr == '\0') || (strcmp(usr, "*") == 0))
                        continue;

                if (getpwnam(usr) == NULL) {
                        util_add_errmsg(errlst, gettext(
                            "User \"%s\" does not exist"), usr);
                        ret = 1;
                }
                for (i = 0; (u = ulist[i]) != NULL; i++) {
                        if (strcmp(u, usr) == 0) {
                                util_add_errmsg(errlst, gettext(
                                    "Duplicate user names \"%s\""), usr);
                                ret = 1;
                        }
                }
                /* Add the user to the temporary list if not found */
                if (u == NULL) {
                        *ulast++ = usr;
                }
        }
        free(ulist);
        free(susrs);
        return (ret);
}

int
projent_validate_groups(char *groups, lst_t *errlst)
{
        char *sgrps, *grps, *grp;
        char *g, **glast, **glist;
        int ret = 0;
        int i;

        sgrps = grps = util_safe_strdup(groups);
        glast = glist = util_safe_zmalloc(
            (strlen(groups) + 1) * sizeof (char *));
        while ((grp = strsep(&grps, ",")) != NULL) {
                if (*grp == '!')
                        grp++;
                if ((*grp == '\0') || (strcmp(grp, "*") == 0))
                        continue;

                if (getgrnam(grp) == NULL) {
                        util_add_errmsg(errlst, gettext(
                            "Group \"%s\" does not exist"), grp);
                        ret = 1;
                }
                for (i = 0; (g = glist[i]) != NULL; i++) {
                        if (strcmp(g, grp) == 0) {
                                util_add_errmsg(errlst, gettext(
                                    "Duplicate group names \"%s\""), grp);
                                ret = 1;
                        }
                }
                /* Add the group to the temporary list if not found */
                if (g == NULL) {
                        *glast++ = grp;
                }
        }
        free(glist);
        free(sgrps);
```

```
128         return (ret);
129 }

131 int
132 projent_validate_attributes(lst_t *attrs, lst_t *errlst)
133 {
134         return (attrib_validate_lst(attrs, errlst));
135 }

137 char *
138 projent_tostring(projent_t *ent)
139 {
140         char *attrs;
141         char *ret = NULL;
142         if ((attrs = attrib_lst_tostring(ent->attrs)) != NULL) {
143                 (void) asprintf(&ret, "%s:%ld:%s:%s:%s:%s",
144                     ent->projname,
145                     ent->projid,
146                     ent->comment,
147                     ent->userlist,
148                     ent->grouplist,
149                     attrs);
150                 free(attrs);
151         }
152         return (ret);
153 }

155 int
156 projent_validate(projent_t *pent, int flags, lst_t *errlst)
157 {
158         char *str;

160         (void) projent_validate_name(pent->projname, errlst);
161         (void) projent_validate_projid(pent->projid, flags, errlst);
162         (void) projent_validate_comment(pent->comment, errlst);
163         (void) projent_validate_users(pent->userlist, errlst);
164         (void) projent_validate_groups(pent->grouplist, errlst);
165         (void) projent_validate_attributes(pent->attrs, errlst);

167         if ((str = projent_tostring(pent)) == NULL) {
168                 util_add_errmsg(errlst, gettext("error allocating memory"));
169         } else {
170                 if (strlen(str) > (PROJECT_BUFSZ - 2)) {
171                         util_add_errmsg(errlst, gettext(
172                             "projent line too long"));
173                 }
174                 free(str);
175         }
176         return (lst_is_empty(errlst) == 0);
177 }

179 int
180 projent_validate_lst(lst_t *plst, int flags, lst_t *errlst)
181 {
182         int e, i, idx;
183         projent_t *ent;
184         char *pnames = NULL;
185         projid_t *pids = NULL;
186         int ret = 0;

188         idx = 0;
189         for (e = 0; e < lst_size(plst); e++) {
190                 ent = lst_at(plst, e);
191                 /* Check for duplicate projname */
192                 if (pnames != NULL && strstr(pnames, ent->projname) != NULL) {
193                         util_add_errmsg(errlst, gettext(
```



```
194                             "Duplicate project name %s"), ent->projname);
195                         ret++;
196                 }

198                 /* Check for duplicate projid if DUP is not allowed */
199                 if (!(flags & F_PAR_DUP) && pids != NULL) {
200                         for (i = 0; i < idx; i++) {
201                                 if (ent->projid == pids[i]) {
202                                         util_add_errmsg(errlst, gettext(
203                                             "Duplicate proid %d"), ent->projid);
204                                         ret++;
205                                         break;
206                                 }
207                         }
208                 }

210                 /* Add the projname an projid to out temp list */
211                 pnames = UTIL_STR_APPEND2(pnames, "|", ent->projname);
212                 pids = util_safe_realloc(pids, (idx + 1) * sizeof (projid_t));
213                 pids[idx] = ent->projid;
214                 idx++;

216                 /* Validate the projet */
217                 ret += projent_validate(ent, flags, errlst);
218         }

220         free(pnames);
221         free(pids);

223         return (ret);
224 }

226 void
227 projent_free_attributes(lst_t *attribs)
228 {
229         attrib_free_lst(attribs);
230 }

232 void
233 projent_sort_attributes(lst_t *attribs)
234 {
235         attrib_sort_lst(attribs);
236 }

238 char *
239 projent_attrib_tostring(void *attrib)
240 {
241         return (attrib_tostring(attrib));
242 }

244 char *
245 projent_attrib_lst_tostring(lst_t *lst)
246 {
247         return (attrib_lst_tostring(lst));
248 }

250 void
251 projent_merge_attributes(lst_t **eattrs, lst_t *nattrs, int flags,
252     lst_t *errlst)
253 {
254         attrib_merge_attrib_lst(eattrs, nattrs, flags, errlst);
255 }

257 lst_t *
258 projent_parse_attributes(char *attribs, int flags, lst_t *errlst)
259 {
```

```
260         return (attrib_parse_attributes(attribs, flags, errlst));
261 }

263 void
264 projent_merge_usrgrp(char *usrgrp, char **elist, char *nlist,
265     int flags, lst_t *errlst)
266 {
267         char *res = NULL;
268         char *seusrs, *eusrs, *eusr;
269         char *snusrs, *nusrs, *nusr;
270         char *sn1usrs, *n1usrs, *n1usr;
271         char *sep;
272         int i, j;

274         sep = (flags & F_PAR_SPC) ? " ," : ",";

276         if (flags & F_MOD_ADD) {
277                 res = util_safe_strdup(*elist);

279                 snusrs = nusrs = util_safe_strdup(nlist);
280                 while ((nusr = strsep(&nusrs, sep)) != NULL) {
281                         if (*nusr == '\0')
282                                 continue;
283                         seusrs = eusrs = util_safe_strdup(*elist);
284                         while ((eusr = strsep(&eusrs, sep)) != NULL) {
285                                 if (*eusr == '\0')
286                                         continue;
287                                 if (strcmp(eusr, nusr) == 0) {
288                                         util_add_errmsg(errlst, gettext(
289                                             "Project already contains"
290                                             " %s \"%s\""), usrgrp, nusr);
291                                         UTIL_FREE_SNULL(res);
292                                         free(seusrs);
293                                         free(snusrs);
294                                         goto out;
295                                 }
296                         }
297                         free(seusrs);
298                         /* Append nusr to the result */
299                         if (*res != '\0')
300                                 res = UTIL_STR_APPEND1(res, ",");
301                         res = UTIL_STR_APPEND1(res, nusr);
302                 }
303                 free(snusrs);
304         } else if (flags & F_MOD_REM) {

306                 snusrs = nusrs = util_safe_strdup(nlist);
307                 for (i = 0; (nusr = strsep(&nusrs, sep)) != NULL; i++) {
308                         if (*nusr == '\0')
309                                 continue;
310                         sn1usrs = n1usrs = util_safe_strdup(nlist);
311                         for (j = 0; (n1usr = strsep(&n1usrs, sep)) != NULL;
312                             j++) {
313                                 if (i != j && strcmp(nusr, n1usr) == 0) {
314                                         util_add_errmsg(errlst, gettext(
315                                             "Duplicate %s name \"%s\""),
316                                             usrgrp, nusr);
317                                         free(sn1usrs);
318                                         free(snusrs);
319                                         goto out;
320                                 }
321                         }
322                         free(sn1usrs);

324                         seusrs = eusrs = util_safe_strdup(*elist);
325                         while ((eusr = strsep(&eusrs, sep)) != NULL) {
```



```
326                                 if (strcmp(nusr, eusr) == 0) {
327                                         break;
328                                 }
329                         }
330                         free(seusrs);

332                         if (eusr == NULL) {
333                                 util_add_errmsg(errlst, gettext(
334                                     "Project does not contain %s name \"%s\""),
335                                     usrgrp, nusr);
336                                 free(snusrs);
337                                 goto out;
338                         }
339                 }
340                 free(snusrs);


343                 res = util_safe_zmalloc(1);
344                 seusrs = eusrs = util_safe_strdup(*elist);
345                 while ((eusr = strsep(&eusrs, sep)) != NULL) {
346                         if (*eusr == '\0')
347                                 continue;
348                         snusrs = nusrs = util_safe_strdup(nlist);
349                         while ((nusr = strsep(&nusrs, sep)) != NULL) {
350                                 if (strcmp(eusr, nusr) == 0) {
351                                         break;
352                                 }
353                         }
354                         free(snusrs);

356                         if (nusr == NULL) {
357                                 if (*res != '\0')
358                                         res = UTIL_STR_APPEND1(res, ",");
359                                 res = UTIL_STR_APPEND1(res, eusr);
360                         }
361                 }
362                 free(seusrs);
363         } else if (flags & F_MOD_SUB || flags & F_MOD_REP) {
364                 res = util_safe_strdup(nlist);
365         }

367 out:
368         if (res != NULL) {
369                 free(*elist);
370                 *elist = res;
371         }
372 }

374 char *
375 projent_parse_users(char *nlist, int flags, lst_t *errlst)
376 {
377         char *ulist = NULL;
378         char *susrs, *usrs, *usr;
379         regex_t usernexp;
380         char *sep;

382         if (regcomp(&usernexp, USERN_EXP, REG_EXTENDED) != 0) {
383                 util_add_errmsg(errlst, gettext(
384                     "Failed to compile regular expression: \"%s\""),
385                     USERN_EXP);
386                 goto out;
387         }

389         sep = (flags & F_PAR_SPC) ? " ," : ",";
390         susrs = usrs = util_safe_strdup(nlist);
391         ulist = util_safe_zmalloc(1);
```

```
393         while ((usr = strsep(&usrs, sep)) != NULL) {
394                 if (*usr == '\0')
395                         continue;
396
397                 if (regexec(&usernexp, usr, 0, NULL, 0) != 0 &&
398                     strcmp(usr, "*") != 0 &&
399                     strcmp(usr, "!*") != 0) {
400                         util_add_errmsg(errlst, gettext(
401                             "Invalid user name \"%s\""), usr);
402                         UTIL_FREE_SNULL(ulist);
403                         break;
404                 }
405                 /* Append ',' first if required */
406                 if (*ulist != '\0')
407                         ulist = UTIL_STR_APPEND1(ulist, ",");
408                 ulist = UTIL_STR_APPEND1(ulist, usr);
409         }
410
411         free(susrs);
412         regfree(&usernexp);
413 out:
414         return (ulist);
415 }
416
417
418 char *
419 projent_parse_groups(char *nlist, int flags, lst_t *errlst)
420 {
421         char *glist = NULL;
422         char *sgrps, *grps, *grp;
423         regex_t groupnexp;
424         char *sep;
425
426         if (regcomp(&groupnexp, GRUPN_EXP, REG_EXTENDED) != 0) {
427                 util_add_errmsg(errlst, gettext(
428                     "Failed to compile regular expression: \"%s\""),
429                     GRUPN_EXP);
430                 goto out;
431         }
432
433         sep = (flags & F_PAR_SPC) ? " ," : ",";
434         sgrps = grps = util_safe_strdup(nlist);
435         glist = util_safe_zmalloc(1);
436
437         while ((grp = strsep(&grps, sep)) != NULL) {
438                 if (*grp == '\0')
439                         continue;
440
441                 if (regexec(&groupnexp, grp, 0, NULL, 0) != 0 &&
442                     strcmp(grp, "*") != 0 &&
443                     strcmp(grp, "!*") != 0) {
444                         util_add_errmsg(errlst, gettext(
445                             "Invalid group name \"%s\""), grp);
446                         UTIL_FREE_SNULL(glist);
447                         break;
448                 }
449                 /* Append ',' first if required */
450                 if (*glist != '\0')
451                         glist = UTIL_STR_APPEND1(glist, ",");
452                 glist = UTIL_STR_APPEND1(glist, grp);
453         }
454
455         free(sgrps);
456         regfree(&groupnexp);
457 out:
```



```
458         return (glist);
459 }
460
461 int
462 projent_parse_comment(char *comment, lst_t *errlst)
463 {
464         int ret = 0;
465         if (strchr(comment, ':') != NULL) {
466                 util_add_errmsg(errlst, gettext(
467                     "Invalid Comment \"%s\": should not contain ':'"),
468                     comment);
469                 ret = 1;
470         }
471         return (ret);
472 }
473
474 int
475 projent_validate_unique_id(lst_t *plst, projid_t projid, lst_t *errlst)
476 {
477         int e;
478         projent_t *ent;
479         for (e = 0; e < lst_size(plst); e++) {
480                 ent = lst_at(plst, e);
481                 if (ent->projid == projid) {
482                         util_add_errmsg(errlst, gettext(
483                             "Duplicate projid \"%d\""), projid);
484                         return (1);
485                 }
486         }
487         return (0);
488 }
489 int
490 projent_validate_projid(projid_t projid, int flags, lst_t *errlst)
491 {
492         projid_t maxprojid;
493
494         maxprojid = (flags & F_PAR_RES) ? 0 : 100;
495
496         if (projid < maxprojid) {
497                 util_add_errmsg(errlst, gettext(
498                     "Invalid projid \"%d\": "
499                     "must be >= 100"),
500                     projid);
501                 return (1);
502         }
503         return (0);
504 }
505
506 int
507 projent_parse_projid(char *projidstr, projid_t *pprojid, lst_t *errlst)
508 {
509         char *ptr;
510         long long llid;
511         regex_t prjidexp;
512         int ret = 0;
513
514         if (regcomp(&prjidexp, PRJID_EXP, REG_EXTENDED) != 0) {
515                 util_add_errmsg(errlst, gettext(
516                     "Failed to compile regular expression: \"%s\""), PRJID_EXP);
517                 return (1);
518         }
519
520
521         if (regexec(&prjidexp, projidstr, 0, NULL, 0) != 0) {
522                 util_add_errmsg(errlst, gettext("Invalid project id: \"%s\""),
523                     projidstr);
```

```
524                 ret = 1;
525                 goto out;
526         }

528         llid = strtoll(projidstr, &ptr, 10);

530         /* projid should be a positive number */
531         if (llid == 0 && errno == ERANGE && *ptr != '\0') {
532                 util_add_errmsg(errlst, gettext("Invalid project id: \"%s\""),
533                     projidstr);
534                 ret = 1;
535                 goto out;
536         }

538         /* projid should be a positive number >= 0 */
539         if (llid < 0) {
540                 util_add_errmsg(errlst, gettext(
541                     "Invalid projid \"%lld\": must be >= 0"), llid);
542                 ret = 1;
543                 goto out;
544         }

546         /* projid should be less than UID_MAX */
547         if (llid > INT_MAX) {
548                 util_add_errmsg(errlst, gettext(
549                     "Invalid projid \"%lld\": must be <= %d"),
550                     llid, INT_MAX);
551                 ret = 1;
552                 goto out;
553         }

555         if (pprojid != NULL)
556                 *pprojid = llid;
557 out:
558         regfree(&prjidexp);
559         return (ret);
560 }

562 int
563 projent_validate_unique_name(lst_t *plst, char *pname, lst_t *errlst)
564 {
565         int e;
566         projent_t *ent;
567         for (e = 0; e < lst_size(plst); e++) {
568                 ent = lst_at(plst, e);
569                 if (strcmp(ent->projname, pname) == 0) {
570                         util_add_errmsg(errlst, gettext(
571                             "Duplicate project name \"%s\""), pname);
572                         return (1);
573                 }
574         }
575         return (0);
576 }

578 int
579 projent_parse_name(char *pname, lst_t *errlst)
580 {
581         int ret = 1;
582         regex_t projnexp;
583         if (regcomp(&projnexp, PROJN_EXP, REG_EXTENDED) != 0) {
584                 util_add_errmsg(errlst, gettext(
585                     "Failed to compile regular expression: \"%s\""),
586                     PROJN_EXP);
587                 goto out;
588         }
```



```
590         if (regexec(&projnexp, pname, 0, NULL, 0) != 0) {
591                 util_add_errmsg(errlst, gettext(
592                     "Invalid project name \"%s\", "
593                     "contains invalid characters"), pname);
594         } else if (strlen(pname) > PROJNAME_MAX) {
595                 util_add_errmsg(errlst, gettext(
596                     "Invalid project name \"%s\", "
597                     "name too long"), pname);
598         } else {
599                 ret = 0;
600         }

602         regfree(&projnexp);
603 out:
604         return (ret);
605 }

607 void
608 projent_free(projent_t *ent)
609 {
610         free(ent->projname);
611         free(ent->comment);
612         free(ent->userlist);
613         free(ent->grouplist);
614         attrib_free_lst(ent->attrs);
615         free(ent->attrs);
616 }

618 projent_t *
619 projent_parse_components(char *projname, char *idstr, char *comment,
620     char *users, char *groups, char *attr, int flags, lst_t *errlst)
621 {
622         projent_t *ent;
623         int reterr = 0;

625         ent = util_safe_zmalloc(sizeof (projent_t));


628         ent->projname = util_safe_strdup(projname);
629         ent->comment = util_safe_strdup(comment);

631         reterr += projent_parse_name(ent->projname, errlst);
632         reterr += projent_parse_projid(idstr, &ent->projid, errlst);
633         reterr += projent_parse_comment(ent->comment, errlst);
634         ent->userlist = projent_parse_users(users, flags, errlst);
635         ent->grouplist = projent_parse_groups(groups, flags, errlst);
636         ent->attrs = projent_parse_attributes(attr, flags, errlst);

638         if (reterr > 0 || ent->userlist == NULL ||
639             ent->grouplist == NULL || ent->attrs == NULL) {
640                 projent_free(ent);
641                 UTIL_FREE_SNULL(ent);
642         }

644         return (ent);
645 }
646 projent_t *
647 projent_parse(char *projstr, int flags, lst_t *errlst)
648 {
649         char *str, *sstr;
650         char *projname, *idstr, *comment, *users, *groups, *attrstr;
651         projent_t *ent;

653         if (projstr == NULL)
654                 return (NULL);
```

```
656         projname = idstr = comment = users = groups = attrstr = NULL;
657         ent = NULL;

659         sstr  = str = util_safe_strdup(projstr);

661         if ((projname = util_safe_strdup(strsep(&str, ":"))) == NULL ||
662             (idstr = util_safe_strdup(strsep(&str, ":"))) == NULL ||
663             (comment = util_safe_strdup(strsep(&str, ":"))) == NULL ||
664             (users = util_safe_strdup(strsep(&str, ":"))) == NULL ||
665             (groups = util_safe_strdup(strsep(&str, ":"))) == NULL ||
666             (attrstr = util_safe_strdup(strsep(&str, ":"))) == NULL ||
667             strsep(&str, ":") != NULL) {
668                 util_add_errmsg(errlst, gettext(
669                     "Incorrect number of fields.  Should have 5 \":\"'s."));
670                 goto out;
671         }

673         ent = projent_parse_components(projname, idstr, comment, users, groups,
674             attrstr, flags, errlst);
675 out:
676         free(sstr);
677         free(projname); free(idstr); free(comment);
678         free(users); free(groups); free(attrstr);
679         return (ent);
680 }


683 void
684 projent_free_lst(lst_t *plst)
685 {
686         projent_t *ent;

688         if (plst == NULL)
689                 return;

691         while (!lst_is_empty(plst)) {
692                 ent = lst_at(plst, 0);
693                 (void) lst_remove(plst, ent);
694                 projent_free(ent);
695                 free(ent);
696         }
697 }


700 void
701 projent_put_lst(char *projfile, lst_t *plst, lst_t *errlst)
702 {
703         char *tmpprojfile, *attrs;
704         FILE *fp;
705         projent_t *ent;
706         struct stat statbuf;
707         int e, ret;

709         tmpprojfile = NULL;
710         if (asprintf(&tmpprojfile, "%s.%ld_tmp", projfile, getpid()) == -1) {
711                 util_add_errmsg(errlst, gettext(
712                     "Failed to allocate memory"));
713                 goto out;
714         }

716         if (stat(projfile, &statbuf) != 0) {
717                 util_add_errmsg(errlst, gettext(
718                     "Failed to access %s: %s"),
719                     projfile, strerror(errno));
720                 goto out;
721         }
```



```
722         if ((fp = fopen(tmpprojfile, "wx")) == NULL) {
723                 util_add_errmsg(errlst, gettext(
724                     "Cannot create %s: %s"),
725                     tmpprojfile, strerror(errno));
726                 goto out;
727         }

729         for (e = 0; e < lst_size(plst); e++) {
730                 ent = lst_at(plst, e);
731                 attrs = attrib_lst_tostring(ent->attrs);
732                 ret = fprintf(fp, "%s:%ld:%s:%s:%s:%s\n", ent->projname,
733                     ent->projid, ent->comment, ent->userlist, ent->grouplist,
734                     attrs);
735                 free(attrs);
736                 if (ret < 0) {
737                         util_add_errmsg(errlst, gettext(
738                             "Failed to write to  %s: %s"),
739                             tmpprojfile, strerror(errno));
740                         /* Remove the temporary file and exit */
741                         (void) unlink(tmpprojfile);
742                         goto out1;
743                 }
744         }

746         if (chown(tmpprojfile, statbuf.st_uid, statbuf.st_gid) != 0) {
747                 util_add_errmsg(errlst, gettext(
748                     "Cannot set ownership of  %s: %s"),
749                     tmpprojfile, strerror(errno));
750                 (void) unlink(tmpprojfile);
751                 goto out1;
752         }

754         if (rename(tmpprojfile, projfile) != 0) {
755                 util_add_errmsg(errlst, gettext(
756                     "Cannot rename %s to %s : %s"),
757                     tmpprojfile, projfile, strerror(errno));
758                 (void) unlink(tmpprojfile);
759                 goto out1;
760         }

762 out1:
763         (void) fclose(fp);
764 out:
765         free(tmpprojfile);
766 }

768 lst_t *
769 projent_get_lst(char *projfile, int flags, lst_t *errlst)
770 {
771         FILE *fp;
772         lst_t *plst;
773         int line = 0;
774         char *buf = NULL, *nlp;
775         size_t cap = 0;
776         projent_t *ent;

778         plst = util_safe_malloc(sizeof (lst_t));
779         lst_create(plst);

781         if ((fp = fopen(projfile, "r")) == NULL) {
782                 if (errno == ENOENT) {
783                         /*
784                          * There is no project file,
785                          * return an empty lst
786                          */
787                         return (plst);
```

```
788                 } else {
789                         /* Report the error unable to open the file */
790                         util_add_errmsg(errlst, gettext(
791                             "Cannot open %s: %S"),
792                             projfile, strerror(errno));
793
794                         free(plst);
795                         return (NULL);
796                 }
797         }
798
799         while ((getline(&buf, &cap, fp)) != -1 && ++line) {
800
801                 if ((nlp = strchr(buf, '\n')) != NULL)
802                         *nlp = '\0';
803
804                 if ((ent = projent_parse(buf, flags, errlst)) != NULL) {
805                         lst_insert_tail(plst, ent);
806                 } else {
807                         /* Report the error */
808                         util_add_errmsg(errlst, gettext(
809                             "Error parsing: %s line: %d: \"%s\""),
810                             projfile, line, buf);
811
812                         /* free the allocated resources */
813                         projent_free_lst(plst);
814                         UTIL_FREE_SNULL(plst);
815                         goto out;
816                 }
817         }
818
819         if (flags & F_PAR_VLD && plst != NULL) {
820                 if (projent_validate_lst(plst, flags, errlst) != 0) {
821                         projent_free_lst(plst);
822                         UTIL_FREE_SNULL(plst);
823                 }
824         }
825
826 out:
827         (void) fclose(fp);
828         free(buf);
829         return (plst);
830 }
```

```
**********************************************************
    2070 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/projent.h
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#ifndef _PROJENT_PROJENT_H
#define _PROJENT_PROJENT_H

#include <sys/types.h>
#include <regex.h>
#include <project.h>
#include <sys/varargs.h>

#include "lst.h"

#define F_PAR_VLD       0x0001  /* Run validation after parsing */
#define F_PAR_SPC       0x0002  /* Allow spaces between names */
#define F_PAR_UNT       0x0004  /* Allow units in attribs values */
#define F_PAR_RES       0x0008  /* Allow projid < 100 */
#define F_PAR_DUP       0x0010  /* Allow duplicate projids */

#define F_MOD_ADD       0x0100
#define F_MOD_REM       0x0200
#define F_MOD_SUB       0x0400
#define F_MOD_REP       0x0800

#ifdef  __cplusplus
extern "C" {
#endif


typedef struct projent {
        char *projname;
        projid_t projid;
        char *comment;
        char *userlist;
        char *grouplist;
        lst_t *attrs;
} projent_t;


extern void projent_free(projent_t *);
extern projent_t *projent_parse(char *, int, lst_t *);
extern projent_t *projent_parse_components(char *, char *, char *, char *,
    char *, char *, int, lst_t *);
extern int projent_validate(projent_t *, int, lst_t *);
extern lst_t *projent_get_lst(char *, int, lst_t *);
extern void projent_free_lst(lst_t *);
extern int  projent_parse_name(char *, lst_t *);
extern int  projent_validate_unique_name(lst_t *, char *, lst_t *);
extern int  projent_parse_projid(char *, projid_t *, lst_t *);
extern int  projent_validate_projid(projid_t, int, lst_t *);
extern int  projent_validate_unique_id(lst_t *, projid_t, lst_t *);
extern int  projent_parse_comment(char *, lst_t *);
extern void projent_merge_usrgrp(char *, char **, char *, int, lst_t *);
extern char *projent_parse_users(char *, int, lst_t *);
extern char *projent_parse_groups(char *, int, lst_t *);
extern void projent_merge_attributes(lst_t **, lst_t *, int, lst_t *);
extern lst_t *projent_parse_attributes(char *, int, lst_t *);
extern void projent_sort_attributes(lst_t *);
extern void projent_free_attributes(lst_t *);
extern char *projent_attrib_tostring(void *);
extern char *projent_attrib_lst_tostring(lst_t *);
extern void projent_put_lst(char *, lst_t *, lst_t *);

#ifdef  __cplusplus
```



```
}
#endif

#endif /* _PROJENT_PROJENT_H */
```

```
**********************************************************
    3682 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/resctl.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <locale.h>
#include <stddef.h>
#include <limits.h>
#include <rctl.h>
#include <regex.h>
#include <ctype.h>
#include <zone.h>
#include <pool.h>
#include <sys/pool_impl.h>
#include <unistd.h>
#include <stropts.h>

#include "util.h"
#include "resctl.h"

sig_t sigs[SIGS_CNT] = {
        /* Signal names */
        {"ABRT", RESCTL_SIG_ABRT},
        {"XRES", RESCTL_SIG_XRES},
        {"HUP",  RESCTL_SIG_HUP},
        {"STOP", RESCTL_SIG_STOP},
        {"TERM", RESCTL_SIG_TERM},
        {"KILL", RESCTL_SIG_KILL},
        {"XFSZ", RESCTL_SIG_XFSZ},
        {"XCPU", RESCTL_SIG_XCPU},

        /* Singnal numbers */
        {"6",  RESCTL_SIG_ABRT},
        {"38", RESCTL_SIG_XRES},
        {"1",  RESCTL_SIG_HUP},
        {"23", RESCTL_SIG_STOP},
        {"15", RESCTL_SIG_TERM},
        {"9",  RESCTL_SIG_KILL},
        {"31", RESCTL_SIG_XFSZ},
        {"30", RESCTL_SIG_XCPU},
};

/*
 * Check the existance of a resource pool in the system
 */
int
resctl_pool_exist(char *name)
{
        pool_conf_t *conf;
        pool_status_t status;
        int fd;

        /*
         * Determine if pools are enabled using /dev/pool, as
         * libpool may not be present.
         */
        if (getzoneid() != GLOBAL_ZONEID ||
            (fd = open("/dev/pool", O_RDONLY)) < 0) {
                return (1);
        }

        if (ioctl(fd, POOL_STATUS, &status) < 0) {
                (void) close(fd);
                return (1);
        }

        (void) close(fd);

        if (status.ps_io_state != 1)
                return (1);

        /* If pools are enabled, assume libpool is present. */
        if ((conf = pool_conf_alloc()) == NULL)
                return (1);

        if (pool_conf_open(conf, pool_dynamic_location(), PO_RDONLY)) {
                pool_conf_free(conf);
                return (1);
        }

        if (pool_get_pool(conf, name) == NULL) {
                (void) pool_conf_close(conf);
                pool_conf_free(conf);
                return (1);
        }

        (void) pool_conf_close(conf);
        pool_conf_free(conf);
        return (0);
}

int
resctl_get_info(char *name, resctl_info_t *pinfo)
{
        rctlblk_t *blk1, *blk2, *tmp;
        rctl_priv_t priv;
        int ret = 1;

        blk1 = blk2 = tmp = NULL;
        blk1 = util_safe_malloc(rctlblk_size());
        blk2 = util_safe_malloc(rctlblk_size());

        if (getrctl(name, NULL, blk1, RCTL_FIRST) == 0) {
                priv = rctlblk_get_privilege(blk1);
                while (priv != RCPRIV_SYSTEM) {
                        tmp = blk2;
                        blk2 = blk1;
                        blk1 = tmp;
                        if (getrctl(name, blk2, blk1, RCTL_NEXT) != 0) {
                                goto out;
                        }
                        priv = rctlblk_get_privilege(blk1);
                }

                pinfo->value = rctlblk_get_value(blk1);
                pinfo->flags = rctlblk_get_global_flags(blk1);
                ret = 0;
        }

out:
        free(blk1);
        free(blk2);
        return (ret);
}
```

```
128 void
129 resctl_get_rule(resctl_info_t *pinfo, resctlrule_t *prule)
130 {

132         prule->resctl_max = pinfo->value;
133         if (pinfo->flags & RCTL_GLOBAL_BYTES) {
134                 prule->resctl_type = RESCTL_TYPE_BYTES;
135         } else if (pinfo->flags & RCTL_GLOBAL_SECONDS) {
136                 prule->resctl_type = RESCTL_TYPE_SCNDS;
137         } else if (pinfo->flags & RCTL_GLOBAL_COUNT) {
138                 prule->resctl_type = RESCTL_TYPE_COUNT;
139         } else {
140                 prule->resctl_type = RESCTL_TYPE_UNKWN;
141         }

143         if (pinfo->flags & RCTL_GLOBAL_NOBASIC) {
144                 prule->resctl_privs = RESCTL_PRIV_PRIVE | RESCTL_PRIV_PRIVD;
145         } else {
146                 prule->resctl_privs = RESCTL_PRIV_ALLPR;
147         }

149         if (pinfo->flags & RCTL_GLOBAL_DENY_ALWAYS) {
150                 prule->resctl_action = RESCTL_ACTN_DENY;
151         } else if (pinfo->flags & RCTL_GLOBAL_DENY_NEVER) {
152                 prule->resctl_action = RESCTL_ACTN_NONE;
153         } else {
154                 prule->resctl_action = RESCTL_ACTN_NONE | RESCTL_ACTN_DENY;
155         }

157         if (pinfo->flags & RCTL_GLOBAL_SIGNAL_NEVER) {
158                 prule->resctl_sigs = 0;
159         } else {
160                 prule->resctl_action |= RESCTL_ACTN_SIGN;
161                 prule->resctl_sigs = RESCTL_SIG_CMN;
162                 if (pinfo->flags & RCTL_GLOBAL_CPU_TIME) {
163                         prule->resctl_sigs |= RESCTL_SIG_XCPU;
164                 }
165                 if (pinfo->flags & RCTL_GLOBAL_FILE_SIZE) {
166                         prule->resctl_sigs |= RESCTL_SIG_XFSZ;
167                 }
168         }
169 }
```

```
**********************************************************
    1386 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/resctl.h
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#ifndef _PROJENT_RESCTL_H
#define _PROJENT_RESCTL_H


/*
 * Put includes here if needed to be.
 */

#ifdef  __cplusplus
extern "C" {
#endif


#define RESCTL_TYPE_UNKWN       0x00
#define RESCTL_TYPE_BYTES       0x01
#define RESCTL_TYPE_SCNDS       0x02
#define RESCTL_TYPE_COUNT       0x03

#define RESCTL_PRIV_PRIVE       0x01
#define RESCTL_PRIV_PRIVD       0x02
#define RESCTL_PRIV_BASIC       0x04
#define RESCTL_PRIV_ALLPR       0x07

#define RESCTL_ACTN_NONE        0x01
#define RESCTL_ACTN_DENY        0x02
#define RESCTL_ACTN_SIGN        0x04
#define RESCTL_ACTN_ALLA        0x07

#define RESCTL_SIG_ABRT         0x01
#define RESCTL_SIG_XRES         0x02
#define RESCTL_SIG_HUP          0x04
#define RESCTL_SIG_STOP         0x08
#define RESCTL_SIG_TERM         0x10
#define RESCTL_SIG_KILL         0x20
#define RESCTL_SIG_XFSZ         0x40
#define RESCTL_SIG_XCPU         0x80

#define RESCTL_SIG_CMN          0x3f
#define RESCTL_SIG_ALL          0xff

typedef struct sig_s {
        char sig[6];
        int mask;
} sig_t;

#define SIGS_CNT        16
extern sig_t sigs[SIGS_CNT];

typedef struct resctlrule_s {
        uint64_t resctl_max;
        uint8_t  resctl_type;
        uint8_t  resctl_privs;
        uint8_t  resctl_action;
        uint8_t  resctl_sigs;
} resctlrule_t;

typedef struct resctl_info_s {
        unsigned long long value;
        int flags;
} resctl_info_t;
```



```
extern int resctl_pool_exist(char *);
extern int resctl_get_info(char *, resctl_info_t *);
extern void resctl_get_rule(resctl_info_t *, resctlrule_t *);

#ifdef  __cplusplus
}
#endif
#endif /* _PROJENT_RESCTL_H */
```

```
**********************************************************
    6662 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/util.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <locale.h>
#include <stddef.h>
#include <limits.h>

#include <project.h>

#include <ctype.h>

#include "util.h"



#define BOSTR_REG_EXP   "^"
#define EOSTR_REG_EXP   "$"
#define FLTNM_REG_EXP   "([[:digit:]]+(\\.[[:digit:]]+)?)"
#define MODIF_REG_EXP   "([kmgtpeKMGTPE])?"
#define UNIT__REG_EXP   "([bsBS])?"
#define TOKEN_REG_EXP   "[[:alnum:]_./=+-]*"
#define VALUE_REG_EXP   FLTNM_REG_EXP MODIF_REG_EXP UNIT__REG_EXP

#define TO_EXP(X)       BOSTR_REG_EXP X EOSTR_REG_EXP
#define TOKEN_EXP       TO_EXP(TOKEN_REG_EXP)
#define VALUE_EXP       TO_EXP(VALUE_REG_EXP)

#define BYTES_SCALE     1
#define SCNDS_SCALE     2

char *
util_safe_strdup(char *str)
{
        char *ptr;
        if (str == NULL)
                return (NULL);

        if ((ptr = strdup(str)) == NULL) {
                (void) fprintf(stderr, gettext("error allocating memory"));
                exit(1);
        }
        return (ptr);
}

void *
util_safe_realloc(void *ptr, size_t sz)
{
        if ((ptr = realloc(ptr, sz)) == NULL) {
                (void) fprintf(stderr, gettext(
                    "error reallocating %d bytes of memory"), sz);
                exit(1);
        }
        return (ptr);
}


void *
```



```
util_safe_malloc(size_t sz)
{
        char *ptr;
        if ((ptr = malloc(sz)) == NULL) {
                (void) fprintf(stderr, gettext(
                    "error allocating %d bytes of memory\n"), sz);
                exit(1);
        }
        return (ptr);
}

void *
util_safe_zmalloc(size_t sz)
{
        return (memset(util_safe_malloc(sz), 0, sz));
}

void
util_print_errmsgs(lst_t *errlst)
{
        char *errmsg;
        while (!lst_is_empty(errlst)) {
                errmsg = lst_at(errlst, 0);
                (void) lst_remove(errlst, errmsg);
                (void) fprintf(stderr, "%s\n", errmsg);
                free(errmsg);
        }
}


void
util_add_errmsg(lst_t *errlst, char *format, ...)
{
        va_list args;
        char *errmsg;

        va_start(args, format);
        if (vasprintf(&errmsg, format, args) < 0) {
                va_end(args);
                (void) fprintf(stderr, gettext(
                    "error allocating memory\n"));
                exit(1);
        }
        va_end(args);
        lst_insert_tail(errlst, errmsg);
}

char *
util_str_append(char *str, int nargs, ...)
{
        va_list ap;
        int i, len;
        char *s;

        if (str == NULL)
                str = util_safe_zmalloc(1);

        len = strlen(str) + 1;
        va_start(ap, nargs);
        for (i = 0; i < nargs; i++) {
                s = va_arg(ap, char *);
                len += strlen(s);
                str = util_safe_realloc(str, len);
                (void) strcat(str, s);
        }
        va_end(ap);
```

```
128         return (str);
129 }

131 char *
132 util_substr(regex_t *reg, regmatch_t *mat, char *str, int idx)
133 {
134         int mat_len;
135         char *ret;

137         if (idx < 0 || idx > reg->re_nsub)
138                 return (NULL);

140         mat_len = mat[idx].rm_eo - mat[idx].rm_so;

142         ret = util_safe_malloc(mat_len + 1);
143         *ret = '\0';

145         (void) strlcpy(ret, str + mat[idx].rm_so, mat_len + 1);
146         return (ret);
147 }

149 typedef struct {
150         char unit;
151         uint64_t val;
152 } scl;

154 #define SCLS    7

156 int
157 util_scale(char *unit, int scale, uint64_t *res, lst_t *errlst)
158 {
159         int i;
160         scl *sc;
161         scl bscale[SCLS] = {
162                 {'\0', 1ULL},
163                 {'k', 1024ULL},
164                 {'m', 1048576ULL},
165                 {'g', 1073741824ULL},
166                 {'t', 1099511627776ULL},
167                 {'p', 1125899906842624ULL},
168                 {'e', 1152921504606846976ULL},
169         };

171         scl oscale[SCLS] = {
172                 {'\0', 1ULL},
173                 {'k', 1000ULL},
174                 {'m', 1000000ULL},
175                 {'g', 1000000000ULL},
176                 {'t', 1000000000000ULL},
177                 {'p', 1000000000000000ULL},
178                 {'e', 1000000000000000000ULL},
179         };

181         sc = (scale == BYTES_SCALE) ? bscale : oscale;

183         for (i = 0; i < SCLS; i++) {
184                 if (tolower(*unit) == sc[i].unit) {
185                         *res = sc[i].val;
186                         return (0);
187                 }
188         }

190         util_add_errmsg(errlst, gettext(
191             "Invalid scale: %d"), scale);
192         return (1);
193 }
```



```
196 int
197 util_val2num(char *value, int scale, lst_t *errlst, char **retnum,
198     char **retmod, char **retunit)
199 {
200         int ret = 1;
201         regex_t valueexp;
202         regmatch_t *mat;
203         int nmatch;
204         char *num, *modifier, *unit;
205         char *ptr;

207         uint64_t mul64;
208         long double dnum;

210         *retnum = *retmod = *retunit = NULL;

212         if (regcomp(&valueexp, VALUE_EXP, REG_EXTENDED) != 0) {
213                 util_add_errmsg(errlst, gettext(
214                     "Failed to compile regex: '%s'"), VALUE_EXP);
215                 return (1);
216         }

218         nmatch = valueexp.re_nsub + 1;
219         mat = util_safe_malloc(nmatch * sizeof (regmatch_t));

221         if (regexec(&valueexp, value, nmatch, mat, 0) != 0) {
222                 util_add_errmsg(errlst, gettext(
223                     "Invalid value: '%s'"), value);
224                 regfree(&valueexp);
225                 return (1);
226         }
227         regfree(&valueexp);

229         num = util_substr(&valueexp, mat, value, 1);
230         modifier = util_substr(&valueexp, mat, value, 3);
231         unit = util_substr(&valueexp, mat, value, 4);

233         if ((num == NULL || modifier == NULL || unit == NULL) ||
234             (strlen(modifier) == 0 && strchr(num, '.') != NULL) ||
235             (util_scale(modifier, scale, &mul64, errlst) != 0) ||
236             (scale == BYTES_SCALE && *unit != '\0' && tolower(*unit) != 'b') ||
237             (scale == SCNDS_SCALE && *unit != '\0' && tolower(*unit) != 's') ||
238             (scale == COUNT_SCALE && *unit != '\0') ||
239             (scale == UNKWN_SCALE && *unit != '\0')) {
240                 util_add_errmsg(errlst, gettext("Error near: \"%s\""),
241                     value);
242                 goto out;
243         }

245         dnum = strtold(num, &ptr);
246         if (dnum == 0 &&
247             (errno == EINVAL || errno == ERANGE) &&
248             *ptr != '\0') {
249                 util_add_errmsg(errlst, gettext("Invalid value:  \"%s\""),
250                     value);
251                 goto out;
252         }

254         if (UINT64_MAX / mul64 <= dnum) {
255                 util_add_errmsg(errlst, gettext("Too big value:  \"%s\""),
256                     value);
257                 goto out;
258         }
```

```
260         if (asprintf(retnum, "%llu",
261             (unsigned long long)(mul64 * dnum)) == -1) {
262                 goto out;
263         }

265         free(num);
266         *retmod = modifier;
267         *retunit = unit;
268         return (0);
269 out:
270         free(num); free(modifier); free(unit);
271         return (ret);
272 }

274 void
275 util_free_tokens(char **tokens)
276 {
277         char *token;
278         while ((token = *tokens++) != NULL) {
279                 free(token);
280         }
281 }

283 char **
284 util_tokenize(char *values, lst_t *errlst)
285 {
286         char *token, *t;
287         char *v;
288         regex_t tokenexp;
289         char **ctoken, **tokens = NULL;

291         if (regcomp(&tokenexp, TOKEN_EXP, REG_EXTENDED | REG_NOSUB) != 0)
292                 return (tokens);

294         /* Assume each character will be a token + NULL terminating value. */
295         ctoken = tokens = util_safe_malloc(
296             (strlen(values) + 1) * sizeof (char *));
297         token = util_safe_malloc(strlen(values) + 1);

299         v = values;
300         *ctoken = NULL;
301         while (v < (values + strlen(values))) {

303                 /* get the next token */
304                 t = token;
305                 while ((*t = *v++) != '\0') {
306                         if (*t == '(' || *t == ')' || *t == ','||
307                             *v == '(' || *v == ')' || *v == ',') {
308                                 *++t = '\0';
309                                 break;
310                         }
311                         t++;
312                 }

314                 if (strcmp(token, "(") != 0 && strcmp(token, ")") != 0 &&
315                     strcmp(token, ",") != 0) {
316                         if (regexec(&tokenexp, token, 0, NULL, 0) != 0) {
317                                 util_add_errmsg(errlst, gettext(
318                                     "Invalid Character at or near \"%s\""),
319                                     token);
320                                 util_free_tokens(tokens);
321                                 UTIL_FREE_SNULL(tokens);
322                                 goto out1;
323                         }
324                 }
325                 *ctoken++ = util_safe_strdup(token);
```



```
326                 *ctoken = NULL;
327         }

329 out1:
330         free(token);
331         regfree(&tokenexp);
332         return (tokens);
333 }
```

```
**********************************************************
    1106 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/common/util.h
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#ifndef _PROJENT_UTIL_H
#define _PROJENT_UTIL_H

#include <sys/types.h>
#include <regex.h>
#include <sys/varargs.h>

#include "lst.h"

#ifdef  __cplusplus
extern "C" {
#endif

/* UTIL_STR_APPEND */
#define UTIL_STR_APPEND1(S, S1)         util_str_append((S), 1, (S1))
#define UTIL_STR_APPEND2(S, S1, S2)     util_str_append((S), 2, (S1), (S2))

/* UTIL_FREE_SNULL */
#define UTIL_FREE_SNULL(ptr) {  \
        free(ptr);              \
        ptr = NULL;             \
}

#define UNKWN_SCALE     0
#define BYTES_SCALE     1
#define SCNDS_SCALE     2
#define COUNT_SCALE     3


extern char *util_safe_strdup(char *);
extern void *util_safe_realloc(void*, size_t);
extern void *util_safe_malloc(size_t);
extern void *util_safe_zmalloc(size_t);
extern char **util_tokenize(char *, lst_t *);
extern void util_free_tokens(char **);
extern char *util_substr(regex_t *, regmatch_t *, char *, int);
extern char *util_str_append(char *, int, ...);
extern int util_val2num(char *, int, lst_t *, char **, char **, char **);
extern void util_add_errmsg(lst_t *, char *format, ...);
extern void util_print_errmsgs(lst_t *);

#ifdef  __cplusplus
}
#endif
#endif /* _PROJENT_UTIL_H */
```

```
**********************************************************
    1722 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/projadd/Makefile
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG = projadd
OBJS = projadd.o
SRCS =$(OBJS:%.o=%.c) $(COMMON_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/projadd/Makefile.projadd

LDLIBS += -lpool
CFLAGS += $(CCVERBOSE) -I${PROJADDCOMMONDIR}
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-parentheses

CPPFLAGS_sparc += -I$(SRC)/uts/sfmmu
CPPFLAGS_sparc += -I$(SRC)/uts/sun4u/sunfire
CPPFLAGS += $(CPPFLAGS_$(MACH))

FILEMODE= 0555

lint := LINTFLAGS = -muxs -I$(PROJADDCOMMONDIR)

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

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

%.o : $(PROJADDCOMMONDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

clean:
```



```
        -$(RM) $(PROG) $(OBJS) $(COMMON_OBJS)

lint: lint_SRCS

include $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    5012 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/projadd/projadd.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

#include <stdio.h>
#include <stdlib.h>
#include <libintl.h>
#include <locale.h>
#include <errno.h>
#include <string.h>
#include <stddef.h>
#include <sys/types.h>

#include "projent.h"
#include "util.h"


#define CHECK_ERRORS_FREE_PLST(errlst, plst, attrs, ecode) {    \
        if (!lst_is_empty(errlst)) {                            \
                util_print_errmsgs(errlst);                     \
                if (plst != NULL) {                             \
                        projent_free_lst(plst);                 \
                        free(plst);                             \
                }                                               \
                free(attrs);                                    \
                usage();                                        \
                exit(ecode);                                    \
        }                                                       \
}

/*
 * Print usage
 */
static void
usage(void)
{
        (void) fprintf(stderr, gettext(
            "Usage:\n"
            "projadd [-n] [-f filename] [-p projid [-o]] [-c comment]\n"
            "        [-U user[,user...]] [-G group[,group...]]\n"
            "        [-K name[=value[,value...]]] project\n"));
}

/*
```



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

        extern char *optarg;
        extern int optind, optopt;
        projid_t maxpjid = 99;
        lst_t *plst = NULL;
        int flags = 0;
        char *projfile = PROJF_PATH;

        projent_t *ent;
        boolean_t nflag, pflag, oflag;          /* Command line flags.  */
        char *pname, *projidstr, *comment;      /* project fields.      */
        char *users, *groups, *attrs;
        projid_t projid;

        lst_t errlst;

        /* Project file defaults to system project file "/etc/project" */
        users = groups = comment = projidstr = "";

        nflag = pflag = oflag = B_FALSE;
        attrs = util_safe_zmalloc(1);
        lst_create(&errlst);


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

        /* Parse the command line argument list */
        while ((c = getopt(argc, argv, ":hnf:p:oc:U:G:K:")) != EOF)
                switch (c) {
                        case 'h':
                                usage();
                                exit(0);
                                break;
                        case 'n':
                                nflag = B_TRUE;
                                break;
                        case 'f':
                                projfile = optarg;
                                break;
                        case 'p':
                                pflag = B_TRUE;
                                projidstr = optarg;
                                break;
                        case 'o':
                                oflag = B_TRUE;
                                break;
                        case 'c':
                                comment = optarg;
                                break;
                        case 'U':
                                users = optarg;
                                break;
                        case 'G':
                                groups = optarg;
                                break;
                        case 'K':
```

```
128                                 attrs = UTIL_STR_APPEND2(attrs, ";", optarg);
129                                 break;
130                         default:
131                                 util_add_errmsg(&errlst, gettext(
132                                     "Invalid option: -%c"), optopt);
133                                 break;
134                 }



138         if (oflag && !pflag) {
139                 util_add_errmsg(&errlst, gettext(
140                     "-o requires -p projid to be specified"));
141         }

143         if (optind != argc -1) {
144                 util_add_errmsg(&errlst, gettext("No project name specified"));
145         }

147         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

149         if (!nflag)
150                 flags |= F_PAR_VLD;
151         flags |= F_PAR_RES | F_PAR_DUP;

153         /* Parse the project file to get the list of the projects */
154         plst = projent_get_lst(projfile, flags, &errlst);
155         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);


158         /* Parse and validate new project id */
159         if (pflag && projent_parse_projid(projidstr, &projid, &errlst) == 0) {
160                 if (!nflag) {
161                         projent_validate_projid(projid, 0, &errlst);
162                         if (!oflag) {
163                                 projent_validate_unique_id(plst, projid,
164                                     &errlst);
165                         }
166                 }
167         }
168         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

170         /* Find the maxprojid */
171         for (e = 0; e < lst_size(plst); e++) {
172                 ent = lst_at(plst, e);
173                 maxpjid = (ent->projid > maxpjid) ? ent->projid : maxpjid;
174         }


177         if (!pflag && asprintf(&projidstr, "%ld", maxpjid + 1) == -1) {
178                 util_add_errmsg(&errlst, gettext("Failed to allocate memory"));
179                 CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);
180         }

182         pname = argv[optind];
183         ent = projent_parse_components(pname, projidstr, comment, users,
184             groups, attrs, F_PAR_SPC | F_PAR_UNT, &errlst);
185         if (!pflag)
186                 free(projidstr);

188         if (!nflag)
189                 projent_validate_unique_name(plst, pname, &errlst);

191         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

193         /* Sort attributes list */
```



```
194         projent_sort_attributes(ent->attrs);


197         /* Add the new project entry to the list */
198         lst_insert_tail(plst, ent);

200         /* Validate the projent before writing the list to the project file */
201         (void) projent_validate(ent, flags, &errlst);
202         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

204         /* Write out the project file */
205         projent_put_lst(projfile, plst, &errlst);
206         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

208         return (0);
209 }
```

```
**********************************************************
    1722 Thu Dec 15 19:51:09 2016
new/usr/src/cmd/projadd/projdel/Makefile
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG = projdel
OBJS = projdel.o
SRCS =$(OBJS:%.o=%.c) $(COMMON_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/projadd/Makefile.projadd

LDLIBS += -lpool
CFLAGS += $(CCVERBOSE) -I${PROJADDCOMMONDIR}
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-parentheses

CPPFLAGS_sparc += -I$(SRC)/uts/sfmmu
CPPFLAGS_sparc += -I$(SRC)/uts/sun4u/sunfire
CPPFLAGS += $(CPPFLAGS_$(MACH))

FILEMODE= 0555

lint := LINTFLAGS = -muxs -I$(PROJADDCOMMONDIR)

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

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

%.o : $(PROJADDCOMMONDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

clean:
```



```
        -$(RM) $(PROG) $(OBJS) $(COMMON_OBJS)

lint: lint_SRCS

include $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    3275 Thu Dec 15 19:51:10 2016
new/usr/src/cmd/projadd/projdel/projdel.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

#include <stdio.h>
#include <stdlib.h>
#include <libintl.h>
#include <locale.h>
#include <errno.h>
#include <string.h>
#include <stddef.h>
#include <sys/types.h>

#include "projent.h"
#include "util.h"

#define SEQU(str1, str2)                (strcmp(str1, str2) == 0)

/*
 * Print usage
 */
static void
usage(void)
{
        (void) fprintf(stderr, gettext(
            "Usage:\n"
            "projdel [-f filename] project\n"));
}

/*
 * main()
 */
int
main(int argc, char **argv)
{
        int e, c, ret = 0;
        int flags;
        extern char *optarg;
        extern int optind, optopt;
        lst_t *plst;                    /* Projects list */
        projent_t *ent, *delent;
        int del;
        char *pname;                    /* Project name */
        lst_t errlst;                   /* Errors list */
```



```
        char *projfile = PROJF_PATH;    /* Project file "/etc/project" */

        lst_create(&errlst);


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

        /* Parse the command line argument list */
        while ((c = getopt(argc, argv, ":hf:")) != EOF)
                switch (c) {
                        case 'h':
                                usage();
                                exit(0);
                                break;
                        case 'f':
                                projfile = optarg;
                                break;
                        default:
                                util_add_errmsg(&errlst, gettext(
                                    "Invalid option: -%c"), optopt);
                                break;
                }

        if (optind != argc -1) {
                (void) fprintf(stderr, gettext("No project name specified\n"));
                exit(2);
        }

        /* Name of the project to delete */
        pname = argv[optind];

        flags = F_PAR_VLD | F_PAR_RES | F_PAR_DUP;

        /* Parse the project file to get the list of the projects */
        plst = projent_get_lst(projfile, flags, &errlst);

        if (!lst_is_empty(&errlst)) {
                util_print_errmsgs(&errlst);
                usage();
                exit(2);
        }

        /* Find the project to be deleted */
        del = 0;
        for (e = 0; e < lst_size(plst); e++) {
                ent = lst_at(plst, e);
                if (SEQU(ent->projname, pname)) {
                        del++;
                        delent = ent;
                }
        }

        if (del == 0) {
                (void) fprintf(stderr, gettext(
                    "Project \"%s\" does not exist\n"), pname);
                usage();
                ret = 2;
                goto out;
        } else if (del > 1) {
                (void) fprintf(stderr, gettext(
                    "Duplicate project name \"%s\""), pname);
                usage();
```

```
128                 ret = 2;
129                 goto out;
130         }

132         /* Remove the project entry from the list */
133         (void) lst_remove(plst, delent);

135         /* Write out the project file */
136         projent_put_lst(projfile, plst, &errlst);

138         if (!lst_is_empty(&errlst)) {
139                 util_print_errmsgs(&errlst);
140                 usage();
141                 ret = 2;
142         }
143 out:
144         projent_free_lst(plst);
145         free(plst);
146         return (ret);
147 }
```

```
**********************************************************
    1732 Thu Dec 15 19:51:10 2016
new/usr/src/cmd/projadd/projmod/Makefile
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG = projmod
OBJS = projmod.o
SRCS =$(OBJS:%.o=%.c) $(COMMON_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/projadd/Makefile.projadd

LDLIBS += -lpool -lproject
CFLAGS += $(CCVERBOSE) -I${PROJADDCOMMONDIR}
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-parentheses

CPPFLAGS_sparc += -I$(SRC)/uts/sfmmu
CPPFLAGS_sparc += -I$(SRC)/uts/sun4u/sunfire
CPPFLAGS += $(CPPFLAGS_$(MACH))

FILEMODE= 0555

lint := LINTFLAGS = -muxs -I$(PROJADDCOMMONDIR)

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

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

%.o : $(PROJADDCOMMONDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

clean:
```



```
        -$(RM) $(PROG) $(OBJS) $(COMMON_OBJS)

lint: lint_SRCS

include $(SRC)/cmd/Makefile.targ
```

```
**********************************************************
    9347 Thu Dec 15 19:51:10 2016
new/usr/src/cmd/projadd/projmod/projmod.c
Want projadd, projdel and projmod in C.
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

#include <stdio.h>
#include <stdlib.h>
#include <libintl.h>
#include <locale.h>
#include <errno.h>
#include <string.h>
#include <stddef.h>
#include <sys/types.h>
#include <sys/task.h>

#include <sys/debug.h>


#include "projent.h"
#include "util.h"

#define SEQU(str1, str2)                (strcmp(str1, str2) == 0)

#define CHECK_ERRORS_FREE_PLST(errlst, plst, attrs, ecode) {    \
        if (!lst_is_empty(errlst)) {                            \
                util_print_errmsgs(errlst);                     \
                if (plst != NULL) {                             \
                        projent_free_lst(plst);                 \
                        free(plst);                             \
                }                                               \
                free(attrs);                                    \
                usage();                                        \
                exit(ecode);                                    \
        }                                                       \
}


/*
 * Print usage
 */
static void
usage(void)
{
        (void) fprintf(stderr, gettext(
            "Usage:\n"
            "projmod [-n] [-A|-f filename] [-p projid [-o]] [-c comment]\n"
            "        [-a|-s|-r] [-U user[,user...]] [-G group[,group...]]\n"
            "        [-K name[=value[,value...]]] [-l new_projectname]\n"
            "        project\n"));
}


/*
 * main()
 */
int
main(int argc, char **argv)
{
        int e, c, error;

        extern char *optarg;
        extern int optind, optopt;
        lst_t *plst = NULL;
        int flags = 0;
        projent_t *ent, *modent;

        /* Command line options */
        boolean_t fflag, nflag, cflag, oflag, pflag, lflag;
        boolean_t sflag, rflag, aflag;
        boolean_t Uflag, Gflag, Kflag, Aflag;
        boolean_t modify;

        /* Project entry fields */
        char *pname, *npname;
        char *comment, *users, *groups, *attrs;
        char *pusers, *pgroups;

        lst_t *pattribs;

        lst_t errlst;

        /* Project file defaults to system project file "/etc/project" */
        char *projfile = PROJF_PATH;
        struct project proj, *projp;
        char buf[PROJECT_BUFSZ];
        char *str;

        comment = users = groups = "";
        pname = npname = NULL;

        fflag = nflag = cflag = oflag = pflag = lflag = B_FALSE;
        sflag = rflag = aflag = B_FALSE;
        Uflag = Gflag = Kflag = Aflag = B_FALSE;

        modify = B_FALSE;



        attrs = util_safe_zmalloc(1);
        lst_create(&errlst);


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

        /* Parse the command line argument list */
        while ((c = getopt(argc, argv, ":hf:nc:op:l:sraU:G:K:A")) != EOF)
                switch (c) {
```

```
128                         case 'h':
129                                 usage();
130                                 exit(0);
131                                 break;
132                         case 'f':
133                                 fflag = B_TRUE;
134                                 projfile = optarg;
135                                 break;
136                         case 'n':
137                                 nflag = B_TRUE;
138                                 break;
139                         case 'c':
140                                 cflag = B_TRUE;
141                                 comment = optarg;
142                                 break;
143                         case 'o':
144                                 oflag = B_TRUE;
145                                 break;
146                         case 'p':
147                                 pflag = B_TRUE;
148                                 break;
149                         case 'l':
150                                 lflag = B_TRUE;
151                                 npname = optarg;
152                                 break;
153                         case 's':
154                                 sflag = B_TRUE;
155                                 break;
156                         case 'r':
157                                 rflag = B_TRUE;
158                                 break;
159                         case 'a':
160                                 aflag = B_TRUE;
161                                 break;
162                         case 'U':
163                                 Uflag = B_TRUE;
164                                 users = optarg;
165                                 break;
166                         case 'G':
167                                 Gflag = B_TRUE;
168                                 groups = optarg;
169                                 break;
170                         case 'K':
171                                 Kflag = B_TRUE;
172                                 attrs = UTIL_STR_APPEND2(attrs, ";", optarg);
173                                 break;
174                         case 'A':
175                                 Aflag = B_TRUE;
176                                 break;
177                         default:
178                                 util_add_errmsg(&errlst, gettext(
179                                     "Invalid option: -%c"), optopt);
180                                 break;
181                 }

183         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

185         if (optind == argc - 1) {
186                 pname = argv[optind];
187         }

189         if (cflag || Gflag || lflag || pflag || Uflag || Kflag || Aflag) {
190                 modify = B_TRUE;
191                 if (pname == NULL) {
192                         util_add_errmsg(&errlst, gettext(
193                             "No project name specified"));
```



```
194                 }
195         } else if (pname != NULL) {
196                 util_add_errmsg(&errlst, gettext(
197                     "missing -c, -G, -l, -p, -U, or -K"));
198         }

200         if (Aflag && fflag) {
201                 util_add_errmsg(&errlst, gettext(
202                     "-A and -f are mutually exclusive"));
203         }

205         if (oflag && !pflag) {
206                 util_add_errmsg(&errlst, gettext(
207                     "-o requires -p projid to be specified"));
208         }

210         if ((aflag && (rflag || sflag)) || (rflag && (aflag || sflag)) ||
211             (sflag && (aflag || rflag))) {
212                 util_add_errmsg(&errlst, gettext(
213                     "-a, -r, and -s are mutually exclusive"));
214         }

216         if ((aflag || rflag || sflag) && !(Uflag || Gflag || Kflag)) {
217                 util_add_errmsg(&errlst, gettext(
218                     "-a, -r, and -s require -U users, -G groups "
219                     "or -K attributes to be specified"));
220         }

222         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

224         if (aflag) {
225                 flags |= F_MOD_ADD;
226         } else if (rflag) {
227                 flags |= F_MOD_REM;
228         } else if (sflag) {
229                 flags |= F_MOD_SUB;
230         } else {
231                 flags |= F_MOD_REP;
232         }
233         if (!nflag) {
234                 flags |= F_PAR_VLD;
235         }
236         flags |= F_PAR_RES | F_PAR_DUP;

238         plst = projent_get_lst(projfile, flags, &errlst);
239         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

241         modent = NULL;
242         if (pname != NULL) {
243                 for (e = 0; e < lst_size(plst); e++) {
244                         ent = lst_at(plst, e);
245                         if (SEQU(ent->projname, pname)) {
246                                 modent = ent;
247                         }
248                 }
249                 if (modent == NULL) {
250                         util_add_errmsg(&errlst, gettext(
251                             "Project \"%s\" does not exist"), pname);
252                 }
253         }

255         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

257         /*
258          * If there is no modification options, simply reading the file, which
259          * includes parsing and verifying, is sufficient.
```

```
260          */
261         if (!modify) {
262                 exit(0);
263         }

265         VERIFY(modent != NULL);

267         if (lflag && projent_parse_name(pname, &errlst) == 0 &&
268             projent_validate_unique_name(plst, npname, &errlst) == 0) {
269                 free(modent->projname);
270                 modent->projname = util_safe_strdup(npname);
271         }

273         if (cflag && projent_parse_comment(comment, &errlst) == 0) {
274                 free(modent->comment);
275                 modent->comment = util_safe_strdup(comment);
276         }

278         if (Uflag) {
279                 pusers = projent_parse_users(users, F_PAR_SPC,
280                     &errlst);
281                 if (pusers != NULL) {
282                         projent_merge_usrgrp("user", &modent->userlist,
283                             pusers, flags, &errlst);
284                         free(pusers);
285                 }
286         }

288         if (Gflag) {
289                 pgroups = projent_parse_groups(groups, F_PAR_SPC,
290                     &errlst);
291                 if (pgroups != NULL) {
292                         projent_merge_usrgrp("group", &modent->grouplist,
293                             pgroups, flags, &errlst);
294                         free(pgroups);
295                 }
296         }

298         if (Kflag) {
299                 pattribs = projent_parse_attributes(attrs, F_PAR_UNT, &errlst);
300                 if (pattribs != NULL) {
301                         projent_merge_attributes(&modent->attrs,
302                             pattribs, flags, &errlst);
303                         projent_sort_attributes(modent->attrs);
304                         projent_free_attributes(pattribs);
305                         UTIL_FREE_SNULL(pattribs);
306                 }
307         }

309         if (!nflag) {
310                 (void) projent_validate(modent, flags, &errlst);
311         }
312         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

314         if (modify) {
315                 projent_put_lst(projfile, plst, &errlst);
316         }
317         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

319         if (Aflag && (error = setproject(pname, "root",
320             TASK_FINAL|TASK_PROJ_PURGE)) != 0) {
321                 if (error == SETPROJ_ERR_TASK) {
322                         if (errno == EAGAIN) {
323                                 util_add_errmsg(&errlst, gettext(
324                                     "resource control limit has been reached"));
325                         } else if (errno == ESRCH) {
```



```
326                                 util_add_errmsg(&errlst, gettext(
327                                     "user \"%s\" is not a member "
328                                     "of project \"%s\""), "root", pname);
329                         } else {
330                                 util_add_errmsg(&errlst, gettext(
331                                     "could not join project \"%s\""), pname);
332                         }
333                 } else if (error == SETPROJ_ERR_POOL) {
334                         if (errno == EACCES) {
335                                 util_add_errmsg(&errlst, gettext(
336                                     "no resource pool accepting default "
337                                     "bindings exists for project \"%s\""),
338                                     pname);
339                         } else if (errno == ESRCH) {
340                                 util_add_errmsg(&errlst, gettext(
341                                     "specified resource pool does not exist "
342                                     "for project \"%s\""), pname);
343                         } else {
344                                 util_add_errmsg(&errlst, gettext(
345                                     "could not bind to default resource pool "
346                                     "for project \"%s\""), pname);
347                         }
348                 } else {
349                         /*
350                          * error represents the position - within the
351                          * semi-colon delimited attribute - that generated
352                          * the error.
353                          */
354                         if (error <= 0) {
355                                 util_add_errmsg(&errlst, gettext(
356                                     "setproject failed for project \"%s\""),
357                                     pname);
358                         } else {
359                                 /* To be completed */
360                                 projp = getprojbyname(pname, &proj, buf,
361                                     sizeof (buf));
362                                 pattribs = (projp != NULL) ?
363                                     projent_parse_attributes(projp->pj_attr,
364                                     0, &errlst) : NULL;
365                                 if (projp != NULL && pattribs != NULL &&
366                                     (str = projent_attrib_tostring(
367                                     lst_at(pattribs, error - 1))) != NULL) {
368                                         util_add_errmsg(&errlst, gettext(
369                                             "warning, \"%s\" resource control "
370                                             "assignment failed for project "
371                                             "\"%s\""), str, pname);
372                                         free(str);
373                                 } else {
374                                         util_add_errmsg(&errlst, gettext(
375                                             "warning, resource control "
376                                             "assignment failed for project "
377                                             "\"%s\" attribute %d"), pname,
378                                             error);
379                                 }

381                                 if (pattribs != NULL) {
382                                         projent_free_attributes(pattribs);
383                                         UTIL_FREE_SNULL(pattribs);
384                                 }
385                         }
386                 }
387         }

389         CHECK_ERRORS_FREE_PLST(&errlst, plst, attrs, 2);

391         return (0);
```

```
392 }
```

```
**********************************************************
    1724 Thu Dec 15 19:51:10 2016
new/usr/src/cmd/projadd/projtest/Makefile
Want projadd, projdel and projmod in C.
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG = projtest
OBJS = projtest.o
SRCS =$(OBJS:%.o=%.c) $(COMMON_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/projadd/Makefile.projadd

LDLIBS += -lpool
CFLAGS += $(CCVERBOSE) -I${PROJADDCOMMONDIR}
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-parentheses

CPPFLAGS_sparc += -I$(SRC)/uts/sfmmu
CPPFLAGS_sparc += -I$(SRC)/uts/sun4u/sunfire
CPPFLAGS += $(CPPFLAGS_$(MACH))

FILEMODE= 0555

lint := LINTFLAGS = -muxs -I$(PROJADDCOMMONDIR)

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

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

%.o : $(PROJADDCOMMONDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

clean:
```



```
        -$(RM) $(PROG) $(OBJS) $(COMMON_OBJS)

lint: lint_SRCS

include $(SRC)/cmd/Makefile.targ
```