```
**********************************************************
    3700 Thu Jan 17 14:40:13 2019
new/usr/src/lib/librpcsvc/common/rusers_simple.c
10107 librpcsvc needs smatch fixes
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * rusers_simple.c
 * These are the "easy to use" interfaces to rusers.
 *
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <string.h>
#include <rpc/rpc.h>
#include <rpcsvc/rusers.h>
#include <stdlib.h>

int
rusers3(host, uap)
        char *host;
        utmp_array *uap;
{
        struct utmpidlearr up;

        if (rpc_call(host, RUSERSPROG, RUSERSVERS_3, RUSERSPROC_NAMES,
                        xdr_void, (char *) NULL,
                        xdr_utmp_array, (char *) uap, (char *) NULL) != 0) {
                /*
                 * If version 3 isn't available, try version 2.  We'll have to
                 * convert a utmpidlearr structure into a utmp_array.
                 */
                up.uia_cnt = 0;
                up.uia_arr = NULL;
                if (rusers(host, &up) != 0)
                        return (-1);
                else {
                        int i;
                        struct ru_utmp forsize;
```



```
                        rusers_utmp *rutp;

                        uap->utmp_array_val = (rusers_utmp *)malloc(up.uia_cnt
                                * sizeof (rusers_utmp));
                        if (uap->utmp_array_val == NULL) {
                                xdr_free(xdr_utmpidlearr, (char *)&up);
                                return (-1);
                        }
                        uap->utmp_array_len = up.uia_cnt;
                        for (rutp = uap->utmp_array_val, i = 0;
                                i < up.uia_cnt; rutp++, i++) {
                                rutp->ut_line = (char *)malloc(sizeof
                                        (forsize.ut_line)+1);
                                rutp->ut_user = (char *)malloc(sizeof
                                        (forsize.ut_name)+1);
                                rutp->ut_host = (char *)malloc(sizeof
                                        (forsize.ut_host)+1);
                                if (rutp->ut_line == NULL ||
                                        rutp->ut_user == NULL ||
                                        rutp->ut_host == NULL) {

                                        while (--rutp >= uap->utmp_array_val) {
                                                free(rutp->ut_line);
                                                free(rutp->ut_user);
                                                free(rutp->ut_host);
                                        }
                                        free(uap->utmp_array_val);
                                        xdr_free(xdr_utmpidlearr, (char *)&up);
                                        return (-1);
                                }
                                (void) strncpy(rutp->ut_line,
                                strncpy(rutp->ut_line,
                                        up.uia_arr[i]->ui_utmp.ut_line,
                                        sizeof (forsize.ut_line)+1);
                                (void) strncpy(rutp->ut_user,
                                strncpy(rutp->ut_user,
                                        up.uia_arr[i]->ui_utmp.ut_name,
                                        sizeof (forsize.ut_name)+1);
                                (void) strncpy(rutp->ut_host,
                                strncpy(rutp->ut_host,
                                        up.uia_arr[i]->ui_utmp.ut_host,
                                        sizeof (forsize.ut_host)+1);
                                rutp->ut_idle = up.uia_arr[i]->ui_idle;
                                rutp->ut_time = up.uia_arr[i]->ui_utmp.ut_time;
                                rutp->ut_type = RUSERS_USER_PROCESS;
                                                        /* assume this */
                        }
                        xdr_free(xdr_utmpidlearr, (char *)&up);
                }
        }
        return (0);
}
______unchanged_portion_omitted_
```