```
************************************************************
   26163 Sat Nov 15 11:43:51 2014
new/usr/src/lib/libnsl/rpc/clnt_dg.c
4962 libnsl: unused variable in clnt_dg_geterr()
************************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
 */

/* Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T */
/* All Rights Reserved */
/*
 * Portions of this source code were derived from Berkeley
 * 4.3 BSD under license from the Regents of the University of
 * California.
 */
/*
 * Copyright 2014 Shruti V Sampat <shrutisampat@gmail.com>
 */

/*
 * Implements a connectionless client side RPC.
 */

#include "mt.h"
#include "rpc_mt.h"
#include <assert.h>
#include <rpc/rpc.h>
#include <errno.h>
#include <sys/poll.h>
#include <syslog.h>
#include <sys/types.h>
#include <sys/kstat.h>
#include <sys/time.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <strings.h>
#include <note.h>
```

```
extern int __rpc_timeval_to_msec(struct timeval *);
extern bool_t xdr_opaque_auth(XDR *, struct opaque_auth *);
extern bool_t __rpc_gss_wrap(AUTH *, char *, uint_t, XDR *, bool_t (*)(),
                                                                caddr_t);
extern bool_t __rpc_gss_unwrap(AUTH *, XDR *, bool_t (*)(), caddr_t);


static struct clnt_ops *clnt_dg_ops(void);
static bool_t time_not_ok(struct timeval *);

/*
 *      This machinery implements per-fd locks for MT-safety.  It is not
 *      sufficient to do per-CLIENT handle locks for MT-safety because a
 *      user may create more than one CLIENT handle with the same fd behind
 *      it.
 *
 *      The current implementation holds locks across the entire RPC and reply,
 *      including retransmissions.  Yes, this is silly, and as soon as this
 *      code is proven to work, this should be the first thing fixed.  One step
 *      at a time.
 */

/*
 * FD Lock handle used by various MT sync. routines
 */
static mutex_t dgtbl_lock = DEFAULTMUTEX;
static  void    *dgtbl = NULL;

static const char mem_err_clnt_dg[] = "clnt_dg_create: out of memory";


#define MCALL_MSG_SIZE 24

/*
 * Private data kept per client handle
 */
struct cu_data {
        int                     cu_fd;          /* connections fd */
        bool_t                  cu_closeit;     /* opened by library */
        struct netbuf           cu_raddr;       /* remote address */
        struct timeval          cu_wait;        /* retransmit interval */
        struct timeval          cu_total;       /* total time for the call */
        struct rpc_err          cu_error;
        struct t_unitdata       *cu_tr_data;
        XDR                     cu_outxdrs;
        char                    *cu_outbuf_start;
        char                    cu_outbuf[MCALL_MSG_SIZE];
        uint_t                  cu_xdrpos;
        uint_t                  cu_sendsz;      /* send size */
        uint_t                  cu_recvsz;      /* recv size */
        struct pollfd           pfdp;
        char                    cu_inbuf[1];
};
______unchanged_portion_omitted_

static void
clnt_dg_geterr(CLIENT *cl, struct rpc_err *errp)
{
        NOTE(ARGUNUSED(cl))
/* LINTED pointer alignment */
        struct cu_data *cu = (struct cu_data *)cl->cl_private;

        *errp = rpc_callerr;
}
______unchanged_portion_omitted_
```