```
**********************************************************
   18205 Thu Jan 17 14:35:53 2019
new/usr/src/lib/libproject/common/setproject.c
10105 libproject needs smatch fixes
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright (c) 2012, Joyent, Inc. All rights reserved.
 */


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

#include <ctype.h>
#include <project.h>
#include <rctl.h>
#include <secdb.h>
#include <signal.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <nss_dbdefs.h>
#include <pwd.h>
#include <pool.h>
#include <libproc.h>
#include <priv.h>
#include <priv_utils.h>
#include <zone.h>
#include <sys/pool.h>
#include <sys/pool_impl.h>
#include <sys/rctl_impl.h>

static void
xstrtolower(char *s)
{
        for (; *s != '\0'; s++)
                *s = tolower(*s);
}
_____unchanged_portion_omitted_

/*
 * A pool_name of NULL means to attempt to bind to the default pool.
 * If the "force" flag is non-zero, the value of "system.bind-default" will be
```



```
 * ignored, and the process will be bound to the default pool if one exists.
 */
static int
bind_to_pool(const char *pool_name, pid_t pid, int force)
{
        pool_value_t *pvals[] = { NULL, NULL };
        pool_t **pools;
        uint_t nelem;
        uchar_t bval;
        pool_conf_t *conf;
        const char *nm;
        int retval;

        if ((conf = pool_conf_alloc()) == NULL)
                return (-1);
        if (pool_conf_open(conf, pool_dynamic_location(), PO_RDONLY) < 0) {
                /*
                 * Pools configuration file is corrupted; allow logins.
                 */
                pool_conf_free(conf);
                return (0);
        }
        if (pool_name != NULL && pool_get_pool(conf, pool_name) != NULL) {
                /*
                 * There was a project.pool entry, and the pool it refers to
                 * is a valid (active) pool.
                 */
                (void) pool_conf_close(conf);
                pool_conf_free(conf);
                if (pool_set_binding(pool_name, P_PID, pid) != PO_SUCCESS) {
                        if (pool_error() != POE_SYSTEM)
                                errno = EINVAL;
                        return (-1);
                }
                return (0);
        }

        /*
         * Bind to the pool with 'pool.default' = 'true' if
         * 'system.bind-default' = 'true'.
         */
        if ((pvals[0] = pool_value_alloc()) == NULL) {
                (void) pool_conf_close(conf);
                pool_conf_close(conf);
                pool_conf_free(conf);
                return (-1);
        }
        if (!force && pool_get_property(conf, pool_conf_to_elem(conf),
            "system.bind-default", pvals[0]) != POC_BOOL ||
            pool_value_get_bool(pvals[0], &bval) != PO_SUCCESS ||
            bval == PO_FALSE) {
                pool_value_free(pvals[0]);
                (void) pool_conf_close(conf);
                pool_conf_close(conf);
                pool_conf_free(conf);
                errno = pool_name == NULL ? EACCES : ESRCH;
                return (-1);
        }
        (void) pool_value_set_name(pvals[0], "pool.default");
        pool_value_set_bool(pvals[0], PO_TRUE);
        if ((pools = pool_query_pools(conf, &nelem, pvals)) == NULL) {
                /*
                 * No default pools exist.
                 */
                pool_value_free(pvals[0]);
                (void) pool_conf_close(conf);
```

```
525                 pool_conf_close(conf);
526                 pool_conf_free(conf);
527                 errno = pool_name == NULL ? EACCES : ESRCH;
528                 return (-1);
529         }
530         if (nelem != 1 ||
531             pool_get_property(conf, pool_to_elem(conf, pools[0]), "pool.name",
532             pvals[0]) != POC_STRING) {
533                 /*
534                  * Configuration is invalid.
535                  */
536                 free(pools);
537                 pool_value_free(pvals[0]);
538                 (void) pool_conf_close(conf);
539                 pool_conf_free(conf);
540                 return (0);
541         }
542         free(pools);
543         (void) pool_conf_close(conf);
544         pool_conf_free(conf);
545         (void) pool_value_get_string(pvals[0], &nm);
546         if (pool_set_binding(nm, P_PID, pid) != PO_SUCCESS) {
547                 if (pool_error() != POE_SYSTEM)
548                         errno = EINVAL;
549                 retval = -1;
550         } else {
551                 retval = 0;
552         }
553         pool_value_free(pvals[0]);
554         return (retval);
555 }
```

_____***unchanged_portion_omitted_***