```
**********************************************************
    3425 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/bnu/uuglist.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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


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

#include        "uucp.h"

#define MAXLENGTH 256
#define C_MAX     512

static void insert();
void rproc(), uproc();

static char  Nnament[MAXLENGTH][NAMESIZE];
static char *Nptr[MAXLENGTH];
static short Nnames = 0;

int
main(argc, argv)
int argc;
char **argv;
{
        int c, i, uopt = 0;
        char prev[2 * NAMESIZE];

        if (eaccess(GRADES, 04) == -1) {
                (void) fprintf(stderr, "No administrator defined service grades
                exit(0);
        }

        while ((c = getopt(argc, argv, "x:u")) != EOF)
                switch(c) {
                case 'u':
```



```
                        uopt++;
                        break;
                case 'x':
                        Debug = atoi(optarg);
                        if (Debug < 0)
                                Debug = 1;
                        break;
                default:
                        (void) fprintf(stderr, "usage: uuglist [-u] [-xLEVEL]\n"
                        exit(-1);
                }

        if (uopt) {
                Uid = getuid();

                if (Uid == 0)
                        (void) setuid(UUCPUID);

                (void) guinfo(Uid, User);

                uproc();
        } else
                rproc();

        for (i = 0; i < Nnames; i++) {

                if (EQUALS(Nptr[i], prev))
                        continue;

                puts(Nptr[i]);
                (void) strcpy(prev, Nptr[i]);
        }
        return (0);
}
```

_____unchanged_portion_omitted_

```
void
rproc()
{
        FILE *cfd;
        char line[BUFSIZ];
        char *carray[C_MAX];
        int na;

        cfd = fopen(GRADES, "r");

        while (rdfulline(cfd, line, BUFSIZ) != 0) {

                (void) getargs(line, carray, C_MAX);
                na = getargs(line, carray, C_MAX);
                insert(carray[0]);
        }

        (void) fclose(cfd);
        return;
}
```

_____unchanged_portion_omitted_

```
**********************************************************
   38453 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/bnu/uustat.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

#pragma ident   "%Z%%M% %I%     %E% SMI"
#include <time.h>
#include "uucp.h"

#ifdef  V7
#define O_RDONLY        0
#endif
#define KILLMSG "the system administrator has killed job"
#define USAGE1  "[-q] | [-m] | [-k JOB [-n]] | [-r JOB [-n]] | [-p]"
#define USAGE2  "[-a] [-s SYSTEM [-j]] [-u USER] [-S STATE]"
#define USAGE3  "-t SYSTEM [-d number] [-c]"
#define LOCK "LCK.."
#define STST_MAX        132
#define MAXDATE         12
#define MINTIME         60
#define MINUTES         60
#define CHAR            "a"
#define MAXSTATE        4
/* #include "logs.h" */
struct m {
        char    mach[15];               /* machine name */
        char    locked;
        int     ccount, xcount;
        int     count, type;
        long    retrytime;
        time_t lasttime;
        short   c_age;                  /* age of oldest C. file */
        short   x_age;                  /* age of oldest X. file */
        char    stst[STST_MAX];
} M[UUSTAT_TBL+2];
```

_____unchanged_portion_omitted_


```
static void
friendlytime(uplimit, lolimit)
char *uplimit, *lolimit;
{

        char c;

        c = *(uplimit+6);
        friendlyptr->uhour[0] = *(uplimit+6);
        friendlyptr->uhour[1] = *(uplimit+7);
        friendlyptr->lhour[0] = *(lolimit+6);
        friendlyptr->lhour[1] = *(lolimit+7);
        friendlyptr->umin[0]  = *(uplimit+8);
        friendlyptr->umin[1]  = *(uplimit+9);
        friendlyptr->lmin[0]  = *(lolimit+8);
        friendlyptr->lmin[1]  = *(lolimit+9);

        friendlyptr->uhour[2] = '\0';
        friendlyptr->lhour[2] = '\0';
        friendlyptr->umin[2] = '\0';
        friendlyptr->lmin[2] = '\0';
        return;
}
```

_____unchanged_portion_omitted_

```
**********************************************************
    3573 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/cmd-inet/usr.bin/talk/get_names.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 1997 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
/*        All Rights Reserved   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

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

#include "talk.h"
#include "ctl.h"
#include <locale.h>
#include <pwd.h>
#include <sys/systeminfo.h>

char *getlogin(), *ttyname(int);

extern CTL_MSG msg;

/*
 * Determine the local and remote user, tty, and machines
 */

struct hostent *gethostbyname();

void
get_names(argc, argv)
int argc;
```



```
char *argv[];
{
        char hostname[HOST_NAME_LENGTH + 1];
        char *rem_name;
        char *my_name;
        char *my_machine_name;
        char *rem_machine_name;
        char *my_tty;
        char *rem_tty;
        char *ptr;
        int name_length;

        if (argc < 2) {
                fprintf(stderr,
                    "Usage: talk %s\n", gettext("address [terminal]"));
                exit(1);
        }
        if (!isatty(0)) {
                fprintf(stderr,
        gettext("Standard input must be a tty, not a pipe or a file\n"));
                exit(1);
        }

        if (!isatty(1)) {
                fprintf(stderr,
        gettext("Standard output must be a tty, not a pipe or a file\n"));
                exit(1);
        }

        if ((my_name = getlogin()) == NULL) {
        struct passwd *pass = getpwuid(getuid());
        if (pass != NULL)
                my_name = pass->pw_name;
        }
        if (my_name == NULL) {
                fprintf(stderr,
        gettext("Who are you?  You have no entry in /etc/utmp!  Aborting..\n"));
                exit(1);
        }

        name_length = HOST_NAME_LENGTH;
        (void) sysinfo(SI_HOSTNAME, hostname, name_length);
        my_machine_name = hostname;

        my_tty = strrchr(ttyname(0), '/') + 1;

        /*
         * check for, and strip out, the machine name of the target
         */

        for (ptr = argv[1]; *ptr != '\0' &&
                            *ptr != '@' &&
                            *ptr != ':' &&
                            *ptr != '!' &&
                            *ptr != '.'; ptr++) {
        }

        if (*ptr == '\0') {

                /* this is a local to local talk */

        rem_name = argv[1];
        rem_machine_name = my_machine_name;

        } else {
```

```
123         if (*ptr == '@') {
124                 /* user@host */
125                 rem_name = argv[1];
126                 rem_machine_name = ptr + 1;
127         } else {
128                 /* host.user or host!user or host:user */
129                 rem_name = ptr + 1;
130                 rem_machine_name = argv[1];
131         }
132         *ptr = '\0';
133         }


136         if (argc > 2) {
137         rem_tty = argv[2];      /* tty name is arg 2 */
138         } else {
139         rem_tty = "";
140         }

142         get_addrs(my_machine_name, rem_machine_name);

144         /* Load these useful values into the standard message header */

146         msg.id_num = 0;

148         strncpy(msg.l_name, my_name, NAME_SIZE);
149         msg.l_name[NAME_SIZE - 1] = '\0';

151         strncpy(msg.r_name, rem_name, NAME_SIZE);
152         msg.r_name[NAME_SIZE - 1] = '\0';

154         strncpy(msg.r_tty, rem_tty, TTY_SIZE);
155         msg.r_tty[TTY_SIZE - 1] = '\0';
156 }
```

_____***unchanged_portion_omitted_***

****
56618 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/cmd-inet/usr.sbin/ipadm/ipadm.c
8485 Remove set but unused variables in usr/src/cmd
****

```
/*
 * 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) 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2017 Nexenta Systems, Inc.
 * Copyright 2017 Gary Mills
 */

#include <arpa/inet.h>
#include <errno.h>
#include <getopt.h>
#include <inet/ip.h>
#include <inet/iptun.h>
#include <inet/tunables.h>
#include <libdladm.h>
#include <libdliptun.h>
#include <libdllink.h>
#include <libinetutil.h>
#include <libipadm.h>
#include <locale.h>
#include <netdb.h>
#include <netinet/in.h>
#include <ofmt.h>
#include <stdarg.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <zone.h>

#define STR_UNKNOWN_VAL "?"
#define LIFC_DEFAULT    (LIFC_NOXMIT | LIFC_TEMPORARY | LIFC_ALLZONES |\
                         LIFC_UNDER_IPMP)

typedef void cmdfunc_t(int, char **, const char *);
static cmdfunc_t do_create_if, do_delete_if, do_enable_if, do_disable_if;
static cmdfunc_t do_show_if;
static cmdfunc_t do_set_prop, do_show_prop, do_set_ifprop;
static cmdfunc_t do_show_ifprop, do_reset_ifprop, do_reset_prop;
```



```
static cmdfunc_t do_show_addrprop, do_set_addrprop, do_reset_addrprop;
static cmdfunc_t do_create_addr, do_delete_addr, do_show_addr;
static cmdfunc_t do_enable_addr, do_disable_addr;
static cmdfunc_t do_up_addr, do_down_addr, do_refresh_addr;

typedef struct  cmd {
        char            *c_name;
        cmdfunc_t       *c_fn;
        const char      *c_usage;
} cmd_t;
```

*_____unchanged_portion_omitted_*

```
/*
 * Print individual columns for the show-*prop subcommands.
 */
static void
print_prop(show_prop_state_t *statep, uint_t flags, char *buf, size_t bufsize)
{
        const char              *prop_name = statep->sps_pname;
        char                    *ifname = statep->sps_ifname;
        char                    *propval = statep->sps_propval;
        uint_t                  proto = statep->sps_proto;
        size_t                  propsize = MAXPROPVALLEN;
        char                    *object;
        ipadm_status_t          status;

        if (statep->sps_ifprop) {
                status = ipadm_get_ifprop(iph, ifname, prop_name, propval,
                    &propsize, proto, flags);
                object = ifname;
        } else if (statep->sps_modprop) {
                status = ipadm_get_prop(iph, prop_name, propval, &propsize,
                    proto, flags);
                object = ipadm_proto2str(proto);
        } else {
                status = ipadm_get_addrprop(iph, prop_name, propval, &propsize,
                    statep->sps_aobjname, flags);
                object = statep->sps_aobjname;
        }

        if (status != IPADM_SUCCESS) {
                if ((status == IPADM_NOTFOUND && (flags & IPADM_OPT_PERSIST)) ||
                    status == IPADM_ENXIO) {
                        propval[0] = '\0';
                        goto cont;
                }
                statep->sps_status = status;
                statep->sps_retstatus = status;
                return;
        }
cont:
        statep->sps_status = IPADM_SUCCESS;
        (void) snprintf(buf, bufsize, "%s", propval);
}
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   28730 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_dhcpv6.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Dynamic Host Configuration Protocol version 6, for IPv6.  Supports
 * RFCs 3315, 3319, 3646, 3898, 4075, 4242, 4280, 4580, 4649, and 4704.
 */

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/dhcp6.h>
#include <arpa/inet.h>
#include <dhcp_impl.h>
#include <dhcp_inittab.h>

#include "snoop.h"

static const char *mtype_to_str(uint8_t);
static const char *option_to_str(uint8_t);
static const char *duidtype_to_str(uint16_t);
static const char *status_to_str(uint16_t);
static const char *entr_to_str(uint32_t);
static const char *reconf_to_str(uint8_t);
static const char *authproto_to_str(uint8_t);
static const char *authalg_to_str(uint8_t, uint8_t);
static const char *authrdm_to_str(uint8_t);
static const char *cwhat_to_str(uint8_t);
static const char *catype_to_str(uint8_t);
static void show_hex(const uint8_t *, int, const char *);
static void show_ascii(const uint8_t *, int, const char *);
static void show_address(const char *, const void *);
```



```
static void show_options(const uint8_t *, int);

int
interpret_dhcpv6(int flags, const uint8_t *data, int len)
{
        int olen = len;
        char *line, *lstart;
        dhcpv6_relay_t d6r;
        dhcpv6_message_t d6m;
        uint_t optlen;
        uint16_t statuscode;

        if (len <= 0) {
                (void) strlcpy(get_sum_line(), "DHCPv6?", MAXLINE);
                return (0);
        }
        if (flags & F_SUM) {
                uint_t ias;
                dhcpv6_option_t *d6o;
                in6_addr_t link, peer;
                char linkstr[INET6_ADDRSTRLEN];
                char peerstr[INET6_ADDRSTRLEN];

                line = lstart = get_sum_line();
                line += snprintf(line, MAXLINE, "DHCPv6 %s",
                    mtype_to_str(data[0]));
                if (data[0] == DHCPV6_MSG_RELAY_FORW ||
                    data[0] == DHCPV6_MSG_RELAY_REPL) {
                        if (len < sizeof (d6r)) {
                                (void) strlcpy(line, "?",
                                    MAXLINE - (line - lstart));
                                return (olen);
                        }
                        /* Not much in DHCPv6 is aligned. */
                        (void) memcpy(&d6r, data, sizeof (d6r));
                        (void) memcpy(&link, d6r.d6r_linkaddr, sizeof (link));
                        (void) memcpy(&peer, d6r.d6r_peeraddr, sizeof (peer));
                        line += snprintf(line, MAXLINE - (line - lstart),
                            " HC=%d link=%s peer=%s", d6r.d6r_hop_count,
                            inet_ntop(AF_INET6, &link, linkstr,
                            sizeof (linkstr)),
                            inet_ntop(AF_INET6, &peer, peerstr,
                            sizeof (peerstr)));
                        data += sizeof (d6r);
                        len -= sizeof (d6r);
                } else {
                        if (len < sizeof (d6m)) {
                                (void) strlcpy(line, "?",
                                    MAXLINE - (line - lstart));
                                return (olen);
                        }
                        (void) memcpy(&d6m, data, sizeof (d6m));
                        line += snprintf(line, MAXLINE - (line - lstart),
                            " xid=%x", DHCPV6_GET_TRANSID(&d6m));
                        data += sizeof (d6m);
                        len -= sizeof (d6m);
                }
                ias = 0;
                d6o = NULL;
                while ((d6o = dhcpv6_find_option(data, len, d6o,
                    DHCPV6_OPT_IA_NA, NULL)) != NULL)
                        ias++;
                if (ias > 0)
                        line += snprintf(line, MAXLINE - (line - lstart),
                            " IAs=%u", ias);
                d6o = dhcpv6_find_option(data, len, NULL,
```

```
128                     DHCPV6_OPT_STATUS_CODE, &optlen);
129                 optlen -= sizeof (*d6o);
130                 if (d6o != NULL && optlen >= sizeof (statuscode)) {
131                         (void) memcpy(&statuscode, d6o + 1,
132                             sizeof (statuscode));
133                         line += snprintf(line, MAXLINE - (line - lstart),
134                             " status=%u", ntohs(statuscode));
135                         optlen -= sizeof (statuscode);
136                         if (optlen > 0) {
137                                 line += snprintf(line,
138                                     MAXLINE - (line - lstart), " \"%.*s\"",
139                                     optlen, (char *)(d6o + 1) + 2);
140                         }
141                 }
142                 d6o = dhcpv6_find_option(data, len, NULL,
143                     DHCPV6_OPT_RELAY_MSG, &optlen);
144                 optlen -= sizeof (*d6o);
145                 if (d6o != NULL && optlen >= 1) {
146                         line += snprintf(line, MAXLINE - (line - lstart),
147                             " relay=%s", mtype_to_str(*(uint8_t *)(d6o + 1)));
148                 }
149         } else if (flags & F_DTAIL) {
150                 show_header("DHCPv6: ",
151                     "Dynamic Host Configuration Protocol Version 6", len);
152                 show_space();
153                 (void) snprintf(get_line(0, 0), get_line_remain(),
154                     "Message type (msg-type) = %u (%s)", data[0],
155                     mtype_to_str(data[0]));
156                 if (data[0] == DHCPV6_MSG_RELAY_FORW ||
157                     data[0] == DHCPV6_MSG_RELAY_REPL) {
158                         if (len < sizeof (d6r)) {
159                                 (void) strlcpy(get_line(0, 0), "Truncated",
160                                     get_line_remain());
161                                 return (olen);
162                         }
163                         (void) memcpy(&d6r, data, sizeof (d6r));
164                         (void) snprintf(get_line(0, 0), get_line_remain(),
165                             "Hop count = %u", d6r.d6r_hop_count);
166                         show_address("Link address", d6r.d6r_linkaddr);
167                         show_address("Peer address", d6r.d6r_peeraddr);
168                         data += sizeof (d6r);
169                         len -= sizeof (d6r);
170                 } else {
171                         if (len < sizeof (d6m)) {
172                                 (void) strlcpy(get_line(0, 0), "Truncated",
173                                     get_line_remain());
174                                 return (olen);
175                         }
176                         (void) memcpy(&d6m, data, sizeof (d6m));
177                         (void) snprintf(get_line(0, 0), get_line_remain(),
178                             "Transaction ID = %x", DHCPV6_GET_TRANSID(&d6m));
179                         data += sizeof (d6m);
180                         len -= sizeof (d6m);
181                 }
182                 show_space();
183                 show_options(data, len);
184                 show_space();
185         }
186         return (olen);
187 }
```

*unchanged_portion_omitted_*

```
584 static void
585 show_options(const uint8_t *data, int len)
586 {
587         dhcpv6_option_t d6o;
```



```
588         uint_t olen;
589         uint_t olen, retlen;
590         uint16_t val16;
591         uint16_t type;
592         uint32_t val32;
593         const uint8_t *ostart;
594         char *str, *sp;
595         char *oldnest;

596         /*
597          * Be very careful with negative numbers; ANSI signed/unsigned
598          * comparison doesn't work as expected.
599          */
600         while (len >= (signed)sizeof (d6o)) {
601                 (void) memcpy(&d6o, data, sizeof (d6o));
602                 d6o.d6o_code = ntohs(d6o.d6o_code);
603                 d6o.d6o_len = olen = ntohs(d6o.d6o_len);
604                 (void) snprintf(get_line(0, 0), get_line_remain(),
605                     "Option Code = %u (%s)", d6o.d6o_code,
606                     option_to_str(d6o.d6o_code));
607                 ostart = data += sizeof (d6o);
608                 len -= sizeof (d6o);
609                 if (olen > len) {
610                         (void) strlcpy(get_line(0, 0), "Option truncated",
611                             get_line_remain());
612                         olen = len;
613                 }
614                 switch (d6o.d6o_code) {
615                 case DHCPV6_OPT_CLIENTID:
616                 case DHCPV6_OPT_SERVERID:
617                         if (olen < sizeof (val16))
618                                 break;
619                         (void) memcpy(&val16, data, sizeof (val16));
620                         data += sizeof (val16);
621                         olen -= sizeof (val16);
622                         type = ntohs(val16);
623                         (void) snprintf(get_line(0, 0), get_line_remain(),
624                             "  DUID Type = %u (%s)", type,
625                             duidtype_to_str(type));
626                         if (type == DHCPV6_DUID_LLT || type == DHCPV6_DUID_LL) {
627                                 if (olen < sizeof (val16))
628                                         break;
629                                 (void) memcpy(&val16, data, sizeof (val16));
630                                 data += sizeof (val16);
631                                 olen -= sizeof (val16);
632                                 val16 = ntohs(val16);
633                                 (void) snprintf(get_line(0, 0),
634                                     get_line_remain(),
635                                     "  Hardware Type = %u (%s)", val16,
636                                     arp_htype(val16));
637                         }
638                         if (type == DHCPV6_DUID_LLT) {
639                                 time_t timevalue;

641                                 if (olen < sizeof (val32))
642                                         break;
643                                 (void) memcpy(&val32, data, sizeof (val32));
644                                 data += sizeof (val32);
645                                 olen -= sizeof (val32);
646                                 timevalue = ntohl(val32) + DUID_TIME_BASE;
647                                 (void) snprintf(get_line(0, 0),
648                                     get_line_remain(),
649                                     "  Time = %lu (%.24s)", ntohl(val32),
650                                     ctime(&timevalue));
651                         }
652                         if (type == DHCPV6_DUID_EN) {
```

```
653                             if (olen < sizeof (val32))
654                                     break;
655                             (void) memcpy(&val32, data, sizeof (val32));
656                             data += sizeof (val32);
657                             olen -= sizeof (val32);
658                             val32 = ntohl(val32);
659                             (void) snprintf(get_line(0, 0),
660                                 get_line_remain(),
661                                 "  Enterprise Number = %lu (%s)", val32,
662                                 entr_to_str(val32));
663                     }
664                     if (olen == 0)
665                             break;
666                     if ((str = malloc(olen * 3)) == NULL)
667                             pr_err("interpret_dhcpv6: no mem");
668                     sp = str + snprintf(str, 3, "%02x", *data++);
669                     while (--olen > 0) {
670                             *sp++ = (type == DHCPV6_DUID_LLT ||
671                                 type == DHCPV6_DUID_LL) ? ':' : ' ';
672                             sp = sp + snprintf(sp, 3, "%02x", *data++);
673                     }
674                     (void) snprintf(get_line(0, 0), get_line_remain(),
675                         (type == DHCPV6_DUID_LLT ||
676                         type == DHCPV6_DUID_LL) ?
677                         "  Link Layer Address = %s" :
678                         "  Identifier = %s", str);
679                     free(str);
680                     break;
681             case DHCPV6_OPT_IA_NA:
682             case DHCPV6_OPT_IA_PD: {
683                     dhcpv6_ia_na_t d6in;

685                     if (olen < sizeof (d6in) - sizeof (d6o))
686                             break;
687                     (void) memcpy(&d6in, data - sizeof (d6o),
688                         sizeof (d6in));
689                     data += sizeof (d6in) - sizeof (d6o);
690                     olen -= sizeof (d6in) - sizeof (d6o);
691                     (void) snprintf(get_line(0, 0), get_line_remain(),
692                         "  IAID = %u", ntohl(d6in.d6in_iaid));
693                     (void) snprintf(get_line(0, 0), get_line_remain(),
694                         "  T1 (renew) = %u seconds", ntohl(d6in.d6in_t1));
695                     (void) snprintf(get_line(0, 0), get_line_remain(),
696                         "  T2 (rebind) = %u seconds", ntohl(d6in.d6in_t2));
697                     nest_options(data, olen, "IA: ",
698                         "Identity Association");
699                     break;
700             }
701             case DHCPV6_OPT_IA_TA: {
702                     dhcpv6_ia_ta_t d6it;

704                     if (olen < sizeof (d6it) - sizeof (d6o))
705                             break;
706                     (void) memcpy(&d6it, data - sizeof (d6o),
707                         sizeof (d6it));
708                     data += sizeof (d6it) - sizeof (d6o);
709                     olen -= sizeof (d6it) - sizeof (d6o);
710                     (void) snprintf(get_line(0, 0), get_line_remain(),
711                         "  IAID = %u", ntohl(d6it.d6it_iaid));
712                     nest_options(data, olen, "IA: ",
713                         "Identity Association");
714                     break;
715             }
716             case DHCPV6_OPT_IAADDR: {
717                     dhcpv6_iaaddr_t d6ia;
```



```
719                     if (olen < sizeof (d6ia) - sizeof (d6o))
720                             break;
721                     (void) memcpy(&d6ia, data - sizeof (d6o),
722                         sizeof (d6ia));
723                     data += sizeof (d6ia) - sizeof (d6o);
724                     olen -= sizeof (d6ia) - sizeof (d6o);
725                     show_address("  Address", &d6ia.d6ia_addr);
726                     (void) snprintf(get_line(0, 0), get_line_remain(),
727                         "  Preferred lifetime = %u seconds",
728                         ntohl(d6ia.d6ia_preflife));
729                     (void) snprintf(get_line(0, 0), get_line_remain(),
730                         "  Valid lifetime = %u seconds",
731                         ntohl(d6ia.d6ia_vallife));
732                     nest_options(data, olen, "ADDR: ", "Address");
733                     break;
734             }
735             case DHCPV6_OPT_ORO:
736                     while (olen >= sizeof (val16)) {
737                             (void) memcpy(&val16, data, sizeof (val16));
738                             val16 = ntohs(val16);
739                             (void) snprintf(get_line(0, 0),
740                                 get_line_remain(),
741                                 "  Requested Option Code = %u (%s)", val16,
742                                 option_to_str(val16));
743                             data += sizeof (val16);
744                             olen -= sizeof (val16);
745                     }
746                     break;
747             case DHCPV6_OPT_PREFERENCE:
748                     if (olen > 0) {
749                             (void) snprintf(get_line(0, 0),
750                                 get_line_remain(),
751                                 *data == 255 ?
752                                 "  Preference = %u (immediate)" :
753                                 "  Preference = %u", *data);
754                     }
755                     break;
756             case DHCPV6_OPT_ELAPSED_TIME:
757                     if (olen == sizeof (val16)) {
758                             (void) memcpy(&val16, data, sizeof (val16));
759                             val16 = ntohs(val16);
760                             (void) snprintf(get_line(0, 0),
761                                 get_line_remain(),
762                                 "  Elapsed Time = %u.%02u seconds",
763                                 val16 / 100, val16 % 100);
764                     }
765                     break;
766             case DHCPV6_OPT_RELAY_MSG:
767                     if (olen > 0) {
768                             oldnest = prot_nest_prefix;
769                             prot_nest_prefix = prot_prefix;
770                             (void) interpret_dhcpv6(F_DTAIL, data, olen);
769                             retlen = interpret_dhcpv6(F_DTAIL, data, olen);
771                             prot_prefix = prot_nest_prefix;
772                             prot_nest_prefix = oldnest;
773                     }
774                     break;
775             case DHCPV6_OPT_AUTH: {
776                     dhcpv6_auth_t d6a;

778                     if (olen < DHCPV6_AUTH_SIZE - sizeof (d6o))
779                             break;
780                     (void) memcpy(&d6a, data - sizeof (d6o),
781                         DHCPV6_AUTH_SIZE);
782                     data += DHCPV6_AUTH_SIZE - sizeof (d6o);
783                     olen += DHCPV6_AUTH_SIZE - sizeof (d6o);
```

```
784                         (void) snprintf(get_line(0, 0), get_line_remain(),
785                             "  Protocol = %u (%s)", d6a.d6a_proto,
786                             authproto_to_str(d6a.d6a_proto));
787                         (void) snprintf(get_line(0, 0), get_line_remain(),
788                             "  Algorithm = %u (%s)", d6a.d6a_alg,
789                             authalg_to_str(d6a.d6a_proto, d6a.d6a_alg));
790                         (void) snprintf(get_line(0, 0), get_line_remain(),
791                             "  Replay Detection Method = %u (%s)", d6a.d6a_rdm,
792                             authrdm_to_str(d6a.d6a_rdm));
793                         show_hex(d6a.d6a_replay, sizeof (d6a.d6a_replay),
794                             "  RDM Data");
795                         if (olen > 0)
796                                 show_hex(data, olen, "  Auth Info");
797                         break;
798                 }
799                 case DHCPV6_OPT_UNICAST:
800                         if (olen >= sizeof (in6_addr_t))
801                                 show_address("  Server Address", data);
802                         break;
803                 case DHCPV6_OPT_STATUS_CODE:
804                         if (olen < sizeof (val16))
805                                 break;
806                         (void) memcpy(&val16, data, sizeof (val16));
807                         val16 = ntohs(val16);
808                         (void) snprintf(get_line(0, 0), get_line_remain(),
809                             "  Status Code = %u (%s)", val16,
810                             status_to_str(val16));
811                         data += sizeof (val16);
812                         olen -= sizeof (val16);
813                         if (olen > 0)
814                                 (void) snprintf(get_line(0, 0),
815                                     get_line_remain(), "  Text = \"%.*s\"",
816                                     olen, data);
817                         break;
818                 case DHCPV6_OPT_VENDOR_CLASS:
819                         if (olen < sizeof (val32))
820                                 break;
821                         (void) memcpy(&val32, data, sizeof (val32));
822                         data += sizeof (val32);
823                         olen -= sizeof (val32);
824                         val32 = ntohl(val32);
825                         (void) snprintf(get_line(0, 0), get_line_remain(),
826                             "  Enterprise Number = %lu (%s)", val32,
827                             entr_to_str(val32));
828                         /* FALLTHROUGH */
829                 case DHCPV6_OPT_USER_CLASS:
830                         while (olen >= sizeof (val16)) {
831                                 (void) memcpy(&val16, data, sizeof (val16));
832                                 data += sizeof (val16);
833                                 olen -= sizeof (val16);
834                                 val16 = ntohs(val16);
835                                 if (val16 > olen) {
836                                         (void) strlcpy(get_line(0, 0),
837                                             "  Truncated class",
838                                             get_line_remain());
839                                         val16 = olen;
840                                 }
841                                 show_hex(data, olen, "  Class");
842                                 data += val16;
843                                 olen -= val16;
844                         }
845                         break;
846                 case DHCPV6_OPT_VENDOR_OPT: {
847                         dhcpv6_option_t sd6o;

849                         if (olen < sizeof (val32))
```



```
850                                 break;
851                         (void) memcpy(&val32, data, sizeof (val32));
852                         data += sizeof (val32);
853                         olen -= sizeof (val32);
854                         val32 = ntohl(val32);
855                         (void) snprintf(get_line(0, 0), get_line_remain(),
856                             "  Enterprise Number = %lu (%s)", val32,
857                             entr_to_str(val32));
858                         while (olen >= sizeof (sd6o)) {
859                                 (void) memcpy(&sd6o, data, sizeof (sd6o));
860                                 sd6o.d6o_code = ntohs(sd6o.d6o_code);
861                                 sd6o.d6o_len = ntohs(sd6o.d6o_len);
862                                 (void) snprintf(get_line(0, 0),
863                                     get_line_remain(),
864                                     "  Vendor Option Code = %u", d6o.d6o_code);
865                                 data += sizeof (d6o);
866                                 olen -= sizeof (d6o);
867                                 if (sd6o.d6o_len > olen) {
868                                         (void) strlcpy(get_line(0, 0),
869                                             "  Vendor Option truncated",
870                                             get_line_remain());
871                                         sd6o.d6o_len = olen;
872                                 }
873                                 if (sd6o.d6o_len > 0) {
874                                         show_hex(data, sd6o.d6o_len,
875                                             "    Data");
876                                         data += sd6o.d6o_len;
877                                         olen -= sd6o.d6o_len;
878                                 }
879                         }
880                         break;
881                 }
882                 case DHCPV6_OPT_REMOTE_ID:
883                         if (olen < sizeof (val32))
884                                 break;
885                         (void) memcpy(&val32, data, sizeof (val32));
886                         data += sizeof (val32);
887                         olen -= sizeof (val32);
888                         val32 = ntohl(val32);
889                         (void) snprintf(get_line(0, 0), get_line_remain(),
890                             "  Enterprise Number = %lu (%s)", val32,
891                             entr_to_str(val32));
892                         /* FALLTHROUGH */
893                 case DHCPV6_OPT_INTERFACE_ID:
894                 case DHCPV6_OPT_SUBSCRIBER:
895                         if (olen > 0)
896                                 show_hex(data, olen, "  ID");
897                         break;
898                 case DHCPV6_OPT_RECONF_MSG:
899                         if (olen > 0) {
900                                 (void) snprintf(get_line(0, 0),
901                                     get_line_remain(),
902                                     "  Message Type = %u (%s)", *data,
903                                     reconf_to_str(*data));
904                         }
905                         break;
906                 case DHCPV6_OPT_SIP_NAMES:
907                 case DHCPV6_OPT_DNS_SEARCH:
908                 case DHCPV6_OPT_NIS_DOMAIN:
909                 case DHCPV6_OPT_BCMCS_SRV_D: {
910                         dhcp_symbol_t *symp;
911                         char *sp2;

913                         symp = inittab_getbycode(
914                             ITAB_CAT_STANDARD | ITAB_CAT_V6, ITAB_CONS_SNOOP,
915                             d6o.d6o_code);
```

```
916                         if (symp != NULL) {
917                                 str = inittab_decode(symp, data, olen, B_TRUE);
918                                 if (str != NULL) {
919                                         sp = str;
920                                         do {
921                                                 sp2 = strchr(sp, ' ');
922                                                 if (sp2 != NULL)
923                                                         *sp2++ = '\0';
924                                                 (void) snprintf(get_line(0, 0),
925                                                     get_line_remain(),
926                                                     "  Name = %s", sp);
927                                         } while ((sp = sp2) != NULL);
928                                         free(str);
929                                 }
930                                 free(symp);
931                         }
932                         break;
933                 }
934                 case DHCPV6_OPT_SIP_ADDR:
935                 case DHCPV6_OPT_DNS_ADDR:
936                 case DHCPV6_OPT_NIS_SERVERS:
937                 case DHCPV6_OPT_SNTP_SERVERS:
938                 case DHCPV6_OPT_BCMCS_SRV_A:
939                         while (olen >= sizeof (in6_addr_t)) {
940                                 show_address("  Address", data);
941                                 data += sizeof (in6_addr_t);
942                                 olen -= sizeof (in6_addr_t);
943                         }
944                         break;
945                 case DHCPV6_OPT_IAPREFIX: {
946                         dhcpv6_iaprefix_t d6ip;

948                         if (olen < DHCPV6_IAPREFIX_SIZE - sizeof (d6o))
949                                 break;
950                         (void) memcpy(&d6ip, data - sizeof (d6o),
951                             DHCPV6_IAPREFIX_SIZE);
952                         data += DHCPV6_IAPREFIX_SIZE - sizeof (d6o);
953                         olen -= DHCPV6_IAPREFIX_SIZE - sizeof (d6o);
954                         show_address("  Prefix", d6ip.d6ip_addr);
955                         (void) snprintf(get_line(0, 0), get_line_remain(),
956                             "  Preferred lifetime = %u seconds",
957                             ntohl(d6ip.d6ip_preflife));
958                         (void) snprintf(get_line(0, 0), get_line_remain(),
959                             "  Valid lifetime = %u seconds",
960                             ntohl(d6ip.d6ip_vallife));
961                         (void) snprintf(get_line(0, 0), get_line_remain(),
962                             "  Prefix length = %u", d6ip.d6ip_preflen);
963                         nest_options(data, olen, "ADDR: ", "Address");
964                         break;
965                 }
966                 case DHCPV6_OPT_INFO_REFTIME:
967                         if (olen < sizeof (val32))
968                                 break;
969                         (void) memcpy(&val32, data, sizeof (val32));
970                         (void) snprintf(get_line(0, 0), get_line_remain(),
971                             "  Refresh Time = %lu seconds", ntohl(val32));
972                         break;
973                 case DHCPV6_OPT_GEOCONF_CVC: {
974                         dhcpv6_civic_t d6c;
975                         int solen;

977                         if (olen < DHCPV6_CIVIC_SIZE - sizeof (d6o))
978                                 break;
979                         (void) memcpy(&d6c, data - sizeof (d6o),
980                             DHCPV6_CIVIC_SIZE);
981                         data += DHCPV6_CIVIC_SIZE - sizeof (d6o);
```



```
982                         olen -= DHCPV6_CIVIC_SIZE - sizeof (d6o);
983                         (void) snprintf(get_line(0, 0), get_line_remain(),
984                             "  What Location = %u (%s)", d6c.d6c_what,
985                             cwhat_to_str(d6c.d6c_what));
986                         (void) snprintf(get_line(0, 0), get_line_remain(),
987                             "  Country Code = %.*s", sizeof (d6c.d6c_cc),
988                             d6c.d6c_cc);
989                         while (olen >= 2) {
990                                 (void) snprintf(get_line(0, 0),
991                                     get_line_remain(),
992                                     "  CA Element = %u (%s)", *data,
993                                     catype_to_str(*data));
994                                 solen = data[1];
995                                 data += 2;
996                                 olen -= 2;
997                                 if (solen > olen) {
998                                         (void) strlcpy(get_line(0, 0),
999                                             "  CA Element truncated",
1000                                            get_line_remain());
1001                                    solen = olen;
1002                                }
1003                                if (solen > 0) {
1004                                        show_ascii(data, solen, "  CA Data");
1005                                        data += solen;
1006                                        olen -= solen;
1007                                }
1008                        }
1009                        break;
1010                }
1011                case DHCPV6_OPT_CLIENT_FQDN: {
1012                        dhcp_symbol_t *symp;

1014                        if (olen == 0)
1015                                break;
1016                        (void) snprintf(get_line(0, 0), get_line_remain(),
1017                            "  Flags = %02x", *data);
1018                        (void) snprintf(get_line(0, 0), get_line_remain(),
1019                            "        %s", getflag(*data, DHCPV6_FQDNF_S,
1020                            "Perform AAAA RR updates", "No AAAA RR updates"));
1021                        (void) snprintf(get_line(0, 0), get_line_remain(),
1022                            "        %s", getflag(*data, DHCPV6_FQDNF_O,
1023                            "Server override updates",
1024                            "No server override updates"));
1025                        (void) snprintf(get_line(0, 0), get_line_remain(),
1026                            "        %s", getflag(*data, DHCPV6_FQDNF_N,
1027                            "Server performs no updates",
1028                            "Server performs updates"));
1029                        symp = inittab_getbycode(
1030                            ITAB_CAT_STANDARD | ITAB_CAT_V6, ITAB_CONS_SNOOP,
1031                            d6o.d6o_code);
1032                        if (symp != NULL) {
1033                                str = inittab_decode(symp, data, olen, B_TRUE);
1034                                if (str != NULL) {
1035                                        (void) snprintf(get_line(0, 0),
1036                                            get_line_remain(),
1037                                            "  FQDN = %s", str);
1038                                        free(str);
1039                                }
1040                                free(symp);
1041                        }
1042                        break;
1043                }
1044                }
1045                data = ostart + d6o.d6o_len;
1046                len -= d6o.d6o_len;
1047        }
```

```
1048         if (len != 0) {
1049                 (void) strlcpy(get_line(0, 0), "Option entry truncated",
1050                     get_line_remain());
1051         }
1052 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    9701 Fri Aug 11 10:12:33 2017
new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_rpcsec.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2012 Milan Jurik. All rights reserved.
 * Copyright 2017 Gary Mills
 */

#include <sys/types.h>
#include <sys/errno.h>
#include <sys/tiuser.h>
#include <setjmp.h>

#include <rpc/types.h>
#include <rpc/xdr.h>
#include <rpc/auth.h>
#include <rpc/clnt.h>
#include <rpc/rpc_msg.h>
#include <rpc/rpcsec_gss.h>
#include <string.h>
#include "snoop.h"

extern jmp_buf xdr_err;

struct cache_struct *find_xid();
char *nameof_prog(int prog);
static void print_rpc_gss_init_arg(int, struct cache_struct *);
static void print_rpc_gss_init_res(int);

char *
rpcsec_gss_proc_to_string(unsigned int proc)
{
        switch (proc) {
        case RPCSEC_GSS_DATA:   return "RPCSEC_GSS_DATA"; break;
        case RPCSEC_GSS_INIT:   return "RPCSEC_GSS_INIT"; break;
        case RPCSEC_GSS_CONTINUE_INIT:
                                return ("RPCSEC_GSS_CONTINUE_INIT");
        case RPCSEC_GSS_DESTROY:
                                return ("RPCSEC_GSS_DESTROY");
        default:                return ("unknown");
        }
```



```
        }
}
```

*unchanged_portion_omitted_*

```
/*
 *  Skip the header RPCSEC_GSS cred data and
 *  put service and control type in the xid cache.
 */
void
extract_rpcsec_gss_cred_info(int xid)
{
        unsigned int seq_num;
        unsigned int handle_len;
        unsigned int flavor_len;
        unsigned int rpcsec_gss_ver;
        rpc_gss_service_t rpcsec_gss_service;
        unsigned int rpcsec_gss_proc;
        struct cache_struct *x;

        (void) getxdr_u_long();
        flavor_len = getxdr_u_long();
        rpcsec_gss_ver = getxdr_u_long();
        /* see if we know this version or not */
        if (rpcsec_gss_ver != 1) {
                longjmp(xdr_err, 1);
        }
        rpcsec_gss_proc   = getxdr_u_long();
        (void) getxdr_u_long();
        seq_num    = getxdr_u_long();
        rpcsec_gss_service    = getxdr_enum();
        /* skip the handle */
        xdr_skip(RNDUP(getxdr_u_long()));

        if (x = find_xid(xid)) {
                x->xid_gss_service = rpcsec_gss_service;
                x->xid_gss_proc = rpcsec_gss_proc;
        }

}

/*
 *  Print the argument data for the RPCSEC_GSS_INIT control procedure.
 */
static void
print_rpc_gss_init_arg(int flags, struct cache_struct *x)
{

        char  *line;
        char *token, *line;
        unsigned int token_len;
        int pos = 0;

        /*
         *  see if we need to print out the rpc_gss_init_arg structure
         *  or not.
         */

        if (x->xid_gss_proc != RPCSEC_GSS_INIT &&
            x->xid_gss_proc != RPCSEC_GSS_CONTINUE_INIT) {
                return;
        }

        /* print it */

        (void) sprintf(get_line(pos, getxdr_pos()),
            "RPCSEC_GSS_INIT args:");
```

```
348         pos = getxdr_pos();
349         token_len = getxdr_u_long();
350         (void) getxdr_hex(token_len);
351         token = getxdr_hex(token_len);
351         line = get_line(pos, getxdr_pos());
352         sprintf(line, "   gss_token: length = %d, data = [%d bytes]",
353             token_len, token_len);

355         show_trailer();
356 }

358 /*
359  *  Print the results data for the RPCSEC_GSS_INIT control procedure.
360  */
361 void
362 print_rpc_gss_init_res(int flags)
363 {

365         char *handle, *line;
366         char *handle, *token, *line;
366         unsigned int token_len, handle_len;
367         unsigned int major, minor, seq_window;

369         int pos = 0;
370         struct cache_struct *x;

372         /* print it */

374         (void) sprintf(get_line(pos, getxdr_pos()), "RPCSEC_GSS_INIT result:");

376         pos = getxdr_pos();
377         handle_len = getxdr_u_long();
378         handle = getxdr_hex(handle_len);
379         line = get_line(pos, getxdr_pos());
380         sprintf(line, "   handle: length = %d, data = [%s]",
381             handle_len, handle);
382         pos = getxdr_pos();
383         major = getxdr_u_long();
384         minor = getxdr_u_long();
385         seq_window = getxdr_u_long();

387         (void) sprintf(get_line(pos, getxdr_pos()),
388             "   gss_major status = %u", major);

390         (void) sprintf(get_line(pos, getxdr_pos()),
391             "   gss_minor status = %u", minor);

393         (void) sprintf(get_line(pos, getxdr_pos()),
394             "   sequence window  = %u", seq_window);
395         pos = getxdr_pos();
396         token_len = getxdr_u_long();
397         (void) getxdr_hex(token_len);
398         token = getxdr_hex(token_len);
398         line = get_line(pos, getxdr_pos());
399         sprintf(line, "   gss_token: length = %d, data = [%d bytes]",
400             token_len, token_len);
401         show_trailer();
402 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   53182 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/gss/gssd/gssdtest.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
 1 /*
 2  * CDDL HEADER START
 3  *
 4  * The contents of this file are subject to the terms of the
 5  * Common Development and Distribution License, Version 1.0 only
 6  * (the "License").  You may not use this file except in compliance
 7  * with the License.
 8  *
 9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10  * or http://www.opensolaris.org/os/licensing.
11  * See the License for the specific language governing permissions
12  * and limitations under the License.
13  *
14  * When distributing Covered Code, include this CDDL HEADER in each
15  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16  * If applicable, add the following below this CDDL HEADER, with the
17  * fields enclosed by brackets "[]" replaced with your own identifying
18  * information: Portions Copyright [yyyy] [name of copyright owner]
19  *
20  * CDDL HEADER END
21  */
22 /*
23  * Copyright 2017 Gary Mills
24  * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
25  * Use is subject to license terms.
26  */

28 /*
29  * Test client for gssd.  This program is not shipped on the binary
30  * release.
31  */

33 #include <stdio.h>
34 #include <strings.h>
35 #include <ctype.h>
36 #include <stdlib.h>
37 #include <gssapi/gssapi.h>
38 #include <gssapi/gssapi_ext.h>
39 #include "gssd.h"
40 #include <rpc/rpc.h>

42 #define _KERNEL
43 #include <gssapi/gssapi.h>
44 #undef  _KERNEL

46 int gss_major_code;
47 int gss_minor_code;

49 int init_sec_context_phase = 0;
50 int accept_sec_context_phase = 0;

52 gss_ctx_id_t    initiator_context_handle;
53 gss_ctx_id_t    acceptor_context_handle;
54 gss_cred_id_t   acceptor_credentials;
55 gss_buffer_desc init_token_buffer;
56 gss_buffer_desc accept_token_buffer;
57 gss_buffer_desc delete_token_buffer;
58 gss_buffer_desc message_buffer;
59 gss_buffer_desc msg_token;

61 #define LOOP_COUNTER  100
```



```
62 #define GSS_KRB5_MECH_OID "1.2.840.113554.1.2.2"
63 #define GSS_DUMMY_MECH_OID "1.3.6.1.4.1.42.2.26.1.2"
64 #ifdef _KERNEL
65 #define OCTAL_MACRO "%03o."
66 #define MALLOC(n) kmem_alloc((n), KM_SLEEP)
67 #define CALLOC(n, s) kmem_zalloc((n)*(s), KM_SLEEP)
68 #define FREE(x, n) kmem_free((x), (n))
69 #define memcpy(dst, src, n) bcopy((src), (dst), (n))
70 #define fprintf(s, m) printf(m)
71 #define isspace(s) ((s) == ' ' || (s) == '\t' || (s) == '\n' || \
72                 (s) == '\r' || (s) == '\v' || (s) == '\f')

74 static char *strdup(const char *s)
75 {
76         int len = strlen(s);
77         char *new = MALLOC(len+1);
78         strcpy(new, s);
79         return (new);
80 }
_____unchanged_portion_omitted_

1290 static void
1291 _gss_delete_sec_context(argc, argv)
1292 int argc;
1293 char **argv;
1294 {
1295         OM_UINT32 status;
1296         gss_ctx_id_t *context_handle;
1297         OM_uint32 minor_status;
1297         uid_t uid;

1299         uid = (uid_t) getuid();

1300         /* parse the command line to determine the variable input argument */

1302         if (argc == 0) {
1303                 usage();
1304                 return;
1305         }

1307         if (strcmp(argv[0], "initiator") == 0) {
1308                 context_handle = &initiator_context_handle;
1309         } else if (strcmp(argv[0], "acceptor") == 0) {
1310                 context_handle = &acceptor_context_handle;
1311         } else {
1312                 printf(gettext(
1313                         "must specify either \"initiator\" or \"acceptor\"\n"));
1314                 return;
1315         }

1317         argc--;
1318         argv++;

1320         if (argc != 0) {
1321                 usage();
1322                 return;
1323         }


1326         status = kgss_delete_sec_context(&minor_status,
1327                                          context_handle,
1328                                          &delete_token_buffer);


1331         /* store major and minor status for gss_display_status() call */
```

```
1333         gss_major_code = status;
1334         gss_minor_code = minor_status;

1336         if (status != GSS_S_COMPLETE) {

1338                 printf(gettext("server ret err (octal) %o (%s)\n"),
1339                         status, gettext("gss_delete_sec_context error"));
1340                 return;

1342         } else {
1343                 printf(gettext("\ndelete succeeded\n\n"));
1344                 return;
1345         }
1346 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    7994 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/hal/addons/network-devices/common.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
   1 /*
   2  * Copyright 2017 Gary Mills
   3  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
   4  *
   5  * Licensed under the Academic Free License version 2.1
   6  */

   8 #include <stdio.h>
   9 #include <stdlib.h>
  10 #include <unistd.h>
  11 #include <signal.h>
  12 #include <string.h>
  13 #include <sys/types.h>
  14 #include <sys/socket.h>
  15 #include <sys/ioctl.h>
  16 #include <sys/sockio.h>
  17 #include <net/if.h>
  18 #include <net/if_arp.h>
  19 #include <netinet/in.h>
  20 #include <arpa/inet.h>
  21 #include <netdb.h>

  23 #include <libhal.h>
  24 #include <logger.h>

  26 #include <glib.h>

  28 #include "network-discovery.h"
  29 #define NP(x)   (x?x:"NULL")

  31 extern int snmp_printer_info(char *hostname, char *community,
  32                 char **manufacturer, char **model, char **description,
  33                 char **serial_no, char ***command_set, char **uri);

  35 void
  36 network_device_name_to_udi(char *udi, size_t size, ...)
  37 {
  38         va_list ap;
  39         char *element;
  40         int i;

  42         udi[0] = '\0';
  43         va_start(ap, size);
  44         while ((element = va_arg(ap, char *)) != NULL) {
  45                 if (element[0] != '/')
  46                         strlcat(udi, "/", size);
  47                 strlcat(udi, element, size);
  48         }
  49         va_end(ap);

  51         for (i = 0; udi[i] != NULL; i++)
  52                 if (udi[i] == '.')
  53                         udi[i] = '_';
  54 }
```

_____***unchanged_portion_omitted_***

```
 114 static char *
 115 pseudo_serialno_from_addr(char *name)
 116 {
 117         int sd, errnum;
 116         int sd, rc, errnum;
```



```
 118         char buf[128];
 119         struct hostent *hp;
 120         struct xarpreq ar;

 122         if (name == NULL)
 123                 return (NULL);

 125         memset(&ar, 0, sizeof (ar));

 127         hp = getipnodebyname(name, AF_INET6, AI_ADDRCONFIG, &errnum);
 128         if (hp != NULL) {
 129                 struct sockaddr_in6 *sin6 = (struct sockaddr_in6 *)&ar.xarp_pa;

 131                 sin6->sin6_family = AF_INET6;
 132                 (void) memcpy(&sin6->sin6_addr, hp->h_addr_list[0],
 133                                 hp->h_length);
 134         } else {
 135                 struct sockaddr_in *sin = (struct sockaddr_in *)&ar.xarp_pa;

 137                 sin->sin_family = AF_INET;
 138                 sin->sin_addr.s_addr = inet_addr(name);
 139         }

 141         sd = socket(AF_INET, SOCK_DGRAM, 0);

 143         ar.xarp_ha.sdl_family = AF_LINK;
 144         (void) ioctl(sd, SIOCGXARP, (caddr_t)&ar);
 143         rc = ioctl(sd, SIOCGXARP, (caddr_t)&ar);

 146         close(sd);

 148         if (ar.xarp_flags & ATF_COM) {  /* use the MAC address */
 149                 uchar_t *ea = (uchar_t *)LLADDR(&ar.xarp_ha);

 151                 addr_to_string("LLADDR-", ea, ar.xarp_ha.sdl_alen,
 152                                         buf, sizeof (buf));

 154         } else if (hp != NULL) {          /* use the IPv6 address */
 155                 addr_to_string("IPV6ADDR-", (uchar_t *)&hp->h_addr_list[0],
 156                                         hp->h_length, buf, sizeof (buf));
 157         } else {                          /* use the IPv4 address */
 158                 struct sockaddr_in *sin = (struct sockaddr_in *)&ar.xarp_pa;

 160                 addr_to_string("IPV4ADDR-", (uchar_t *)&sin->sin_addr.s_addr, 4,
 161                                         buf, sizeof (buf));
 162         }

 164         return (strdup(buf));
 165 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   10679 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/hal/addons/storage/addon-storage.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*****************************************************************************
 *
 * addon-storage.c : watch removable media state changes
 *
 * Copyright 2017 Gary Mills
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Licensed under the Academic Free License version 2.1
 *
 *****************************************************************************/

#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <errno.h>
#include <string.h>
#include <strings.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mnttab.h>
#include <sys/dkio.h>
#include <priv.h>
#include <libsysevent.h>
#include <sys/sysevent/dev.h>

#include <libhal.h>

#include "../../hald/logger.h"

#define SLEEP_PERIOD    5

static char                     *udi;
static char                     *devfs_path;
LibHalContext                   *ctx = NULL;
static sysevent_handle_t        *shp = NULL;

static void     sysevent_dev_handler(sysevent_t *);

static void
my_dbus_error_free(DBusError *error)
{
        if (dbus_error_is_set(error)) {
                dbus_error_free(error);
        }
}
```

*unchanged_portion_omitted_*

```
int
main (int argc, char *argv[])
{
        char *device_file, *raw_device_file;
        DBusError error;
```



```
        char *bus;
        char *drive_type;
        int state, last_state;
        char *support_media_changed_str;
        int support_media_changed;
        int fd = -1;

        if ((udi = getenv ("UDI")) == NULL)
                goto out;
        if ((device_file = getenv ("HAL_PROP_BLOCK_DEVICE")) == NULL)
                goto out;
        if ((raw_device_file = getenv ("HAL_PROP_BLOCK_SOLARIS_RAW_DEVICE")) ==
                goto out;
        if ((bus = getenv ("HAL_PROP_STORAGE_BUS")) == NULL)
                goto out;
        if ((drive_type = getenv ("HAL_PROP_STORAGE_DRIVE_TYPE")) == NULL)
                goto out;
        if ((devfs_path = getenv ("HAL_PROP_SOLARIS_DEVFS_PATH")) == NULL)
                goto out;

        drop_privileges ();

        setup_logger ();

        sysevent_init ();

        support_media_changed_str = getenv ("HAL_PROP_STORAGE_CDROM_SUPPORT_MEDI
        if (support_media_changed_str != NULL && strcmp (support_media_changed_s
                support_media_changed = TRUE;
        else
                support_media_changed = FALSE;

        dbus_error_init (&error);

        if ((ctx = libhal_ctx_init_direct (&error)) == NULL) {
                goto out;
        }
        my_dbus_error_free (&error);

        if (!libhal_device_addon_is_ready (ctx, udi, &error)) {
                goto out;
        }
        my_dbus_error_free (&error);

        printf ("Doing addon-storage for %s (bus %s) (drive_type %s) (udi %s)\n"

        last_state = state = DKIO_NONE;

        /* Linux version of this addon attempts to re-open the device O_EXCL
         * every 2 seconds, trying to figure out if some other app,
         * like a cd burner, is using the device. Aside from questionable
         * value of this (apps should use HAL's locked property or/and
         * Solaris in_use facility), but also frequent opens/closes
         * keeps media constantly spun up. All this needs more thought.
         */
        for (;;) {
                if (is_mounted (device_file)) {
                        close_device (&fd);
                        sleep (SLEEP_PERIOD);
                } else if ((fd < 0) && ((fd = open (raw_device_file, O_RDONLY |
                        HAL_DEBUG (("open failed for %s: %s", raw_device_file, s
                        sleep (SLEEP_PERIOD);
                } else {
                        /* Check if a disc is in the drive */
                        /* XXX initial call always returns inserted
                         * causing unnecessary rescan - optimize?
```

```
377                          */
378                         if (ioctl (fd, DKIOCSTATE, &state) == 0) {
379                                 if (state == last_state) {
380                                         HAL_DEBUG (("state has not changed %d %s
381                                         continue;
382                                 } else {
383                                         HAL_DEBUG (("new state %d %s", state, de
384                                 }

386                                 switch (state) {
387                                 case DKIO_EJECTED:
388                                         HAL_DEBUG (("Media removal detected on %
389                                         last_state = state;

391                                         libhal_device_set_property_bool (ctx, ud
392                                         my_dbus_error_free (&error);

394                                         /* attempt to unmount all childs */
395                                         unmount_childs (ctx, udi);

397                                         /* could have a fs on the main block dev
398                                         libhal_device_rescan (ctx, udi, &error);
399                                         my_dbus_error_free (&error);
400                                         break;

402                                 case DKIO_INSERTED:
403                                         HAL_DEBUG (("Media insertion detected on
404                                         last_state = state;

406                                         libhal_device_set_property_bool (ctx, ud
407                                         my_dbus_error_free (&error);

409                                         /* could have a fs on the main block dev
410                                         libhal_device_rescan (ctx, udi, &error);
411                                         my_dbus_error_free (&error);
412                                         break;

414                                 case DKIO_DEV_GONE:
415                                         HAL_DEBUG (("Device gone detected on %s"
416                                         last_state = state;

418                                         unmount_childs (ctx, udi);
419                                         close_device (&fd);
420                                         goto out;

422                                 case DKIO_NONE:
423                                 default:
424                                         break;
425                                 }
426                         } else {
427                                 HAL_DEBUG (("DKIOCSTATE failed: %s\n", strerror(
428                                 sleep (SLEEP_PERIOD);
429                         }
430                 }
431         }

433 out:
434         sysevent_fini ();
435         if (ctx != NULL) {
436                 my_dbus_error_free (&error);
437                 libhal_ctx_shutdown (ctx, &error);
438                 libhal_ctx_free (ctx);
439         }

441         return 0;
442 }
```

```
**********************************************************
   30988 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/ipf/tools/ipfcomp.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * Copyright (C) 1993-2001 by Darren Reed.
 *
 * See the IPFILTER.LICENCE file for details on licencing.
 *
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2017 Gary Mills
 */

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

#if !defined(lint)
static const char sccsid[] = "@(#)ip_fil.c      2.41 6/5/96 (C) 1993-2000 Darren
static const char rcsid[] = "@(#)$Id: ipfcomp.c,v 1.24.2.2 2004/04/28 10:34:44 d
#endif

#include "ipf.h"


typedef struct {
        int c;
        int e;
        int n;
        int p;
        int s;
} mc_t;


static char *portcmp[] = { "*", "==", "!=", "<", ">", "<=", ">=", "**", "***" };
static int count = 0;

int intcmp __P((const void *, const void *));
static void indent __P((FILE *, int));
static void printeq __P((FILE *, char *, int, int, int));
static void printipeq __P((FILE *, char *, int, int, int));
static void addrule __P((FILE *, frentry_t *));
static void printhooks __P((FILE *, int, int, frgroup_t *));
static void emitheader __P((frgroup_t *, u_int, u_int));
static void emitGroup __P((int, int, void *, frentry_t *, char *,
                           u_int, u_int));
static void emittail __P((void));
static void printCgroup __P((int, frentry_t *, mc_t *, char *));

#define FRC_IFN 0
#define FRC_V   1
#define FRC_P   2
#define FRC_FL  3
#define FRC_TOS 4
#define FRC_TTL 5
#define FRC_SRC 6
#define FRC_DST 7
#define FRC_TCP 8
#define FRC_SP  9
#define FRC_DP  10
#define FRC_OPT 11
#define FRC_SEC 12
#define FRC_ATH 13
#define FRC_ICT 14
#define FRC_ICC 15
#define FRC_MAX 16
```



```
static  FILE    *cfile = NULL;

/*
 * This is called once per filter rule being loaded to emit data structures
 * required.
 */
void printc(fr)
frentry_t *fr;
{
        fripf_t *ipf;
        u_long *ulp;
        char *and;
        FILE *fp;
        int i;

        if (fr->fr_v != 4)
                return;
        if ((fr->fr_type != FR_T_IPF) && (fr->fr_type != FR_T_NONE))
                return;
        if ((fr->fr_type == FR_T_IPF) &&
            ((fr->fr_datype != FRI_NORMAL) || (fr->fr_satype != FRI_NORMAL)))
                return;
        ipf = fr->fr_ipf;

        if (cfile == NULL)
                cfile = fopen("ip_rules.c", "w");
        if (cfile == NULL)
                return;
        fp = cfile;
        if (count == 0) {
                fprintf(fp, "/*\n");
                fprintf(fp, "* Copyright (C) 1993-2000 by Darren Reed.\n");
                fprintf(fp, "*\n");
                fprintf(fp, "* Redistribution and use in source and binary forms
                fprintf(fp, "* provided that this notice is preserved and due cr
                fprintf(fp, "* to the original author and the contributors.\n");
                fprintf(fp, "*/\n\n");

                fprintf(fp, "#include <sys/types.h>\n");
                fprintf(fp, "#include <sys/time.h>\n");
                fprintf(fp, "#include <sys/socket.h>\n");
                fprintf(fp, "#if !defined(__FreeBSD__) && !defined(__OpenBSD__)
                fprintf(fp, "# include <sys/systm.h>\n");
                fprintf(fp, "#endif\n");
                fprintf(fp, "#include <sys/errno.h>\n");
                fprintf(fp, "#include <sys/param.h>\n");
                fprintf(fp,
"#if !defined(__SVR4) && !defined(__svr4__) && !defined(__hpux)\n");
                fprintf(fp, "# include <sys/mbuf.h>\n");
                fprintf(fp, "#endif\n");
                fprintf(fp,
"#if defined(__FreeBSD__) && (__FreeBSD_version > 220000)\n");
                fprintf(fp, "# include <sys/sockio.h>\n");
                fprintf(fp, "#else\n");
                fprintf(fp, "# include <sys/ioctl.h>\n");
                fprintf(fp, "#endif /* FreeBSD */\n");
                fprintf(fp, "#include <net/if.h>\n");
                fprintf(fp, "#include <netinet/in.h>\n");
                fprintf(fp, "#include <netinet/in_systm.h>\n");
                fprintf(fp, "#include <netinet/ip.h>\n");
                fprintf(fp, "#include <netinet/tcp.h>\n");
                fprintf(fp, "#include \"netinet/ip_compat.h\"\n");
                fprintf(fp, "#include \"netinet/ip_fil.h\"\n\n");
                fprintf(fp, "#include \"netinet/ip_rules.h\"\n\n");
```

```
124                 fprintf(fp, "#ifndef _KERNEL\n");
125                 fprintf(fp, "# include <string.h>\n");
126                 fprintf(fp, "#endif /* _KERNEL */\n");
127                 fprintf(fp, "\n");
128                 fprintf(fp, "#ifdef IPFILTER_COMPILED\n");
129         }

131         addrule(fp, fr);
132         fr->fr_type |= FR_T_BUILTIN;
133         and = "";
134         fr->fr_ref = 1;
135         i = sizeof(*fr);
136         if (i & -(1 - sizeof(*ulp)))
137                 i += sizeof(u_long);
138         for (i /= sizeof(u_long), ulp = (u_long *)fr; i > 0; i--) {
139                 fprintf(fp, "%s%#lx", and, *ulp++);
140                 and = ", ";
141         }
142         fprintf(fp, "\n};\n");
143         fr->fr_type &= ~FR_T_BUILTIN;

145         count++;

147         fflush(fp);
148 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   18007 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/ipf/tools/ippool.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * Copyright (C) 2003 by Darren Reed.
 *
 * See the IPFILTER.LICENCE file for details on licencing.
 *
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright (c) 2014, Joyent, Inc.  All rights reserved.
 * Copyright 2017 Gary Mills
 */

#include <sys/types.h>
#include <sys/time.h>
#include <sys/param.h>
#include <sys/socket.h>
#if defined(BSD) && (BSD >= 199306)
# include <sys/cdefs.h>
#endif
#include <sys/ioctl.h>

#include <net/if.h>
#if __FreeBSD_version >= 300000
# include <net/if_var.h>
#endif
#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <ctype.h>
#include <unistd.h>
#include <nlist.h>

#include "ipf.h"
#include "netinet/ipl.h"
#include "netinet/ip_lookup.h"
#include "netinet/ip_pool.h"
#include "netinet/ip_htable.h"
#include "kmem.h"
#include "ipfzone.h"

extern  int     ippool_yyparse __P((void));
extern  int     ippool_yydebug;
extern  FILE    *ippool_yyin;
extern  char    *optarg;
extern  int     lineNum;

void    showpools __P((ip_pool_stat_t *));
void    usage __P((char *));
int     main __P((int, char **));
int     poolcommand __P((int, int, char *[]));
int     poolnodecommand __P((int, int, char *[]));
int     loadpoolfile __P((int, char *[], char *));
int     poollist __P((int, char *[]));
int     poolflush __P((int, char *[]));
int     poolstats __P((int, char *[]));
```



```
int     gettype __P((char *, u_int *));
int     getrole __P((char *));
void    poollist_dead __P((int, char *, int, char *, char *));
void    showpools_live(int, int, ip_pool_stat_t *, char *, int);
void    showhashs_live(int, int, iphtstat_t *, char *, int);

int     opts = 0;
int     fd = -1;
int     use_inet6 = 0;


void usage(prog)
char *prog;
{
        const char *zoneopt = "[-G|-z zonename] ";
        fprintf(stderr, "Usage:\t%s\n", prog);
        fprintf(stderr, "\t\t\t-a [-dnv] %s[-m <name>] [-o <role>] -i <ipaddr>[/
            zoneopt);
        fprintf(stderr, "\t\t\t-A [-dnv] %s[-m <name>] [-o <role>] [-S <seed>] [
            zoneopt);
        fprintf(stderr, "\t\t\t-f <file> %s[-dnuv]\n", zoneopt);
        fprintf(stderr, "\t\t\t-F [-dv] %s[-o <role>] [-t <type>]\n", zoneopt);
        fprintf(stderr, "\t\t\t-l [-dv] %s[-m <name>] [-t <type>]\n", zoneopt);
        fprintf(stderr, "\t\t\t-r [-dnv] %s[-m <name>] [-o <role>] -i <ipaddr>[/
            zoneopt);
        fprintf(stderr, "\t\t\t-R [-dnv] %s[-m <name>] [-o <role>] [-t <type>]\n
            zoneopt);
        fprintf(stderr, "\t\t\t-s [-dtv] %s[-M <core>] [-N <namelist>]\n",
            zoneopt);
        exit(1);
}
```

_____***unchanged_portion_omitted_***

```
int poollist(argc, argv)
int argc;
char *argv[];
{
        char *kernel, *core, *poolname;
        int c, role, type, live_kernel;
        ip_pool_stat_t  plstat;
        iphtstat_t  htstat;
        ip_pool_stat_t *plstp, plstat;
        iphtstat_t *htstp, htstat;
        iphtable_t *hptr;
        iplookupop_t op;
        ip_pool_t *ptr;

        core = NULL;
        kernel = NULL;
        live_kernel = 1;
        type = IPLT_ALL;
        poolname = NULL;
        role = IPL_LOGALL;

        while ((c = getopt(argc, argv, "dG:m:M:N:o:Rt:vz:")) != -1)
                switch (c)
                {
                case 'd' :
                        opts |= OPT_DEBUG;
                        break;
                case 'G' :
                        setzonename_global(optarg);
                        break;
                case 'm' :
                        poolname = optarg;
```

```
410                         break;
411                 case 'M' :
412                         live_kernel = 0;
413                         core = optarg;
414                         break;
415                 case 'N' :
416                         live_kernel = 0;
417                         kernel = optarg;
418                         break;
419                 case 'o' :
420                         role = getrole(optarg);
421                         if (role == IPL_LOGNONE) {
422                                 fprintf(stderr, "unknown role '%s'\n", optarg);
423                                 return -1;
424                         }
425                         break;
426                 case 'R' :
427                         opts |= OPT_NORESOLVE;
428                         break;
429                 case 't' :
430                         type = gettype(optarg, NULL);
431                         if (type == IPLT_NONE) {
432                                 fprintf(stderr, "unknown type '%s'\n", optarg);
433                                 return -1;
434                         }
435                         break;
436                 case 'v' :
437                         opts |= OPT_VERBOSE;
438                         break;
439                 case 'z' :
440                         setzonename(optarg);
441                         break;
442                 }

444         if (opts & OPT_DEBUG)
445                 fprintf(stderr, "poollist: opts = %#x\n", opts);

447         if (!(opts & OPT_DONOTHING) && (fd == -1)) {
448                 fd = open(IPLOOKUP_NAME, O_RDWR);
449                 if (fd == -1) {
450                         perror("open(IPLOOKUP_NAME)");
451                         exit(1);
452                 }

454                 if (setzone(fd) != 0) {
455                         close(fd);
456                         exit(1);
457                 }
458         }

460         bzero((char *)&op, sizeof(op));
461         if (poolname != NULL) {
462                 strncpy(op.iplo_name, poolname, sizeof(op.iplo_name));
463                 op.iplo_name[sizeof(op.iplo_name) - 1] = '\0';
464         }
465         op.iplo_unit = role;

467         if (live_kernel == 0) {
468                 poollist_dead(role, poolname, type, kernel, core);
469                 return (0);
470         }

472         if (type == IPLT_ALL || type == IPLT_POOL) {
472                 plstp = &plstat;
473                 op.iplo_type = IPLT_POOL;
474                 op.iplo_size = sizeof(plstat);
```



```
475                 op.iplo_struct = &plstat;
476                 op.iplo_name[0] = '\0';
477                 op.iplo_arg = 0;

479                 if (role != IPL_LOGALL) {
480                         op.iplo_unit = role;

482                         c = ioctl(fd, SIOCLOOKUPSTAT, &op);
483                         if (c == -1) {
484                                 perror("ioctl(SIOCLOOKUPSTAT)");
485                                 return -1;
486                         }

488                         showpools_live(fd, role, &plstat, poolname, opts);
489                 } else {
490                         for (role = 0; role <= IPL_LOGMAX; role++) {
491                                 op.iplo_unit = role;

493                                 c = ioctl(fd, SIOCLOOKUPSTAT, &op);
494                                 if (c == -1) {
495                                         perror("ioctl(SIOCLOOKUPSTAT)");
496                                         return -1;
497                                 }

499                                 showpools_live(fd, role, &plstat, poolname, opts
500                         }

502                         role = IPL_LOGALL;
503                 }
504         }
505         if (type == IPLT_ALL || type == IPLT_HASH) {
506                 htstp = &htstat;
506                 op.iplo_type = IPLT_HASH;
507                 op.iplo_size = sizeof(htstat);
508                 op.iplo_struct = &htstat;
509                 op.iplo_name[0] = '\0';
510                 op.iplo_arg = 0;

512                 if (role != IPL_LOGALL) {
513                         op.iplo_unit = role;

515                         c = ioctl(fd, SIOCLOOKUPSTAT, &op);
516                         if (c == -1) {
517                                 perror("ioctl(SIOCLOOKUPSTAT)");
518                                 return -1;
519                         }
520                         showhashs_live(fd, role, &htstat, poolname, opts);
521                 } else {
522                         for (role = 0; role <= IPL_LOGMAX; role++) {

524                                 op.iplo_unit = role;
525                                 c = ioctl(fd, SIOCLOOKUPSTAT, &op);
526                                 if (c == -1) {
527                                         perror("ioctl(SIOCLOOKUPSTAT)");
528                                         return -1;
529                                 }

531                                 showhashs_live(fd, role, &htstat, poolname, opts
532                         }
533                 }
534         }
535         return 0;
536 }
```

*_____unchanged_portion_omitted_*

```
640 int poolstats(argc, argv)
```

```
641 int argc;
642 char *argv[];
643 {
644         int c, type, role;
645         int c, type, role, live_kernel;
645         ip_pool_stat_t plstat;
647         char *kernel, *core;
646         iphtstat_t htstat;
647         iplookupop_t op;

651         core = NULL;
652         kernel = NULL;
653         live_kernel = 1;
649         type = IPLT_ALL;
650         role = IPL_LOGALL;

652         bzero((char *)&op, sizeof(op));

654         while ((c = getopt(argc, argv, "dG:M:N:o:t:vz:")) != -1)
655                 switch (c)
656                 {
657                 case 'd' :
658                         opts |= OPT_DEBUG;
659                         break;
660                 case 'G' :
661                         setzonename_global(optarg);
662                         break;
663                 case 'M' :
669                         live_kernel = 0;
670                         core = optarg;
664                         break;
665                 case 'N' :
673                         live_kernel = 0;
674                         kernel = optarg;
666                         break;
667                 case 'o' :
668                         role = getrole(optarg);
669                         if (role == IPL_LOGNONE) {
670                                 fprintf(stderr, "unknown role '%s'\n", optarg);
671                                 return -1;
672                         }
673                         break;
674                 case 't' :
675                         type = gettype(optarg, NULL);
676                         if (type != IPLT_POOL) {
677                                 fprintf(stderr,
678                                         "-s not supported for this type yet\n");
679                                 return -1;
680                         }
681                         break;
682                 case 'v' :
683                         opts |= OPT_VERBOSE;
684                         break;
685                 case 'z' :
686                         setzonename(optarg);
687                         break;
688                 }

690         if (opts & OPT_DEBUG)
691                 fprintf(stderr, "poolstats: opts = %#x\n", opts);

693         if (!(opts & OPT_DONOTHING) && (fd == -1)) {
694                 fd = open(IPLOOKUP_NAME, O_RDWR);
695                 if (fd == -1) {
696                         perror("open(IPLOOKUP_NAME)");
697                         exit(1);
```



```
698                 }

700                 if (setzone(fd) != 0) {
701                         close(fd);
702                         exit(1);
703                 }
704         }

706         if (type == IPLT_ALL || type == IPLT_POOL) {
707                 op.iplo_type = IPLT_POOL;
708                 op.iplo_struct = &plstat;
709                 op.iplo_size = sizeof(plstat);
710                 if (!(opts & OPT_DONOTHING)) {
711                         c = ioctl(fd, SIOCLOOKUPSTAT, &op);
712                         if (c == -1) {
713                                 perror("ioctl(SIOCLOOKUPSTAT)");
714                                 return -1;
715                         }
716                         printf("Pools:\t%lu\n", plstat.ipls_pools);
717                         printf("Nodes:\t%lu\n", plstat.ipls_nodes);
718                 }
719         }

721         if (type == IPLT_ALL || type == IPLT_HASH) {
722                 op.iplo_type = IPLT_HASH;
723                 op.iplo_struct = &htstat;
724                 op.iplo_size = sizeof(htstat);
725                 if (!(opts & OPT_DONOTHING)) {
726                         c = ioctl(fd, SIOCLOOKUPSTAT, &op);
727                         if (c == -1) {
728                                 perror("ioctl(SIOCLOOKUPSTAT)");
729                                 return -1;
730                         }
731                         printf("Hash Tables:\t%lu\n", htstat.iphs_numtables);
732                         printf("Nodes:\t%lu\n", htstat.iphs_numnodes);
733                         printf("Out of Memory:\t%lu\n", htstat.iphs_nomem);
734                 }
735         }
736         return 0;
737 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    7480 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

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

```
  1 /*
  2  * kadmin/ldap_util/kdb5_ldap_list.c
  3  */

  5 /* Copyright (c) 2004-2005, Novell, Inc.
  6  * All rights reserved.
  7  *
  8  * Redistribution and use in source and binary forms, with or without
  9  * modification, are permitted provided that the following conditions are met:
 10  *
 11  *   * Redistributions of source code must retain the above copyright notice,
 12  *       this list of conditions and the following disclaimer.
 13  *   * Redistributions in binary form must reproduce the above copyright
 14  *       notice, this list of conditions and the following disclaimer in the
 15  *       documentation and/or other materials provided with the distribution.
 16  *   * The copyright holder's name is not used to endorse or promote products
 17  *       derived from this software without specific prior written permission.
 18  *
 19  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 20  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 21  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 22  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 23  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 24  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 25  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 26  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 27  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 28  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 29  * POSSIBILITY OF SUCH DAMAGE.
 30  */

 32 /*
 33  * Miscellaneous functions for managing the string and integer lists
 34  */

 36 #include <k5-int.h>
 37 #include "kdb5_ldap_list.h"

 39 /*
 40  * Counts the number of entries in the given array of strings
 41  */
 42 int list_count_str_array(char **list)
 43 {
 44     int i = 0;

 46     if (list == NULL)
 47         return 0;

 49     for (i = 0; *list != NULL; list++) {
 50         i++;
 51     }

 53     return i;
 54 }
```

```
______unchanged_portion_omitted_
```

```
152 /*
153  * Modifies the destination list to contain or not to contain the
```



```
154  * entries present in the source list, depending on the mode
155  * (ADD or DELETE).
156  */
157 void list_modify_str_array(destlist, sourcelist, mode)
158     char ***destlist;
159     const char **sourcelist;
160     int mode;
161 {
162     char **dlist = NULL, **tmplist = NULL;
163     const char **slist = NULL;
164     int dcount = 0, scount = 0, copycount = 0;
167     int found = 0;

166     if ((destlist == NULL) || (*destlist == NULL) || (sourcelist == NULL))
167         return;

169     /* We need to add every entry present in the source list to
170      * the destination list */
171     if (mode == LIST_MODE_ADD) {
172         /* Traverse throught the end of destlist for appending */
173         for (dlist = *destlist, dcount = 0; *dlist != NULL;
174              dlist++, dcount++) {
175             ;   /* NULL statement */
176         }
177         /* Count the number of entries in the source list */
178         for (slist = sourcelist, scount = 0; *slist != NULL;
179              slist++, scount++) {
180             ;   /* NULL statement */
181         }
182         /* Reset the slist pointer to the start of source list */
183         slist = sourcelist;

185         /* Now append the source list to the existing destlist */
186         if ((dcount + scount) < MAX_LIST_ENTRIES)
187             copycount = scount;
188         else
189             /* Leave the last entry for list terminator(=NULL) */
190             copycount = (MAX_LIST_ENTRIES -1) - dcount;

192         memcpy(dlist, slist, (sizeof(char *) * copycount));
193         dlist += copycount;
194         *dlist = NULL;
195     } else if (mode == LIST_MODE_DELETE) {
196         /* We need to delete every entry present in the source list
197          * from the destination list */
198         for (slist = sourcelist; *slist != NULL; slist++) {
199             for (dlist = *destlist; *dlist != NULL; dlist++) {
203                 found = 0; /* value not found */
200                 /* DN is case insensitive string */
201                 if (strcasecmp(*dlist, *slist) == 0) {
206                     found = 1;
202                     free(*dlist);
203                     /* Advance the rest of the entries by one */
204                     for (tmplist = dlist; *tmplist != NULL; tmplist++) {
205                         *tmplist = *(tmplist+1);
206                     }
207                     break;
208                 }
209             }
210         }
211     }

213     return;
214 }
```

```
______unchanged_portion_omitted_
```

```
**********************************************************
   80652 Fri Aug 11 10:12:34 2017
new/usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
   1 /*
   2  * Copyright 2017 Gary Mills
   3  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
   4  * Use is subject to license terms.
   5  */

   7 /*
   8  * kadmin/ldap_util/kdb5_ldap_realm.c
   9  *
  10  * Copyright 1990,1991,2001, 2002 by the Massachusetts Institute of Technology.
  11  * All Rights Reserved.
  12  *
  13  * Export of this software from the United States of America may
  14  *   require a specific license from the United States Government.
  15  *   It is the responsibility of any person or organization contemplating
  16  *   export to obtain such a license before exporting.
  17  *
  18  * WITHIN THAT CONSTRAINT, permission to use, copy, modify, and
  19  * distribute this software and its documentation for any purpose and
  20  * without fee is hereby granted, provided that the above copyright
  21  * notice appear in all copies and that both that copyright notice and
  22  * this permission notice appear in supporting documentation, and that
  23  * the name of M.I.T. not be used in advertising or publicity pertaining
  24  * to distribution of the software without specific, written prior
  25  * permission.  Furthermore if you modify this software you must label
  26  * your software as modified software and not distribute it in such a
  27  * fashion that it might be confused with the original M.I.T. software.
  28  * M.I.T. makes no representations about the suitability of
  29  * this software for any purpose.  It is provided "as is" without express
  30  * or implied warranty.
  31  */

  33 /*
  34  * Copyright (C) 1998 by the FundsXpress, INC.
  35  *
  36  * All rights reserved.
  37  *
  38  * Export of this software from the United States of America may require
  39  * a specific license from the United States Government.  It is the
  40  * responsibility of any person or organization contemplating export to
  41  * obtain such a license before exporting.
  42  *
  43  * WITHIN THAT CONSTRAINT, permission to use, copy, modify, and
  44  * distribute this software and its documentation for any purpose and
  45  * without fee is hereby granted, provided that the above copyright
  46  * notice appear in all copies and that both that copyright notice and
  47  * this permission notice appear in supporting documentation, and that
  48  * the name of FundsXpress. not be used in advertising or publicity pertaining
  49  * to distribution of the software without specific, written prior
  50  * permission.  FundsXpress makes no representations about the suitability of
  51  * this software for any purpose.  It is provided "as is" without express
  52  * or implied warranty.
  53  *
  54  * THIS SOFTWARE IS PROVIDED ''AS IS'' AND WITHOUT ANY EXPRESS OR
  55  * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
  56  * WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.
  57  */

  59 /* Copyright (c) 2004-2005, Novell, Inc.
  60  * All rights reserved.
  61  *
```



```
  62  * Redistribution and use in source and binary forms, with or without
  63  * modification, are permitted provided that the following conditions are met:
  64  *
  65  *   * Redistributions of source code must retain the above copyright notice,
  66  *       this list of conditions and the following disclaimer.
  67  *   * Redistributions in binary form must reproduce the above copyright
  68  *       notice, this list of conditions and the following disclaimer in the
  69  *       documentation and/or other materials provided with the distribution.
  70  *   * The copyright holder's name is not used to endorse or promote products
  71  *       derived from this software without specific prior written permission.
  72  *
  73  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  74  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  75  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  76  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
  77  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
  78  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
  79  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
  80  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
  81  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  82  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  83  * POSSIBILITY OF SUCH DAMAGE.
  84  */

  86 /*
  87  * Create / Modify / Destroy / View / List realm(s)
  88  */

  90 /* Needed for getting the definition of KRB5_TL_DB_ARGS */
  91 #define SECURID

  93 #include <stdio.h>
  94 #include <k5-int.h>
  95 #include <kadm5/admin.h>
  96 #include <libintl.h>
  97 #include <locale.h>
  98 #include "kdb5_ldap_util.h"
  99 #include "kdb5_ldap_list.h"
 100 #include <ldap_principal.h>
 101 #include <ldap_krbcontainer.h>
 102 extern time_t get_date(char *); /* kadmin/cli/getdate.o */

 104 char *yes = "yes\n"; /* \n to compare against result of fgets */
 105 krb5_key_salt_tuple def_kslist = {ENCTYPE_DES_CBC_CRC, KRB5_KDB_SALTTYPE_NORMAL}

 107 struct realm_info rblock = {
 108     KRB5_KDB_MAX_LIFE,
 109     KRB5_KDB_MAX_RLIFE,
 110     KRB5_KDB_EXPIRATION,
 111     KRB5_KDB_DEF_FLAGS,
 112     (krb5_keyblock *) NULL,
 113     1,
 114     &def_kslist
 115 };
______unchanged_portion_omitted_

 133 extern char *mkey_password;
 134 extern char *progname;
 135 extern kadm5_config_params global_params;

 137 static void print_realm_params(krb5_ldap_realm_params *rparams, int mask);
 138 static int kdb_ldap_create_principal (krb5_context context, krb5_principal
 139                                       princ, enum ap_op op, struct realm_info *p


 142 static char *strdur(time_t duration);
```

```
143 static int get_ticket_policy(krb5_ldap_realm_params *rparams, int *i, char *argv
144 static krb5_error_code krb5_dbe_update_mod_princ_data_new (krb5_context context,
145 static krb5_error_code krb5_dbe_update_tl_data_new ( krb5_context context, krb5_

147 #define ADMIN_LIFETIME 60*60*3 /* 3 hours */
148 #define CHANGEPW_LIFETIME 60*5 /* 5 minutes */

150 static int get_ticket_policy(rparams,i,argv,argc)
151     krb5_ldap_realm_params *rparams;
152     int *i;
153     char *argv[];
154     int argc;
155 {
156     time_t date;
157     time_t now;
158     int mask = 0;
159     krb5_error_code retval = 0;
159     krb5_boolean no_msg = FALSE;

161     krb5_boolean print_usage = FALSE;
161     /* Solaris Kerberos */
162     char *me = progname;

164     time(&now);
165     if (!strcmp(argv[*i], "-maxtktlife")) {
166         if (++(*i) > argc-1)
167             goto err_nomsg;
168             goto err_usage;
168         date = get_date(argv[*i]);
169         if (date == (time_t)(-1)) {
170             retval = EINVAL;
171             com_err (me, retval, gettext("while providing time specification"));
172             goto err_nomsg;
173         }
174         rparams->max_life = date-now;
175         mask |= LDAP_REALM_MAXTICKETLIFE;
176     }


179     else if (!strcmp(argv[*i], "-maxrenewlife")) {
180         if (++(*i) > argc-1)
181             goto err_nomsg;
182             goto err_usage;

183         date = get_date(argv[*i]);
184         if (date == (time_t)(-1)) {
185             retval = EINVAL;
186             com_err (me, retval, gettext("while providing time specification"));
187             goto err_nomsg;
188         }
189         rparams->max_renewable_life = date-now;
190         mask |= LDAP_REALM_MAXRENEWLIFE;
191     } else if (!strcmp((argv[*i] + 1), "allow_postdated")) {
192         if (*(argv[*i]) == '+')
193             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_POSTDATED);
194         else if (*(argv[*i]) == '-')
195             rparams->tktflags |= KRB5_KDB_DISALLOW_POSTDATED;
196         else
197             goto err_nomsg;
198             goto err_usage;

199         mask |= LDAP_REALM_KRBTICKETFLAGS;
200     } else if (!strcmp((argv[*i] + 1), "allow_forwardable")) {
201         if (*(argv[*i]) == '+')
202             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_FORWARDABLE);
```



```
204         else if (*(argv[*i]) == '-')
205             rparams->tktflags |= KRB5_KDB_DISALLOW_FORWARDABLE;
206         else
207             goto err_nomsg;
208             goto err_usage;

209         mask |= LDAP_REALM_KRBTICKETFLAGS;
210     } else if (!strcmp((argv[*i] + 1), "allow_renewable")) {
211         if (*(argv[*i]) == '+')
212             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_RENEWABLE);
213         else if (*(argv[*i]) == '-')
214             rparams->tktflags |= KRB5_KDB_DISALLOW_RENEWABLE;
215         else
216             goto err_nomsg;
217             goto err_usage;

218         mask |= LDAP_REALM_KRBTICKETFLAGS;
219     } else if (!strcmp((argv[*i] + 1), "allow_proxiable")) {
220         if (*(argv[*i]) == '+')
221             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_PROXIABLE);
222         else if (*(argv[*i]) == '-')
223             rparams->tktflags |= KRB5_KDB_DISALLOW_PROXIABLE;
224         else
225             goto err_nomsg;
226             goto err_usage;

227         mask |= LDAP_REALM_KRBTICKETFLAGS;
228     } else if (!strcmp((argv[*i] + 1), "allow_dup_skey")) {
229         if (*(argv[*i]) == '+')
230             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_DUP_SKEY);
231         else if (*(argv[*i]) == '-')
232             rparams->tktflags |= KRB5_KDB_DISALLOW_DUP_SKEY;
233         else
234             goto err_nomsg;
235             goto err_usage;

236         mask |= LDAP_REALM_KRBTICKETFLAGS;
237     }

239     else if (!strcmp((argv[*i] + 1), "requires_preauth")) {
240         if (*(argv[*i]) == '+')
241             rparams->tktflags |= KRB5_KDB_REQUIRES_PRE_AUTH;
242         else if (*(argv[*i]) == '-')
243             rparams->tktflags &= (int)(~KRB5_KDB_REQUIRES_PRE_AUTH);
244         else
245             goto err_nomsg;
246             goto err_usage;

247         mask |= LDAP_REALM_KRBTICKETFLAGS;
248     } else if (!strcmp((argv[*i] + 1), "requires_hwauth")) {
249         if (*(argv[*i]) == '+')
250             rparams->tktflags |= KRB5_KDB_REQUIRES_HW_AUTH;
251         else if (*(argv[*i]) == '-')
252             rparams->tktflags &= (int)(~KRB5_KDB_REQUIRES_HW_AUTH);
253         else
254             goto err_nomsg;
255             goto err_usage;

256         mask |= LDAP_REALM_KRBTICKETFLAGS;
257     } else if (!strcmp((argv[*i] + 1), "allow_svr")) {
258         if (*(argv[*i]) == '+')
259             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_SVR);
260         else if (*(argv[*i]) == '-')
261             rparams->tktflags |= KRB5_KDB_DISALLOW_SVR;
262         else
263             goto err_nomsg;
```

```
264             goto err_usage;

265         mask |= LDAP_REALM_KRBTICKETFLAGS;
266     } else if (!strcmp((argv[*i] + 1), "allow_tgs_req")) {
267         if (*(argv[*i]) == '+')
268             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_TGT_BASED);
269         else if (*(argv[*i]) == '-')
270             rparams->tktflags |= KRB5_KDB_DISALLOW_TGT_BASED;
271         else
272             goto err_nomsg;
273             goto err_usage;

274         mask |= LDAP_REALM_KRBTICKETFLAGS;
275     } else if (!strcmp((argv[*i] + 1), "allow_tix")) {
276         if (*(argv[*i]) == '+')
277             rparams->tktflags &= (int)(~KRB5_KDB_DISALLOW_ALL_TIX);
278         else if (*(argv[*i]) == '-')
279             rparams->tktflags |= KRB5_KDB_DISALLOW_ALL_TIX;
280         else
281             goto err_nomsg;
282             goto err_usage;

283         mask |= LDAP_REALM_KRBTICKETFLAGS;
284     } else if (!strcmp((argv[*i] + 1), "needchange")) {
285         if (*(argv[*i]) == '+')
286             rparams->tktflags |= KRB5_KDB_REQUIRES_PWCHANGE;
287         else if (*(argv[*i]) == '-')
288             rparams->tktflags &= (int)(~KRB5_KDB_REQUIRES_PWCHANGE);
289         else
290             goto err_nomsg;
291             goto err_usage;

292         mask |= LDAP_REALM_KRBTICKETFLAGS;
293     } else if (!strcmp((argv[*i] + 1), "password_changing_service")) {
294         if (*(argv[*i]) == '+')
295             rparams->tktflags |= KRB5_KDB_PWCHANGE_SERVICE;
296         else if (*(argv[*i]) == '-')
297             rparams->tktflags &= (int)(~KRB5_KDB_PWCHANGE_SERVICE);
298         else
299             goto err_nomsg;
300             goto err_usage;

301         mask |=LDAP_REALM_KRBTICKETFLAGS;
302     }
304 err_usage:
305     print_usage = TRUE;

304 err_nomsg:
308     no_msg = TRUE;

306     return mask;
307 }
______unchanged_portion_omitted_
```

****
13040 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/ldap/common/fileurl.c
8485 Remove set but unused variables in usr/src/cmd
****

```
/*
 * Copyright 2017 Gary Mills
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * The contents of this file are subject to the Netscape Public
 * License Version 1.1 (the "License"); you may not use this file
 * except in compliance with the License. You may obtain a copy of
 * the License at http://www.mozilla.org/NPL/
 *
 * Software distributed under the License is distributed on an "AS
 * IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or
 * implied. See the License for the specific language governing
 * rights and limitations under the License.
 *
 * The Original Code is Mozilla Communicator client code, released
 * March 31, 1998.
 *
 * The Initial Developer of the Original Code is Netscape
 * Communications Corporation. Portions created by Netscape are
 * Copyright (C) 1998-1999 Netscape Communications Corporation. All
 * Rights Reserved.
 *
 * Contributor(s):
 */

/*
 *  LDAP tools fileurl.c -- functions for handling file URLs.
 *  Used by ldapmodify.
 */

#include "ldaptool.h"
#include "fileurl.h"
#include <ctype.h>      /* for isalpha() */
#ifdef SOLARIS_LDAP_CMD
#include <locale.h>
#endif  /* SOLARIS_LDAP_CMD */

#ifndef SOLARIS_LDAP_CMD
#define gettext(s) s
#endif

static int str_starts_with( const char *s, char *prefix );
static void hex_unescape( char *s );
static int unhex( char c );
static void strcpy_escaped_and_convert( char *s1, char *s2 );
static int berval_from_file( const char *path, struct berval *bvp,
        int reporterrs );

/*
 * Convert a file URL to a local path.
 *
 * If successful, LDAPTOOL_FILEURL_SUCCESS is returned and *localpathp is
 * set point to an allocated string.  If not, an different LDAPTOOL_FILEURL_
 * error code is returned.
 *
 * See RFCs 1738 and 2396 for a specification for file URLs... but
 * Netscape Navigator seems to be a bit more lenient in what it will
 * accept, especially on Windows).
```



```
 *
 * This function parses file URLs of these three forms:
 *
 *    file:///path
 *    file:/path
 *    file://localhost/path
 *    file://host/path          (rejected with a ...NONLOCAL error)
 *
 * On Windows, we convert leading drive letters of the form C| to C:
 * and if a drive letter is present we strip off the slash that precedes
 * path.  Otherwise, the leading slash is returned.
 *
 */
int
ldaptool_fileurl2path( const char *fileurl, char **localpathp )
{
    const char  *path;
    char        *pathcopy;

    /*
     * Make sure this is a file URL we can handle.
     */
    if ( !str_starts_with( fileurl, "file:" )) {
        return( LDAPTOOL_FILEURL_NOTAFILEURL );
    }

    path = fileurl + 5;         /* skip past "file:" scheme prefix */

    if ( *path != '/' ) {
        return( LDAPTOOL_FILEURL_MISSINGPATH );
    }

    ++path;                     /* skip past '/' at end of "file:/" */

    if ( *path == '/' ) {
        ++path;                 /* remainder is now host/path or /path */
        if ( *path != '/' ) {
            /*
             * Make sure it is for the local host.
             */
            if ( str_starts_with( path, "localhost/" )) {
                path += 9;
            } else {
                return( LDAPTOOL_FILEURL_NONLOCAL );
            }
        }
    } else {            /* URL is of the form file:/path */
        --path;
    }

    /*
     * The remainder is now of the form /path.  On Windows, skip past the
     * leading slash if a drive letter is present.
     */
#ifdef _WINDOWS
    if ( isalpha( path[1] ) && ( path[2] == '|' || path[2] == ':' )) {
        ++path;
    }
#endif /* _WINDOWS */

    /*
     * Duplicate the path so we can safely alter it.
     * Unescape any %HH sequences.
     */
    if (( pathcopy = strdup( path )) == NULL ) {
        return( LDAPTOOL_FILEURL_NOMEMORY );
```

```
128     }
129     hex_unescape( pathcopy );

131 #ifdef _WINDOWS
132     /*
133      * Convert forward slashes to backslashes for Windows.  Also,
134      * if we see a drive letter / vertical bar combination (e.g., c|)
135      * at the beginning of the path, replace the '|' with a ':'.
136      */
137     {
138         char    *p;

140         for ( p = pathcopy; *p != '\0'; ++p ) {
141             if ( *p == '/' ) {
142                 *p = '\\';
143             }
144         }
145     }

147     if ( isalpha( pathcopy[0] ) && pathcopy[1] == '|' ) {
148         pathcopy[1] = ':';
149     }
150 #endif /* _WINDOWS */

152     *localpathp = pathcopy;
153     return( LDAPTOOL_FILEURL_SUCCESS );
154 }
```

*unchanged_portion_omitted_*

```
348 /*
349  * Populate *bvp with the contents of the file named by "path".
350  *
351  * If reporterrs is non-zero, specific error messages are printed to
352  * stderr.
353  *
354  * If successful, LDAPTOOL_FILEURL_SUCCESS is returned and bvp->bv_len
355  * and bvp->bv_val are set (the latter is set to malloc'd memory).
356  * Upon failure, a different LDAPTOOL_FILEURL_ error code is returned.
357  */

359 static int
360 berval_from_file( const char *path, struct berval *bvp, int reporterrs )
361 {
362     FILE        *fp;
363     long        rlen;
363     int         eof;
364 #if defined( XP_WIN32 )
365     char        mode[20] = "r+b";
366 #else
367     char        mode[20] = "r";
368 #endif

370 #ifdef SOLARIS_LDAP_CMD
371     if (( fp = fopen( path, mode )) == NULL ) {
372 #else
373     if (( fp = ldaptool_open_file( path, mode )) == NULL ) {
374 #endif  /* SOLARIS_LDAP_CMD */
375         if ( reporterrs ) perror( path );
376         return( LDAPTOOL_FILEURL_FILEIOERROR );
377     }

379     if ( fseek( fp, 0L, SEEK_END ) != 0 ) {
380         if ( reporterrs ) perror( path );
381         fclose( fp );
382         return( LDAPTOOL_FILEURL_FILEIOERROR );
```



```
383     }

385     bvp->bv_len = ftell( fp );

387     if (( bvp->bv_val = (char *)malloc( bvp->bv_len + 1 )) == NULL ) {
388         if ( reporterrs ) perror( "malloc" );
389         fclose( fp );
390         return( LDAPTOOL_FILEURL_NOMEMORY );
391     }

393     if ( fseek( fp, 0L, SEEK_SET ) != 0 ) {
394         if ( reporterrs ) perror( path );
395         fclose( fp );
396         return( LDAPTOOL_FILEURL_FILEIOERROR );
397     }

399     rlen = fread( bvp->bv_val, 1, bvp->bv_len, fp );
400     eof = feof( fp );
400     fclose( fp );

402     if ( rlen != (long)bvp->bv_len ) {
403         if ( reporterrs ) perror( path );
404         free( bvp->bv_val );
405         return( LDAPTOOL_FILEURL_FILEIOERROR );
406     }

408     bvp->bv_val[ bvp->bv_len ] = '\0';
409     return( LDAPTOOL_FILEURL_SUCCESS );
410 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    4114 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/lp/lib/papi/ppd.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * This file contains an extremely rudimentary implementation of PPD file
 * parsing support.  The parsing done here converts the contents of a PPD
 * file into a set of PAPI attributes that applications can use to build
 * print panels.
 */

#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <papi.h>

static void
process_line(char *line, char **key, char **value, char **comment)
{
        char *ptr, *ptr2;

        *key = &line[1];
        *value = NULL;
        *comment = NULL;

        if ((ptr = strchr(line, ':')) == NULL)
                return;

        /*
         * line is in the form:
         *    *key: value/comment
         * or
         *    *key value/comment: data
         */
        *ptr++ = NULL;
        while (isspace(*ptr) != 0)
                ptr++;
```



```
        if ((ptr2 = strchr(line, ' ')) != NULL) {
                ptr = ptr2;
                /*
                 * line is in the form:
                 *    *key value/comment: data
                 */
                *ptr++ = NULL;
                while (*ptr == ' ')
                        ptr++;
        }

        if (*ptr == '*')
                ptr++;

        *value = ptr;

        if ((ptr = strchr(ptr, '/')) != NULL) {
                *ptr++ = NULL;
                *comment = ptr;
        }
}

papi_status_t
PPDFileToAttributesList(papi_attribute_t ***attributes, char *filename)
{
        papi_status_t status = PAPI_OK;
        FILE *fp;
        char line[256];
        char capability[256];
        char def[256];
        char supported[256];
        char *current_group_name = NULL;

        int ui = 0;

        if ((fp = fopen(filename, "r")) == NULL)
                return (PAPI_NOT_POSSIBLE);

        while ((status == PAPI_OK) &&
                        (fgets(line, sizeof (line), fp) != NULL)) {
                char *key = NULL, *value = NULL, *text = NULL;

                /* we want *key...: "value" */
                if (line[0] != '*')
                        continue;

                if (strchr(line, ':') == NULL)
                        continue;

                if ((text = strrchr(line, '\n')) != NULL)
                        *text = NULL;

                process_line(line, &key, &value, &text);

                if ((strcasecmp(key, "PageSize") == 0) ||
                    (strcasecmp(key, "InputSlot") == 0))
                        key = "media";

                if (strcasecmp(key, "OpenGroup") == 0) {
                        if (value == NULL)
                                value = "unknown";
                        current_group_name = strdup(value);
                } else if (strcasecmp(key, "OpenUI") == 0) {
                        if ((strcasecmp(value, "PageSize") == 0) ||
                            (strcasecmp(value, "InputSlot") == 0))
```

```
124                                 value = "media";
125                         snprintf(capability, sizeof (capability), "%s", value);
126                         snprintf(def, sizeof (def),
127                                 "%s-default", value);
128                         snprintf(supported, sizeof (supported),
129                                 "%s-supported", value);
130                         ui = 1;
131                 } else if (strcasecmp(key, "CloseGroup") == 0) {
132                         /* do nothing */
133                 } else if (strcasecmp(key, "CloseUI") == 0) {
134                         ui = 0;
135                         /* do nothing */
136                 } else if (strcasecmp(key, "Manufacturer") == 0) {
137                         status = papiAttributeListAddString(attributes,
138                                 PAPI_ATTR_EXCL,
139                                 "printer-make", value);
140                 } else if (strcasecmp(key, "ModelName") == 0) {
141                         status = papiAttributeListAddString(attributes,
142                                 PAPI_ATTR_EXCL,
143                                 "printer-model", value);
144                 } else if (strcasecmp(key, "ShortNickName") == 0) {
145                         status = papiAttributeListAddString(attributes,
146                                 PAPI_ATTR_EXCL,
147                                 "printer-make-and-model", value);
148                 } else if ((strncasecmp(key, "Default", 7) == 0) && ui) {
149                         status = papiAttributeListAddString(attributes,
150                                 PAPI_ATTR_EXCL,
151                                 def, value);
152                 } else if ((strcasecmp(key, capability) == 0) && ui) {
153                         status = papiAttributeListAddString(attributes,
154                                 PAPI_ATTR_APPEND,
155                                 supported, value);
156                 }
157         }
158         fclose(fp);

160         return (status);
161 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    9763 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/mail/copylet.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

#include "mail.h"

/*
    NAME
        copylet - copy a given letter to a file pointer

    SYNOPSIS
        int copylet(int letnum, FILE *f, int type)

    DESCRIPTION
        Copylet() will copy the letter "letnum" to the
        given file pointer.

                letnum  -> index into: letter table
                f       -> file pointer to copy file to
                type    -> copy type

        Returns TRUE on a completely successful copy.
*/

int
copylet(int letnum, FILE *f, int type)
{
        int             pos = ftell(f);
        int             rc  = xxxcopylet(letnum, f, type);

        if (fflush(f) != 0)
                rc = FALSE;

```



```
        /*
         * On error, truncate the file to its original position so that a
         * partial message is not left in the mailbox.
         */
        if (rc == FALSE)
                ftruncate(fileno(f), pos);

        return(rc);
}

int
xxxcopylet(int letnum, FILE *f, int type)
{
        static char     pn[] = "copylet";
        char    buf[LSIZE], lastc;
        char    wbuf[LSIZE];
        int     n;
        long    i, k;
        int     num;
        int     rtrncont = 1;   /* True: nondelivery&content included, or regula
        int     suppress = FALSE;
        int     sav_suppress = FALSE; /* Did we suppress previous hdr line? */
        int     print_from_struct = FALSE; /* print from hdrlines struct */
                                           /* rather than fgets() buffer */
        int     pushrest = FALSE;
        int     ctf = FALSE;
        int     didafflines = FALSE;    /* Did we already put out any */
                                        /* H_AFWDFROM lines? */
        int     didrcvlines = FALSE;    /* Did we already put out any */
                                        /* H_RECEIVED lines? */
        long    clen = -1L;
        int     htype;                  /* header type */
        int     sav_htype;      /* Header type of last non-H_CONT header line */
        struct hdrs *hptr;

        if (!sending) {
                /* Clear out any saved header info from previous message */
                clr_hinfo();
        }

        fseek(tmpf, let[letnum].adr, 0);
        /* Get size of message as stored into tempfile by copymt() */
        k = let[letnum+1].adr - let[letnum].adr;
        Dout(pn, 1, "(letnum = %d, type = %d), k = %ld\n", letnum, type, k);
        while (k>0) {   /* process header */
                num = ((k < sizeof(buf)) ? k+1 : sizeof(buf));
                if (fgets (buf, num, tmpf) == NULL) {
                        return (FALSE);
                }
                if ((n = strlen (buf)) == 0) {
                        k = 0;
                        break;
                }
                k -= n;
                lastc = buf[n-1];
                if (pushrest) {
                        pushrest = (lastc != '\n');
                        continue;
                }
                htype = isheader (buf, &ctf);
                Dout(pn, 5, "loop 1: buf = %s, htype= %d/%s\n", buf, htype, head
                if (htype == H_CLEN) {
                        if (!sending) {
                                savehdrs(buf,htype);
                        }
                        if ((hptr = hdrlines[H_CLEN].head) !=
```

```
125                             (struct hdrs *)NULL) {
126                                 clen = atol (hptr->value);
127                         }
128                 }
129                 if (type == ZAP) {
130                         if (htype != FALSE) {
131                                 pushrest = (lastc != '\n');
132                                 continue;
133                         }
134                         /* end of header.  Print non-blank line and bail. */
135                         Dout(pn, 5, "ZAP end header; n=%d, buf[0] = %d\n", n, bu
136                         if (buf[0] != '\n') {
137                                 if (fwrite(buf,1,n,f) != n) {
138                                         sav_errno = errno;
139                                         return(FALSE);
140                                 }
141                         } else {
142                                 n = 0;
143                         }
144                         break;
145                 }
146                 /* Copy From line appropriately */
147                 if (fwrite(buf,1,n-1,f) != n-1)  {
148                         sav_errno = errno;
149                         return(FALSE);
150                 }
151                 if (lastc != '\n') {
152                         if (fwrite(&lastc,1,1,f) != 1) {
153                                 sav_errno = errno;
154                                 return(FALSE);
155                         }
156                         continue;
157                 }
158                 switch(type) {
159                         case REMOTE:
160                                 if (fprintf(f, rmtmsg, thissys) < 0)
161                                 {
162                                         sav_errno = errno;
163                                         return(FALSE);
164                                 }
165
166                                 break;

168                         case TTY:
169                         case ORDINARY:
170                         default:
171                                 if (fprintf(f, "\n") < 0)
172                                 {
173                                         sav_errno = errno;
174                                         return(FALSE);
175                                 }
176                                 break;
177                 }
178                 if ((error > 0) && (dflag == 1)) {
179                         Dout(pn, 3, "before gendeliv(), uval = '%s'\n", uval);
180                         gendeliv(f, dflag, uval);
181                         if (!(ckdlivopts(H_TCOPY, (int*)0) & RETURN)) {
182                                 rtrncont = 0;
183                         } else {
184                                 /* Account for content-type info */
185                                 /* of returned msg */
186                                 if (fprintf(f, "%s %s\n", header[H_CTYPE].tag,
187                                     (let[letnum].text == TRUE ? "text/plain" : "
188                                 {
189                                         sav_errno = errno;
190                                         return(FALSE);
```



```
191                                 }

193                                 /* Compute Content-Length of what's being */
194                                 /* returned... */
195                                 i = k;
196                                 /* Account for H_AFWDFROM, H_AFWDCNT, */
197                                 /* H_TCOPY, or H_RECEIVED lines which may */
198                                 /* be added later */
199                                 if (affcnt > 0) {
200                                         sprintf(wbuf, "%d", affcnt);
201                                         i += (affbytecnt
202                                                 + strlen(header[H_AFWDCNT].tag)
203                                                 + strlen(wbuf) + 2);
204                                 }
205                                 if (orig_tcopy) {
206                                     if ((hptr = hdrlines[H_TCOPY].head) !=
207                                                                 (struct hdrs *)NULL) {
208                                         i +=
209                                           strlen(hdrlines[H_TCOPY].head->value);
210                                     }
211                                 }
212                                 if ((hptr = hdrlines[H_RECEIVED].head) !=
213                                                                 (struct hdrs *)NULL) {
214                                     i += rcvbytecnt;
215                                 }
216                                 /* Add in strlen of MIME-Version:, */
217                                 /* Content-Length: and Content-Type: */
218                                 /* values for msg being returned... */
219                                 if ((hptr = hdrlines[H_MIMEVERS].head) !=
220                                                                 (struct hdrs *)NULL) {
221                                     i += strlen(hdrlines[H_MIMEVERS].head->value
222                                 }
223                                 if ((hptr = hdrlines[H_CTYPE].head) !=
224                                                                 (struct hdrs *)NULL) {
225                                     i += strlen(hdrlines[H_CTYPE].head->value);
226                                 }
227                                 if ((hptr = hdrlines[H_CLEN].head) !=
228                                                                 (struct hdrs *)NULL) {
229                                     i += strlen(hdrlines[H_CLEN].head->value);
230                                 }
231                                 if (fprintf(f, "%s %ld\n", header[H_CLEN].tag, i
232                                 {
233                                         sav_errno = errno;
234                                         return(FALSE);
235                                 }
236                         }
237                         if (fprintf(f, "\n") < 0)
238                         {
239                                 sav_errno = errno;
240                                 return(FALSE);
241                         }
242                 }
243                 if (fflush(f))
244                 {
245                         sav_errno = errno;
246                         return(FALSE);
247                 }
248
249                 break;
250         }
251         /* if not ZAP, copy balance of header */
252         n = 0;
253         if ((type != ZAP) && rtrncont)
254                 while (k>0 || n>0) {
255                         if ((n > 0) && !suppress) {
256                                 if (print_from_struct == TRUE) {
```

```
257                                     if (printhdr (type, htype, hptr, f) < 0)
258                                             return (FALSE);
259                                     }
260                             } else {
261                                 if (sel_disp(type, htype, buf) >= 0) {
262                                     if (fwrite(buf,1,n,f) != n)  {
263                                             sav_errno = errno;
264                                             return(FALSE);
265                                     }
266                                 }
267                             }
268                             if (htype == H_DATE) {
269                                     dumprcv(type, htype,&didrcvlines,&suppre
270                                     dumpaff(type, htype,&didafflines,&suppre
271                             }
272                     }
273                     if (k <= 0) {
274                             /* Can only get here if k=0 && n>0, which occurs
275                             /* in a message with header lines but no content
276                             /* If we haven't already done it, force out any
277                             /* H_AFWDFROM or H_RECEIVED lines */
278                             dumprcv(type, -1,&didrcvlines,&suppress,f);
279                             dumpaff(type, -1,&didafflines,&suppress,f);
280                             break;
281                     }
282                     num = ((k < sizeof(buf)) ? k+1 : sizeof(buf));
283                     if (fgets (buf, num, tmpf) == NULL) {
284                             return (FALSE);
285                     }
286                     n = strlen (buf);
287                     k -= n;
288                     lastc = buf[n-1];

290                     if (pushrest) {
291                             pushrest = (lastc != '\n');
292                             continue;
293                     }
294                     sav_suppress = suppress;
295                     suppress = FALSE;
296                     print_from_struct = FALSE;
299                     sav_htype = htype;
297                     htype = isheader (buf, &ctf);
298                     Dout(pn, 5, "loop 2: buf = %s, htype= %d/%s\n", buf, hty
299                     /* The following order is defined in the MTA documents.
300                     switch (htype) {
301                     case H_CONT:
302                         if (sending) {
303                             suppress = sav_suppress;
304                         }
305                         continue;
306                     case H_TCOPY:
307                     case H_MIMEVERS:
308                     case H_CTYPE:
309                     case H_CLEN:
310                             if (!sending) {
311                                     savehdrs(buf,htype);
312                             }
313                             hptr = hdrlines[htype].head;
314                             if (htype == H_CLEN) {
315                                     clen = atol (hptr->value);
316                             }
317                             /*
318                              * Use values saved in hdrlines[] structure
319                              * rather than what was read from tmp file.
320                              */
321                             print_from_struct = TRUE;
```



```
322                             /* FALLTHROUGH */
323                     case H_EOH:
324                     case H_AFWDFROM:
325                     case H_AFWDCNT:
326                     case H_RECEIVED:
327                             dumprcv(type, htype,&didrcvlines,&suppress,f);
328                             dumpaff(type, htype,&didafflines,&suppress,f);
329                             continue;       /* next header line */
330                     default:
331                             pushrest = (lastc != '\n');
332                             continue;       /* next header line */
333                     case FALSE:     /* end of header */
334                             break;
335                     }

337                     /* Found the blank line after the headers. */
338                     if (n > 0) {
339                             if (fwrite(buf,1,n,f) != n)  {
340                                     sav_errno = errno;
341                                     return(FALSE);
342                             }
343                     }

345                     Dout(pn, 3,", let[%d].text = %s\n",
346                             letnum, (let[letnum].text ? "TRUE" : "FALSE"));

348                     if ((type == TTY) && (let[letnum].text == FALSE) && !pfl
349                             if (fprintf (f, "\n%s\n", binmsg) < 0)
350                             {
351                                     sav_errno = errno;
352                                     return(FALSE);
353                             }
354                             return (TRUE);
355                     }

357                     if (n == 1 && buf[0] == '\n') {
358                             n = 0;
359                     }
360                     break;
361             }

363         Dout(pn, 1, "header processed, clen/k/n = %ld/%ld/%d\n", clen, k, n);

365         if (clen >= 0) {
366                 if (((clen - n) == k) || ((clen - n) == (k - 1))) {
367                         k = clen - n;
368                 } else {
369                         /* probable content-length mismatch. show it ALL! */
370                         Dout(pn, 1, "clen conflict. using k = %ld\n", k);
371                 }
372         }

374         /* copy balance of message */
375         if (rtrncont)
376                 while (k > 0) {
377                         num = ((k < sizeof(buf)) ? k : sizeof(buf));
378                         if ((n = fread (buf, 1, num, tmpf)) <= 0) {
379                                 Dout(pn, 1, "content-length mismatch. return(FAL
380                                 return(FALSE);
381                         }
382                         k -= n;
383                         if (fwrite(buf,1,n,f) != n)  {
384                                 sav_errno = errno;
385                                 return(FALSE);
386                         }
387                 }
```

```
 389         Dout(pn, 3, "body processed, k=%ld\n", k);

 391         if (rtrncont && type != ZAP && type != REMOTE) {
 392                 if (fwrite("\n",1,1,f) != 1)  {
 393                         sav_errno = errno;
 394                         return(FALSE);
 395                 }
 396         }

 398         return(TRUE);
 399 }
```

unchanged_portion_omitted_

```
**********************************************************
    3827 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/print/bsd-sysv-commands/disable.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 */

/* $Id: disable.c 146 2006-03-24 00:26:54Z njacobs $ */


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <locale.h>
#include <libintl.h>
#include <papi.h>
#include "common.h"

static void
usage(char *program)
{
        char *name;

        if ((name = strrchr(program, '/')) == NULL)
                name = program;
        else
                name++;

        fprintf(stdout,
            gettext("Usage: %s [-c] [-W] [-r reason] destination ...\n"),
            name);
        exit(1);
}
_____unchanged_portion_omitted_

int
main(int ac, char *av[])
{
```



```
        papi_status_t status;
        papi_service_t svc = NULL;
        papi_encryption_t encryption = PAPI_ENCRYPT_NEVER;
        int exit_status = 0;
        int cancel = 0;
        int pending = 0;        /* not implemented */
        char *reason = NULL;
        int c;

        (void) setlocale(LC_ALL, "");
        (void) textdomain("SUNW_OST_OSCMD");

        while ((c = getopt(ac, av, "EcWr:")) != EOF)
                switch (c) {
                case 'c':       /* cancel active job first */
                        cancel = 1;
                        break;
                case 'W':       /* wait for active request, not implemented */
                        pending = 1;
                        break;
                case 'r':       /* reason */
                        reason = optarg;
                        break;
                case 'E':
                        encryption = PAPI_ENCRYPT_NEVER;
                        break;
                default:
                        usage(av[0]);
                }

        if (ac <= optind)
                usage(av[0]);

        while (optind < ac) {
                char *printer = av[optind++];

                status = papiServiceCreate(&svc, printer, NULL, NULL,
                    cli_auth_callback, encryption, NULL);
                if (status != PAPI_OK) {
                        fprintf(stderr, gettext(
                            "Failed to contact service for %s: %s\n"),
                            printer, verbose_papi_message(svc, status));
                        exit_status = 1;
                }

                status = papiPrinterDisable(svc, printer, reason);
                if (status == PAPI_OK) {
                        printf(gettext("printer \"%s\" now disabled\n"),
                            printer);
                } else if (status == PAPI_NOT_ACCEPTING) {
                        fprintf(stderr, gettext(
                            "Destination \"%s\" was already disabled.\n"),
                            printer);
                        exit_status = 1;
                } else {
                        /* The operation is not supported in lpd protocol */
                        if (status == PAPI_OPERATION_NOT_SUPPORTED) {
                                fprintf(stderr,
                                    verbose_papi_message(svc, status));
                        } else {
                                fprintf(stderr, gettext("disable: %s: %s\n"),
                                    printer, verbose_papi_message(svc, status));
                        }
                        exit_status = 1;
                }
```

```
155                 if (cancel != 0)
156                         cancel_active_job(svc, printer);

158                 papiServiceDestroy(svc);
159         }

161         return (exit_status);
162 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    7223 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/print/bsd-sysv-commands/lpr.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 */

/* $Id: lpr.c 146 2006-03-24 00:26:54Z njacobs $ */

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

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <locale.h>
#include <libintl.h>
#include <papi.h>
#include "common.h"

#ifdef HAVE_LIBMAGIC    /* for mimetype auto-detection */
#include <magic.h>
#endif /* HAVE_LIBMAGIC */

static void
usage(char *program)
{
        char *name;

        if ((name = strrchr(program, '/')) == NULL)
                name = program;
        else
                name++;

        fprintf(stdout,
                gettext("Usage: %s [-P printer] [-# copies] [-C class] "
                                "[-J job] [-T title] "
                                "[-p [-i indent] [-w width]] "
                                "[-1|-2|-3|-4 font] [-m] [-h] [-s] "
                                "[-filter_option] [file ..]\n"), name);
```



```
        exit(1);
}

int
main(int ac, char *av[])
{
        papi_status_t status;
        papi_service_t svc = NULL;
        papi_attribute_t **list = NULL;
        papi_job_t job = NULL;
        int exit_code = 0;
        char *printer = NULL;
        char prefetch[3];
        int prefetch_len = sizeof (prefetch);
        papi_encryption_t encryption = PAPI_ENCRYPT_NEVER;
        int dump = 0;
        int validate = 0;
        int remove = 0;
        int copy = 1;   /* default is to copy the data */
        char *document_format = "text/plain";
        int c;

        (void) setlocale(LC_ALL, "");
        (void) textdomain("SUNW_OST_OSCMD");

        while ((c = getopt(ac, av,
                        "EP:#:C:DVJ:T:w:i:hplrstdgvcfmn1:2:3:4:")) != EOF)
                switch (c) {
                case 'E':
                        encryption = PAPI_ENCRYPT_REQUIRED;
                        break;
                case 'P':
                        printer = optarg;
                        break;
                case '#':
                        papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
                                        "copies", atoi(optarg));
                        break;
                case 'C':
                        papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
                                        "rfc-1179-class", optarg);
                        break;
                case 'D':
                        dump = 1;
                        break;
                case 'J':
                        papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
                                        "job-name", optarg);
                        break;
                case 'T':
                        papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
                                        "pr-title", optarg);
                        break;
                case 'p':
                        papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
                                "document-format", "application/x-pr");
                        papiAttributeListAddBoolean(&list, PAPI_ATTR_EXCL,
                                        "pr-filter", 1);
                        break;
                case 'i':
                        papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
                                        "pr-indent", atoi(optarg));
                        break;
                case 'w':
                        papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
                                        "pr-width", atoi(optarg));
```

```
125                     break;
126             case 'h':
127                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
128                                     "job-sheets", "none");
129                     break;
130             case 'l':
131                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
132                             "document-format", "application/octet-stream");
133                     break;
134             case 'o':
135                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
136                             "document-format", "application/postscript");
137                     break;
138             case 'c':
139                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
140                             "document-format", "application/x-cif");
141                     break;
142             case 'd':
143                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
144                             "document-format", "application/x-dvi");
145                     break;
146             case 'f':
147                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
148                             "document-format", "application/x-fortran");
149                     break;
150             case 'g':
151                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
152                             "document-format", "application/x-plot");
153                     break;
154             case 'n':
155                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
156                             "document-format", "application/x-ditroff");
157                     break;
158             case 't':
159                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
160                             "document-format", "application/x-troff");
161                     break;
162             case 'v':
163                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
164                             "document-format", "application/x-raster");
165                     break;
166             case 'm':
167                     papiAttributeListAddBoolean(&list, PAPI_ATTR_EXCL,
168                             "rfc-1179-mail", 1);
169                     break;
170             case 'r':
173                     remove = 1;
171                     break;
172             case 's':
173                     copy = 0;
174                     break;
175             case 'V':       /* validate */
176                     validate = 1;
177                     break;
178             case '1':
179                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
180                             "rfc-1179-font-r", optarg);
181                     break;
182             case '2':
183                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
184                             "rfc-1179-font-i", optarg);
185                     break;
186             case '3':
187                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
188                             "rfc-1179-font-b", optarg);
189                     break;
```



```
190             case '4':
191                     papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
192                             "rfc-1179-font-s", optarg);
193                     break;
194             default:
195                     usage(av[0]);
196             }

198         if ((printer == NULL) &&
199             ((printer = getenv("PRINTER")) == NULL) &&
200             ((printer = getenv("LPDEST")) == NULL))
201                 printer = DEFAULT_DEST;

203         if (((optind + 1) == ac) && (strcmp(av[optind], "-") == 0))
204                 optind = ac;

206         if (optind != ac) {
207                 /* get the mime type of the file data */
208 #ifdef MAGIC_MIME
209                 magic_t ms;

211                 if ((ms = magic_open(MAGIC_MIME)) != NULL) {
212                         document_format = magic_file(ms, av[optind]);
213                         magic_close(ms);
214                 }
215 #else
216                 if (is_postscript(av[optind]) == 1)
217                         document_format = "application/postscript";
218 #endif
219         } else {
220                 if (is_postscript_stream(0, prefetch, &prefetch_len) == 1)
221                         document_format = "application/postscript";
222         }

224         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL, "copies", 1);
225         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
226                                 "document-format", document_format);
227         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
228                                 "job-sheets", "standard");

230         status = papiServiceCreate(&svc, printer, NULL, NULL, cli_auth_callback,
231                                         encryption, NULL);
232         if (status != PAPI_OK) {
233                 fprintf(stderr, gettext(
234                         "Failed to contact service for %s: %s\n"), printer,
235                         verbose_papi_message(svc, status));
236                 exit(1);
237         }

239         if (validate == 1)      /* validate the request can be processed */
240                 status = papiJobValidate(svc, printer, list,
241                                         NULL, &av[optind], &job);
242         else if (optind == ac)  /* no file list, use stdin */
243                 status = jobSubmitSTDIN(svc, printer, prefetch, prefetch_len,
244                                         list, &job);
245         else if (copy == 0)     /* reference the files in the job, default */
246                 status = papiJobSubmitByReference(svc, printer, list,
247                                         NULL, &av[optind], &job);
248         else                    /* copy the files before return, -c */
249                 status = papiJobSubmit(svc, printer, list,
250                                         NULL, &av[optind], &job);

252         papiAttributeListFree(list);

254         if (status != PAPI_OK) {
255                 fprintf(stderr, gettext("%s: %s\n"), printer,
```

```
256                         verbose_papi_message(svc, status));
257                 papiJobFree(job);
258                 papiServiceDestroy(svc);
259                 exit(1);
260         }

262         if (dump != 0) {
263                 list = papiJobGetAttributeList(job);
264                 printf("job attributes:\n");
265                 papiAttributeListPrint(stdout, list, "\t");
266                 printf("\n");
267         }

269         papiJobFree(job);
270         papiServiceDestroy(svc);

272         return (exit_code);
273 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    2296 Fri Aug 11 10:12:35 2017
new/usr/src/cmd/refer/glue4.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * Copyright 2017 Gary Mills
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * Copyright (c) 1980 Regents of the University of California.
 * All rights reserved. The Berkeley software License Agreement
 * specifies the terms and conditions for redistribution.
 */

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


#include <stdio.h>
#include <ctype.h>

extern char gfile[];

void
grepcall(char *in, char *out, char *arg)
{
        char line[200], *s, argig[100], *cv[50];
        char *inp, inb[500];
        FILE *qf, *gf;
        int c, alph = 0, nv = 0;
        int c, oldc = 0, alph = 0, nv = 0;
        int sv0, sv1;
        strcpy(argig, arg);
        strcat(argig, ".ig");
        strcpy(inp = inb, in);
        if (gfile[0] == 0)
                sprintf(gfile, "/tmp/rj%dg", getpid());
#if D1
        fprintf(stderr, "in grepcall, gfile %s in %o out %o\n",
            gfile, in, out);
#endif
        for (cv[nv++] = "fgrep"; c = *inp; inp++) {
                if (c == ' ')
                        c = *inp = 0;
                else if (isupper(c))
                        *inp = tolower(c);
                alph = (c == 0) ? 0 : alph+1;
                if (alph == 1)
                        cv[nv++] = inp;
                if (alph > 6)
                        *inp = 0;
                oldc = c;
        }
#if D1
        fprintf(stderr, "%d args set up\n", nv);
#endif
        {
                sv0 = dup(0);
                close(0);
                if (open(argig, 0) != 0)
                        err("Can't read fgrep index %s", argig);
```



```
                sv1 = dup(1);
                close(1);
                if (creat(gfile, 0666) != 1)
                        err("Can't write fgrep output %s", gfile);
                fgrep(nv, cv);
#if D1
                fprintf(stderr, "fgrep returned, output is..\n");
#endif
                close(0);
                dup(sv0);
                close(sv0);
                close(1);
                dup(sv1);
                close(sv1);
        }

#if D1
        fprintf(stderr, "back from fgrep\n");
#endif
        gf = fopen(gfile, "r");
        if (gf == NULL)
                err("can't read fgrep output %s", gfile);
        while (fgets(line, 100, gf) == line) {
                line[100] = 0;
#if D1
                fprintf(stderr, "read line as //%s//\n", line);
#endif
                for (s = line; *s && (*s != '\t'); s++)
                        ;
                if (*s == '\t') {
                        *s++ = '\n';
                        *s++ = 0;
                }
                if (line[0])
                        strcat(out, line);
#if D1
                fprintf(stderr, "out now /%s/\n", out);
#endif
                while (*s) s++;
#if D1
                fprintf(stderr, "line %o s %o s-1 %o\n", line, s, s[-1]);
#endif
                if (s[-1] != '\n')
                        while (!feof(gf) && getc(gf) != '\n')
                                ;
        }
        fclose(gf);
#if D1
        fprintf(stderr, "back from reading %, out %s\n", out);
#else
        unlink(gfile);
#endif
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    2017 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/refer/hunt7.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * Copyright 2017 Gary Mills
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * Copyright (c) 1980 Regents of the University of California.
 * All rights reserved. The Berkeley software License Agreement
 * specifies the terms and conditions for redistribution.
 */

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

#include <stdio.h>
#include <locale.h>
#include <assert.h>
#define SAME 0
#define FGCT 10
#define FGSIZE 150

int keepold = 1;        /* keep old things for fgrep search */
char fgspace[FGSIZE];
char *fgp = fgspace;
char *fgnames[FGCT];
char **fgnamp = fgnames;

extern char *mindex();

long
findline(char *in, char **out, int outlen, long indexdate)
{
        static char name[100] = "";
        char *p, **ftp;
        extern long gdate();
        static FILE *fa = NULL;
        long lp, llen;
        int k, nofil;
        int len, k, nofil;

#if D1
        fprintf(stderr, "findline: %s\n", in);
#endif
        if (mindex(in, '!'))
                return (0);

        nofil = in[0] == 0;
        for (p = in; *p && *p != ':' && *p != ';'; p++)
                ;
        if (*p) *p++ = 0;
        else p = in;
        k = sscanf(p, "%ld,%ld", &lp, &llen);
#ifdef D1
        fprintf(stderr, "p %s k %d lp %ld llen %ld\n", p, k, lp, llen);
#endif
        if (k < 2) {
                lp = 0;
                llen = outlen;
        }
#if D1
        fprintf(stderr, "lp %ld llen %ld\n", lp, llen);
#endif
#ifdef D1
        fprintf(stderr, "fa now %o, p %o in %o %s\n", fa, p, in, in);
#endif
        if (nofil) {
#if D1
                fprintf(stderr, "set fa to stdin\n");
#endif
                fa = stdin;
        } else
                if (strcmp(name, in) != 0 || 1) {
#if D1
                        fprintf(stderr, "old: %s new %s not equal\n", name, in);
#endif
                        if (fa != NULL)
                                fa = freopen(in, "r", fa);
                        else
                                fa = fopen(in, "r");
#if D1
                        if (fa == NULL)
                                fprintf(stderr, "failed to (re)open *%s*\n",
                                    in);
#endif
                        if (fa == NULL)
                                return (0);
                        /* err("Can't open %s", in); */
                        strcpy(name, in);
                        if (gdate(fa) > indexdate && indexdate != 0) {
                                if (keepold) {
                                        for (ftp = fgnames; ftp < fgnamp; ftp++)
                                                if (strcmp(*ftp, name) == SAME)
                                                        return (0);
                                        strcpy(*fgnamp++ = fgp, name);
                                        assert(fgnamp < fgnames+FGCT);
                                        while (*fgp && *fgp != ':')
                                                fgp++;
                                        *fgp++ = 0;
                                        assert(fgp < fgspace+FGSIZE);
                                        return (0);
                                }
                                fprintf(stderr, gettext(
                                    "Warning: index predates file '%s'\n"),
                                    name);
                        }
                }
#if D1
                else
                        fprintf(stderr, "old %s new %s same fa %o\n",
                            name, in, fa);
#endif
        if (fa != NULL) {
                fseek(fa, lp, 0);
                *out = (char *)malloc(llen + 1);
                if (*out == NULL) {
                        return (0);
                }
                (void) fread(*out, 1, llen, fa);
                len = fread(*out, 1, llen, fa);
                *(*out + llen) = 0;
#ifdef D1
                fprintf(stderr, "length as read is %d\n", len);
#endif
        }
```

```
  93         return (llen);
  94 }
______unchanged_portion_omitted_
```

```
**********************************************************
   10815 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/sh/xec.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 *
 * UNIX shell
 *
 */


#include        "defs.h"
#include        <errno.h>
#include        "sym.h"
#include        "hash.h"
#include        <sys/types.h>
#include        <sys/times.h>

pid_t parent;

void execprint(unsigned char **);

/* ========     command execution       ======== */

/*VARARGS3*/
int
execute(argt, xflags, errorflg, pf1, pf2)
struct trenod *argt;
int xflags, errorflg;
int *pf1, *pf2;
{
        /*
         * 'stakbot' is preserved by this routine
         */
        struct trenod   *t;
```



```
        unsigned char           *sav = savstak();

        sigchk();
        if (!errorflg)
                flags &= ~errflg;

        if ((t = argt) && execbrk == 0) {
                int treeflgs;
                unsigned char **com;
                int type;
                short pos;

                treeflgs = t->tretyp;
                type = treeflgs & COMMSK;

                switch (type)
                {
                case TFND:
                        {
                                struct fndnod   *f = fndptr(t);
                                struct namnod   *n = lookup(f->fndnam);

                                exitval = 0;

                                if (n->namflg & N_RDONLY)
                                        failed(n->namid, wtfailed);

                                if (flags & rshflg && (n == &pathnod ||
                                        eq(n->namid, "SHELL")))
                                        failed(n->namid, restricted);
                                /*
                                 * If function of same name is previously
                                 * defined, it will no longer be used.
                                 */
                                if (n->namflg & N_FUNCTN) {
                                        freefunc(n);
                                } else {
                                        free(n->namval);
                                        free(n->namenv);

                                        n->namval = 0;
                                        n->namflg &= ~(N_EXPORT | N_ENVCHG);
                                }
                                /*
                                 * If function is defined within function,
                                 * we don't want to free it along with the
                                 * free of the defining function. If we are
                                 * in a loop, fndnod may be reused, so it
                                 * should never be freed.
                                 */
                                if (funcnt != 0 || loopcnt != 0)
                                        f->fndref++;

                                /*
                                 * We hang a fndnod on the namenv so that
                                 * ref cnt(fndref) can be increased while
                                 * running in the function.
                                 */
                                n->namenv = (unsigned char *)f;
                                attrib(n, N_FUNCTN);
                                hash_func(n->namid);
                                break;
                        }

                case TCOM:
                        {
```

```
128                             unsigned char   *name;
127                             unsigned char   *a1, *name;
129                             int     argn, internal;
130                             struct argnod   *schain = gchain;
131                             struct ionod    *io = t->treio;
132                             short   cmdhash;
133                             short   comtype;

135                             exitval = 0;

137                             gchain = 0;
138                             argn = getarg(t);
139                             com = scan(argn);
139                             a1 = com[1];
140                             gchain = schain;

142                             if (argn != 0)
143                                     cmdhash = pathlook(com[0], 1, comptr(t)-

145                             if (argn == 0 || (comtype = hashtype(cmdhash)) =
146                                     setlist(comptr(t)->comset, 0);
147                             }

149                             if (argn && (flags&noexec) == 0)
150                             {

152                                     /* print command if execpr */
153                                     if (flags & execpr)
154                                             execprint(com);

156                                     if (comtype == NOTFOUND)
157                                     {
158                                             pos = hashdata(cmdhash);
159                                             if (pos == 1)
160                                                     failure(*com, notfound);
161                                             else if (pos == 2)
162                                                     failure(*com, badexec);
163                                             else
164                                                     failure(*com, badperm);
165                                             break;
166                                     }

168                                     else if (comtype == PATH_COMMAND)
169                                     {
170                                             pos = -1;
171                                     }

173                                     else if (comtype & (COMMAND | REL_COMMAN
174                                     {
175                                             pos = hashdata(cmdhash);
176                                     }

178                                     else if (comtype == BUILTIN) {
179                                             builtin(hashdata(cmdhash),argn,c
180                                             freejobs();
181                                             break;
182                                     }
183                                     else if (comtype == FUNCTION)
184                                     {
185                                             struct dolnod *olddolh;
186                                             struct namnod *n, *opt;
187                                             struct fndnod *f;
188                                             short index;
189                                             unsigned char **olddolv = dolv;
190                                             int olddolc = dolc;
```



```
192                                             n = findnam(com[0]);
193                                             f = fndptr(n->namenv);
194                                             /* just in case */
195                                             if (f == NULL)
196                                                     break;
197                                     /* save current positional parameters */
198                                             olddolh = (struct dolnod *)savar
199                                             f->fndref++;
200                                             funcnt++;
201                                             index = initio(io, 1);
202                                             setargs(com);
203                                             execute(f->fndval, xflags,
204                                                 errorflg, pf1, pf2);
205                                             execbrk = 0;
206                                             restore(index);
207                                             (void) restorargs(olddolh, funcn
208                                             dolv = olddolv;
209                                             dolc = olddolc;
210                                             funcnt--;
211                                             /*
212                                              * n->namenv may have been
213                                              * pointing different func.
214                                              * Therefore, we can't use
215                                              * freefunc(n).
216                                              */
217                                             freetree((struct trenod *)f);

219                                             break;
220                                     }
221                                     }
222                                     else if (t->treio == 0)
223                                     {
224                                             chktrap();
225                                             break;
226                                     }

228                             }

230                     case TFORK:
231                     {
232                             int monitor = 0;
233                             int linked = 0;

235                             exitval = 0;

237                             if (!(xflags & XEC_EXECED) || treeflgs&(FPOU|FAMP))
238                             {

240                                     int forkcnt = 1;

242                                     if (!(treeflgs&FPOU))
243                                     {
244                                             monitor = (!(xflags & XEC_NOSTOP)
245                                               && (flags&(monitorflg|jcflg|jcoff))
246                                               == (monitorflg|jcflg));
247                                             if (monitor) {
248                                                     int savefd;
249                                                     unsigned char *savebot;
250                                                     savefd = setb(-1);
251                                                     savebot = stakbot;
252                                                     prcmd(t);
253                                                     (void)setb(savefd);
254                                                     allocjob(savebot, cwdget(), moni
255                                             } else
256                                                     allocjob("", "", 0);
```

```
258                             }

260                             if (treeflgs & (FPOU|FAMP)) {
261                                     link_iodocs(iotemp);
262                                     linked = 1;
263                             }

265                             while ((parent = fork()) == -1)
266                             {
267                             /*
268                              * FORKLIM is the max period between forks -
269                              * power of 2 usually.  Currently shell tries
270                              * after 2,4,8,16, and 32 seconds and then quits
271                              */

273                             if ((forkcnt = (forkcnt * 2)) > FORKLIM)
274                             {
275                                     switch (errno)
276                                     {
277                                     case ENOMEM:
278                                             deallocjob();
279                                             error(noswap);
280                                             break;
281                                     default:
282                                             deallocjob();
283                                             error(nofork);
284                                             break;
285                                     }
286                             } else if (errno == EPERM) {
287                                     deallocjob();
288                                     error(eacces);
289                                     break;
290                             }
291                             sigchk();
292                             sh_sleep(forkcnt);
293                             }

295                             if (parent) {
296                                     if (monitor)
297                                             setpgid(parent, 0);
298                                     if (treeflgs & FPIN)
299                                             closepipe(pf1);
300                                     if (!(treeflgs&FPOU)) {
301                                             postjob(parent,!(treeflgs&FAMP))
302                                             freejobs();
303                                     }
304                                     chktrap();
305                                     break;
306                             }
307                             mypid = getpid();
308                     }

310                     /*
311                      * Forked process:  assume it is not a subshell for
312                      * now.  If it is, the presence of a left parenthesis
313                      * will trigger the jcoff flag to be turned off.
314                      * When jcoff is turned on, monitoring is not going on
315                      * and waitpid will not look for WUNTRACED.
316                      */

318                     flags |= (forked|jcoff);

320                     fiotemp  = 0;

322                     if (linked == 1) {
323                             swap_iodoc_nm(iotemp);
```



```
324                             xflags |= XEC_LINKED;
325                     } else if (!(xflags & XEC_LINKED))
326                             iotemp = 0;
327 #ifdef ACCT
328                     suspacct();
329 #endif
330                     settmp();
331                     oldsigs();

333                     if (!(treeflgs & FPOU))
334                             makejob(monitor, !(treeflgs & FAMP));

336                     /*
337                      * pipe in or out
338                      */
339                     if (treeflgs & FPIN)
340                     {
341                             renamef(pf1[INPIPE], 0);
342                             close(pf1[OTPIPE]);
343                     }

345                     if (treeflgs & FPOU)
346                     {
347                             close(pf2[INPIPE]);
348                             renamef(pf2[OTPIPE], 1);
349                     }

351                     /*
352                      * io redirection
353                      */
354                     initio(t->treio, 0);

356                     if (type == TFORK)
357                             execute(forkptr(t)->forktre, xflags | XEC_EXECED
358                     else if (com[0] != ENDARGS)
359                     {
360                             eflag = 0;
361                             setlist(comptr(t)->comset, N_EXPORT);
362                             rmtemp(0);
363                             clearjobs();
364                             execa(com, pos);
365                     }
366                     done(0);
367             }

369             case TPAR:
370                     /* Forked process is subshell:  may want job control */
371                     flags &= ~jcoff;
372                     clearjobs();
373                     execute(parptr(t)->partre, xflags, errorflg);
374                     done(0);

376             case TFIL:
377                     {
378                             int pv[2];

380                             chkpipe(pv);
381                             if (execute(lstptr(t)->lstlef, xflags & XEC_NOST
382                                     execute(lstptr(t)->lstrit, xflags, error
383                             else
384                                     closepipe(pv);
385                     }
386                     break;

388             case TLST:
389                     execute(lstptr(t)->lstlef, xflags&XEC_NOSTOP, errorflg);
```

```
390                         /* Update errorflg if set -e is invoked in the sub-sh*/
391                         execute(lstptr(t)->lstrit, xflags, (errorflg | (eflag &
392                         break;
393
394                 case TAND:
395                 case TORF:
396                 {
397                         int xval;
398                         xval = execute(lstptr(t)->lstlef, XEC_NOSTOP, 0);
399                         if ((xval == 0) == (type == TAND))
400                                 execute(lstptr(t)->lstrit, xflags|XEC_NOSTOP, er
401                         break;
402                 }
403
404                 case TFOR:
405                         {
406                                 struct namnod *n = lookup(forptr(t)->fornam);
407                                 unsigned char   **args;
408                                 struct dolnod *argsav = 0;
409
410                                 if (forptr(t)->forlst == 0)
411                                 {
412                                         args = dolv + 1;
413                                         argsav = useargs();
414                                 }
415                                 else
416                                 {
417                                         struct argnod *schain = gchain;
418
419                                         gchain = 0;
420                                         args = scan(getarg(forptr(t)->forlst));
421                                         gchain = schain;
422                                 }
423                                 loopcnt++;
424                                 while (*args != ENDARGS && execbrk == 0)
425                                 {
426                                         assign(n, *args++);
427                                         execute(forptr(t)->fortre, XEC_NOSTOP, e
428                                         if (breakcnt < 0)
429                                                 execbrk = (++breakcnt != 0);
430                                 }
431                                 if (breakcnt > 0)
432                                                 execbrk = (--breakcnt != 0);
433
434                                 loopcnt--;
435                                 if(argsav)
436                                         argfor = (struct dolnod *)freeargs(argsa
437                         }
438                         break;
439
440                 case TWH:
441                 case TUN:
442                         {
443                                 int     i = 0;
444
445                                 loopcnt++;
446                                 while (execbrk == 0 && (execute(whptr(t)->whtre,
447                                     XEC_NOSTOP, 0) == 0) == (type == TWH) &&
448                                     (flags&noexec) == 0)
449 {
450                                         i = execute(whptr(t)->dotre, XEC_NOSTOP,
451                                         if (breakcnt < 0)
452                                                 execbrk = (++breakcnt != 0);
453                                 }
454                                 if (breakcnt > 0)
455                                                 execbrk = (--breakcnt != 0);
```



```
457                                 loopcnt--;
458                                 exitval = i;
459                         }
460                         break;
461
462                 case TIF:
463                         if (execute(ifptr(t)->iftre, XEC_NOSTOP, 0) == 0)
464                                 execute(ifptr(t)->thtre, xflags|XEC_NOSTOP, erro
465                         else if (ifptr(t)->eltre)
466                                 execute(ifptr(t)->eltre, xflags|XEC_NOSTOP, erro
467                         else
468                                 exitval = 0;    /* force zero exit for if-then-f
469                         break;
470
471                 case TSW:
472                         {
473                                 unsigned char   *r = mactrim(swptr(t)->swarg);
474                                 struct regnod *regp;
475
476                                 regp = swptr(t)->swlst;
477                                 while (regp)
478                                 {
479                                         struct argnod *rex = regp->regptr;
480
481                                         while (rex)
482                                         {
483                                                 unsigned char   *s;
484
485                                                 if (gmatch(r, s = macro(rex->arg
486                                                 {
487                                                         execute(regp->regcom, XE
488                                                         regp = 0;
489                                                         break;
490                                                 }
491                                                 else
492                                                         rex = rex->argnxt;
493                                         }
494                                         if (regp)
495                                                 regp = regp->regnxt;
496                                 }
497                         }
498                         break;
499                 }
500                 exitset();
501         }
502         sigchk();
503         tdystak(sav);
504         flags |= eflag;
505         return(exitval);
506 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   18493 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/svr4pkg/pkgremove/special.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */


/*
 * special.c
 *
 * This module contains code required to remove special contents from
 * the contents file when a pkgrm is done on a system upgraded to use
 * the new database.
 */

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <fnmatch.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pkgstrct.h>
#include "pkglib.h"
#include <libintl.h>

/* This specifies the maximum length of a contents file line read in. */
#define LINESZ  8192

#define SPECIAL_MALLOC  "unable to maintain package contents text due to "\
                        "insufficient memory."
#define SPECIAL_ACCESS  "unable to maintain package contents text due to "\
                        "an access failure."
#define SPECIAL_INPUT   "unable to maintain package contents text: alternate "\
                        "root path too long"
```



```
/*
 * strcompare
 *
 * This function is used by qsort to sort an array of special contents
 * rule strings.  This array must be sorted to facilitate efficient
 * rule processing.  See qsort(3c) regarding qsort compare functions.
 */
static int
strcompare(const void *pv1, const void *pv2)
{
        char **ppc1 = (char **) pv1;
        char **ppc2 = (char **) pv2;
        int i = strcmp(*ppc1, *ppc2);
        if (i < 0)
                return (-1);
        if (i > 0)
                return (1);
        return (0);
}
_____unchanged_portion_omitted_

/*
 * -----------------------------------------------------------------------
 * Externally visible function.
 */

/*
 * special_contents_remove
 *
 * Given a set of entries to remove and an alternate root, this function
 * will do everything required to ensure that the entries are removed
 * from the contents file if they are listed in the special_contents
 * file.  The contents file will get changed only in the case that the
 * entire operation has succeeded.
 *
 *  ient        The number of entries.
 *  ppcfent     The entries to remove.
 *  pcroot      The alternate install root.  Could be NULL.  In this
 *              case, assume root is '/'
 *
 * Result: 0 on success, nonzero on failure.  If an error occurs, an
 *    error string will get output to standard error alerting the user.
 * Side effects: The contents file may change as a result of this call,
 *    such that lines in the in the file will be changed or removed.
 *    If the call fails, a t.contents file may be left behind.  This
 *    temporary file should be removed subsequently.
 */
int
special_contents_remove(int ient, struct cfent **ppcfent, const char *pcroot)
{
        int result = 0;         /* Assume we will succeed.  Return result. */
        char **ppcSC = NULL;    /* The special contents rules, sorted. */
        int i;                  /* Index into contents & special contents */
        int i, j;               /* Indexes into contents & special contents */
        FILE *fpi = NULL,       /* Input of contents file */
             *fpo = NULL;       /* Output to temp contents file */
        char cpath[PATH_MAX],   /* Contents file path */
             tcpath[PATH_MAX];  /* Temp contents file path */
        const char *pccontents = "var/sadm/install/contents";
        const char *pctcontents = "var/sadm/install/t.contents";
        char line[LINESZ];      /* Reads in and writes out contents lines. */
        time_t t;               /* Used to create a timestamp comment. */
        int max;                /* Max number of special contents entries. */
        int *piIndex;           /* An index to ppcfents to remove from cfile */

        cpath[0] = tcpath[0] = '\0';
```

```
517         if (ient == 0 || ppcfent == NULL || ppcfent[0] == NULL) {
518                 goto remove_done;
519         }

521         if ((get_special_contents(pcroot, &ppcSC, &max)) != 0) {
522                 result = 1;
523                 goto remove_done;
524         }

526         /* Check if there are no special contents actions to take. */
527         if (ppcSC == NULL) {
528                 goto remove_done;
529         }

531         if (pcroot == NULL) pcroot = "/";
532         if (pcroot[strlen(pcroot) - 1] == '/') {
533                 if (snprintf(cpath, PATH_MAX, "%s%s", pcroot, pccontents)
534                     >= PATH_MAX ||
535                     snprintf(tcpath, PATH_MAX, "%s%s", pcroot, pctcontents)
536                     >= PATH_MAX) {
537                         progerr(gettext(SPECIAL_INPUT));
538                         result = -1;
539                         goto remove_done;
540                 }
541         } else {
542                 if (snprintf(cpath, PATH_MAX, "%s/%s", pcroot, pccontents)
543                     >= PATH_MAX ||
544                     snprintf(tcpath, PATH_MAX, "%s/%s", pcroot, pctcontents)
545                     >= PATH_MAX) {
546                         progerr(gettext(SPECIAL_INPUT));
547                         result = -1;
548                         goto remove_done;
549                 }
550         }

552         /* Open the temporary contents file to write, contents to read. */
553         if (access(cpath, F_OK | R_OK) != 0) {
554                 /*
555                  * This is not a problem since no contents means nothing
556                  * to remove due to special contents rules.
557                  */
558                 result = 0;
559                 cpath[0] = '\0'; /* This signals omission of 'rename cleanup' */
560                 goto remove_done;
561         }

563         if (access(cpath, W_OK) != 0) {
564                 /* can't write contents file, something is wrong. */
565                 progerr(gettext(SPECIAL_ACCESS));
566                 result = 1;
567                 goto remove_done;

569         }

571         if ((fpi = fopen(cpath, "r")) == NULL) {
572                 /* Given the access test above, this should not happen. */
573                 progerr(gettext(SPECIAL_ACCESS));
574                 result = 1;
575                 goto remove_done;
576         }

578         if ((fpo = fopen(tcpath, "w")) == NULL) {
579                 /* open t.contents failed */
580                 progerr(gettext(SPECIAL_ACCESS));
581                 result = 1;
```



```
582                 goto remove_done;
583         }

585         if (generate_special_contents_rules(ient, ppcfent, ppcSC, max, &piIndex)
586              != 0) {
587                 result = 1;
588                 goto remove_done;
589         }

591         /*
592          * Copy contents to t.contents unless there is an entry in
593          * the ppcfent array which corresponds to an index set to 1.
594          *
595          * These items are the removed package contents which matche an
596          * entry in ppcSC (the special_contents rules).
597          *
598          * Since both the contents and rules are sorted, we can
599          * make a single efficient pass.
600          */
601         (void) memset(line, 0, LINESZ);

603         for (i = 0; fgets(line, LINESZ, fpi) != NULL; ) {
602         for (i = 0, j = 0; fgets(line, LINESZ, fpi) != NULL; ) {

605                 char *pcpath = NULL;

607                 /*
608                  * Note:  This could be done better:  We should figure out
609                  * which are the last 2 lines and only trim those off.
610                  * This will suffice to do this and will only be done as
611                  * part of special_contents handling.
612                  */
613                 if (line[0] == '#')
614                         continue; /* Do not copy the final 2 comment lines */

616                 pcpath = get_path(line);

618                 if (pcpath != NULL && i < ient) {
619                         int k;
620                         while (piIndex[i] == 0)
621                                 i++;

623                         if (i < ient)
624                                 k = pathcmp(pcpath, ppcfent[i]);

626                         if (k < 0 || i >= ient) {
627                                 /* Just copy contents -> t.contents */
628                                 /*EMPTY*/
629                         } else if (k == 0) {
630                                 /* We have a match.  Do not copy the content. */
631                                 i++;
632                                 free(pcpath);
633                                 (void) memset(line, 0, LINESZ);
634                                 continue;
635                         } else while (i < ient) {

637                                 /*
638                                  * This is a complex case:  The content
639                                  * entry is further along alphabetically
640                                  * than the rule.  Skip over all rules which
641                                  * apply until we come to a rule which is
642                                  * greater than the current entry, or equal
643                                  * to it.  If equal, do not copy, otherwise
644                                  * do copy the entry.
645                                  */
646                                 if (piIndex[i] == 0) {
```

```
647                                     i++;
648                                     continue;
649                             } else if ((k = pathcmp(pcpath, ppcfent[i]))
650                                 >= 0) {
651                                     i++;
652                                     if (k == 0) {
653                                             free(pcpath);
654                                             (void) memset(line, 0, LINESZ);
655                                             break;
656                                     }
657                             } else {
658                                     /* path < rule, end special case */
659                                     break;
660                             }
661                         }

663                         /*
664                          * Avoid copying the old content when path == rule
665                          * This occurs when the complex case ends on a match.
666                          */
667                         if (k == 0)
668                                 continue;
669                 }

671                 if (fprintf(fpo, "%s", line) < 0) {
672                         /* Failing to write output would be catastrophic. */
673                         progerr(gettext(SPECIAL_ACCESS));
674                         result = 1;
675                         break;
676                 }
677                 (void) memset(line, 0, LINESZ);
678         }

680         t = time(NULL);
681         (void) fprintf(fpo, "# Last modified by pkgremove\n");
682         (void) fprintf(fpo, "# %s", ctime(&t));

684 remove_done:
685         free_special_contents(&ppcSC, max);

687         if (fpi != NULL)
688                 (void) fclose(fpi);

690         if (fpo != NULL)
691                 (void) fclose(fpo);

693         if (result == 0) {
694                 if (tcpath[0] != '\0' && cpath[0] != '\0' &&
695                     rename(tcpath, cpath) != 0) {
696                         progerr(gettext(SPECIAL_ACCESS));
697                         result = 1;
698                 }
699         } else {
700                 if (tcpath[0] != '\0' && remove(tcpath) != 0) {
701                         /*
702                          * Do not output a diagnostic message.  This condition
703                          * occurs only when we are unable to clean up after
704                          * a failure.  A temporary file will linger.
705                          */
706                         result = 1;
707                 }
708         }

710         return (result);
711 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    7107 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/troff/nroff.d/n6.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

#include "tdef.h"
#include "tw.h"
#include "ext.h"
#include <ctype.h>

/*
 * n6.c -- width functions, sizes and fonts
 */

int     bdtab[NFONT+1] ={ 0, 0, 0, 3, 3, 0, };
int     sbold = 0;
int     fontlab[NFONT+1] = { 0, 'R', 'I', 'B', PAIR('B','I'), 'S', 0 };

extern  int     nchtab;

int
width(j)
tchar j;
{
        int     i, k;
```



```
        if (j & (ZBIT|MOT)) {
                if (iszbit(j))
                        return(0);
                if (isvmot(j))
                        return(0);
                k = absmot(j);
                if (isnmot(j))
                        k = -k;
                return(k);
        }
        i = cbits(j);
        if (i < ' ') {
                if (i == '\b')
                        return(-widthp);
                if (i == PRESC)
                        i = eschar;
                else if (iscontrol(i))
                        return(0);
        }
        if (i==ohc)
                return(0);
#ifdef EUC
#ifdef NROFF
        if (multi_locale) {
                if ((j & MBMASK) || (j & CSMASK)) {
                        switch(j & MBMASK) {
                                case BYTE_CHR:
                                case LASTOFMB:
                                        k = t.Char * csi_width[cs(j)];
                                        break;
                                default:
                                        k = 0;
                                        break;
                        }
                        widthp = k;
                        return(k);
                }
        }
        i &= 0x1ff;
#endif /* NROFF */
#endif /* EUC */
        i = trtab[i];
        if (i < 32)
                return(0);
        k = t.width[i] * t.Char;
        widthp = k;
        return(k);
}
```

_____unchanged_portion_omitted_

```
tchar setht()           /* set character height from \H'...' */
{
        int     n;
        tchar c;

        getch();
        (void) inumb(&apts);
        n = inumb(&apts);
        getch();
        return(0);
}
```

_____unchanged_portion_omitted_

```
**********************************************************
    2310 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/volcheck/volcheck.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <strings.h>
#include <signal.h>
#include <errno.h>
#include <libintl.h>
#include <sys/types.h>

#include "vold.h"
#include "rmm_common.h"

char *progname = "volcheck";

static void
usage()
{
        fprintf(stderr,
            gettext("usage: %s [-t #secs -i #secs] [-v] [path | nickname]\n"),
            progname);
        fprintf(stderr,
            gettext("If path is not supplied all media is checked\n"));
}

int
main(int argc, char **argv)
{
        const char      *opts = "itv";
        int             c;
        boolean_t       opt_i = B_FALSE;
        boolean_t       opt_t = B_FALSE;
        boolean_t       opt_v = B_FALSE;
        LibHalContext   *hal_ctx;
```



```
        DBusError       error;
        rmm_error_t     rmm_error;
        int             ret = 0;

        vold_init(argc, argv);

        while ((c = getopt(argc, argv, opts)) != EOF) {
                switch (c) {
                case 'i':
                        opt_i = B_TRUE;
                        break;
                case 't':
                        opt_t = B_TRUE;
                        break;
                case 'v':
                        opt_v = B_TRUE;
                        break;
                default:
                        usage();
                        return (1);
                }
        }

        if ((hal_ctx = rmm_hal_init(0, 0, 0, 0, &error, &rmm_error)) == NULL) {
                (void) fprintf(stderr,
                    gettext("HAL initialization failed: %s\n"),
                    rmm_strerror(&error, rmm_error));
                rmm_dbus_error_free(&error);
                return (1);
        }

        if (optind == argc) {
                /* no name provided, check all */
                ret = rmm_rescan(hal_ctx, NULL, B_FALSE);
        } else {
                for (; optind < argc; optind++) {
                        if (rmm_rescan(hal_ctx, argv[optind], B_FALSE) != 0) {
                                ret = 1;
                        }
                }
        }

        rmm_hal_fini(hal_ctx);

        return (ret);
}
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   13606 Fri Aug 11 10:12:36 2017
new/usr/src/cmd/ypcmd/ypserv.c
8485 Remove set but unused variables in usr/src/cmd
**********************************************************
```

```
/*
 * 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 2017 Gary Mills
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      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.
 */

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

/*
 * This contains the mainline code for the YP server.  Data
 * structures which are process-global are also in this module.
 */

/* this is so that ypserv will compile under 5.5 */
#define _SVID_GETTOD
#include <sys/time.h>
extern int gettimeofday(struct timeval *);

#include "ypsym.h"
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include <netconfig.h>
#include <netdir.h>
#include <sys/select.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdarg.h>
#include <signal.h>
#include "shim.h"
```



```
#include "yptol.h"
#include <syslog.h>

static char register_failed[] = "ypserv:  Unable to register service for ";
bool silent = TRUE;

/*
 * client_setup_failure will be TRUE, if setup of the
 * connection to rpc.nisd_resolv failed
 */
bool client_setup_failure = FALSE;

/* N2L options */
bool init_dit = FALSE;
bool init_containers = FALSE;
bool init_maps = FALSE;
char **ldapCLA = NULL;

/* For DNS forwarding command line option (-d) */
bool dnsforward = FALSE;
int resolv_pid = 0;
CLIENT *resolv_client = NULL;
char *resolv_tp = "ticots";

#ifdef MINUS_C_OPTION
/* For cluster support (-c) */
bool multiflag = FALSE;
#endif

static char logfile[] = "/var/yp/ypserv.log";
void logprintf(char *format, ...);

static void ypexit(void);
static void ypinit(int argc, char **argv);
static void ypdispatch(struct svc_req *rqstp, SVCXPRT *transp);
static void ypolddispatch(struct svc_req *rqstp, SVCXPRT *transp);
static void ypget_command_line_args(int argc, char **argv);
extern void setup_resolv(bool *fwding, int *child,
                         CLIENT **client, char *tp_type, long prognum);
static void cleanup_resolv(int);

/*
 * This is the main line code for the yp server.
 */
int
main(int argc, char **argv)
{
        if (geteuid() != 0) {
                fprintf(stderr, "must be root to run %s\n", argv[0]);
                exit(1);
        }

        /* Set up shop */
        ypinit(argc, argv);

        /* If requested set up the N2L maps. May take a while */
        if (init_dit)
                if (FAILURE == dump_maps_to_dit(init_containers)) {
                        fprintf(stderr, "Fatal error dumping maps to DIT."
                        " See syslog and LDAP server logs for details.\n");
                        exit(1);
                }

        if (init_maps)
                if (FAILURE == dump_dit_to_maps()) {
                        fprintf(stderr, "Fatal error dumping DIT to maps."
```

```
126                         " See syslog and LDAP server logs for details.\n");
127                         exit(1);
128                 }

130         /*
131          * If we were asked to init the maps now exit. User will then use
132          * ypstart to restart ypserv and all the other NIS daemons.
133          */
134         if (init_dit || init_maps) {
135                 printf("Map setup complete. Please now restart NIS daemons "
136                         "with ypstart.\n");
137                 exit(0);
138         }

140         svc_run();

142         /*
143          * This is stupid, but the compiler likes to warn us about the
144          * absence of returns from main()
145          */
146         return (0);
147 }
_____unchanged_portion_omitted_

165 #define MAXSERVICES     (sizeof (service)/sizeof (service[0]))

167 int             service_classes[MAXSERVICES];

169 /*
170  * Does startup processing for the yp server.
171  */
172 static void
173 ypinit(int argc, char **argv)
174 {
175         int pid;
176         int stat;
177         int stat, t;
178         struct sigaction act;
179         int ufd, tfd;
180         SVCXPRT *utransp, *ttransp;
181         struct netconfig *nconf;
182         int connmaxrec = RPC_MAXDATASIZE;
183         int i, j, services = 0;


185         /*
186          * Init yptol flags. Will get redone by init_lock_system() but we need
187          * to know if we should parse yptol cmd line options.
188          */
189         init_yptol_flag();

191         ypget_command_line_args(argc, argv);

193         if (silent) {
194                 pid = (int)fork();

196                 if (pid == -1) {
197                     logprintf("ypserv:  ypinit fork failure.\n");
198                     ypexit();
199                 }

201                 if (pid != 0) {
202                     exit(0);
203                 }
204         }
```



```
206         if (!init_lock_system(FALSE)) {
207                 ypexit();
208         }

210         get_secure_nets(argv[0]);

212         if (silent) {
213                 closelog();
214                 closefrom(3);
215         }

217         if (yptol_mode) {
218                 stat = parseConfig(ldapCLA, NTOL_MAP_FILE);
219                 if (stat == 1) {
220                         logprintf("NIS to LDAP mapping inactive.\n");
221                 } else if (stat != 0) {
222                         logprintf("Aborting after NIS to LDAP mapping "
223                                                         "error.\n");
224                         fflush(stderr);
225                         exit(-1);
226                 }
227         }

229         if (silent) {
230                 freopen("/dev/null", "r", stdin);
231                 if (access(logfile, _IOWRT)) {
232                     freopen("/dev/null", "w", stdout);
233                     freopen("/dev/null", "w", stderr);
234                 } else {
235                     freopen(logfile, "a", stdout);
236                     freopen(logfile, "a", stderr);
237                 }

239                 (void) open("/dev/tty", 2);
240                 t = open("/dev/tty", 2);

241                 setpgrp();
242         }

244 #ifdef  SYSVCONFIG
245         sigset(SIGHUP, (void (*)())sysvconfig);
246 #else
247         sigset(SIGHUP, SIG_IGN);
248 #endif

250         /*
251          * Setting disposition to SIG_IGN will not create zombies when child
252          * processes terminate.
253          */
254         sigset(SIGCHLD, SIG_IGN);

256         act.sa_handler = cleanup_resolv;
257         sigemptyset(&act.sa_mask);
258         act.sa_flags = SA_RESETHAND;
259         sigaction(SIGTERM, &act, (struct sigaction *)NULL);
260         sigaction(SIGQUIT, &act, (struct sigaction *)NULL);
261         sigaction(SIGABRT, &act, (struct sigaction *)NULL);
262         sigaction(SIGBUS, &act, (struct sigaction *)NULL);
263         sigaction(SIGSEGV, &act, (struct sigaction *)NULL);

265         /*
266          * Set non-blocking mode and maximum record size for
267          * connection oriented RPC transports.
268          */
269         if (!rpc_control(RPC_SVC_CONNMAXREC_SET, &connmaxrec)) {
270                 logprintf("unable to set maximum RPC record size");
```

```
271         }

273         svc_unreg(YPPROG, YPVERS);
274         svc_unreg(YPPROG, YPVERS_ORIG);

276         for (i = 0; i < sizeof (service)/sizeof (ypservice_t); i++) {

278                 service_classes[i] = -1;

280                 if ((nconf = getnetconfigent(service[i].netid)) == NULL) {
281                         logprintf("getnetconfigent(\"%s\") failed\n",
282                                         service[i].netid);
283                         continue;
284                 }

286                 if ((service[i].fd = t_open(nconf->nc_device, O_RDWR, NULL)) <
287                         0) {
288                         logprintf("t_open failed for %s\n", service[i].netid);
289                         freenetconfigent(nconf);
290                         continue;
291                 }

293                 if (netdir_options(nconf, ND_SET_RESERVEDPORT, service[i].fd,
294                         NULL) < 0) {
295                         logprintf("could not set reserved port for %s\n",
296                                 service[i].netid);
297                         (void) close(service[i].fd);
298                         service[i].fd = -1;
299                         freenetconfigent(nconf);
300                         continue;
301                 }

303                 if ((service[i].xprt = svc_tli_create(service[i].fd, nconf,
304                         NULL, 0, 0)) == NULL) {
305                         logprintf("svc_tli_create failed for %s\n",
306                                 service[i].netid);
307                         (void) close(service[i].fd);
308                         service[i].fd = -1;
309                         freenetconfigent(nconf);
310                         continue;
311                 }

313                 if (!svc_reg(service[i].xprt, YPPROG, YPVERS, ypdispatch,
314                         nconf)) {
315                         logprintf("%s %s\n", service[i].netid, register_failed);
316                         svc_destroy(service[i].xprt);
317                         service[i].xprt = 0;
318                         (void) close(service[i].fd);
319                         service[i].fd = -1;
320                         freenetconfigent(nconf);
321                         continue;
322                 }

324                 if (service[i].olddispatch && !svc_reg(service[i].xprt, YPPROG,
325                                 YPVERS_ORIG, ypolddispatch, nconf)) {
326                         logprintf("old %s %s\n",
327                                 service[i].netid, register_failed);
328                         /* Can only unregister prognum/versnum */
329                         svc_destroy(service[i].xprt);
330                         service[i].xprt = 0;
331                         (void) close(service[i].fd);
332                         service[i].fd = -1;
333                         freenetconfigent(nconf);
334                         continue;
335                 }
```



```
337                 services++;
338                 service[i].ok = 1;
339                 service_classes[i] = service[i].class;

341                 freenetconfigent(nconf);

343         }

345         /*
346          * Check if we managed to register enough services to continue.
347          * It's OK if we managed to register all IPv4 services but no
348          * IPv6, or the other way around, but not if we (say) registered
349          * IPv4 UDP but not TCP.
350          */
351         if (services > 0) {
352                 for (j = 0; j < MAXSERVICES; j++) {
353                         if (service_classes[j] >= 0) {
354                                 /*
355                                  * Must have all services of this class
356                                  * registered.
357                                  */
358                                 for (i = 0; i < MAXSERVICES; i++) {
359                                         if (service[i].ok == 0 &&
360                                                 service[i].class ==
361                                                 service_classes[j]) {
362                                                 logprintf(
363                         "unable to register all services for class %d\n",
364                                                         service[i].class);
365                                                 ypexit();
366                                         }
367                                 }
368                         }
369                 }
370         } else {
371                 logprintf("unable to register any services\n");
372                 ypexit();
373         }

375         /* Now we setup circuit_n or yp_all() and yp_update() will not work */
376         if (!svc_create(ypdispatch, YPPROG, YPVERS, "circuit_n")) {
377                 logprintf("circuit_n %s\n", register_failed);
378                 ypexit();
379         }

381         if (dnsforward) {
382                 setup_resolv(&dnsforward, &resolv_pid,
383                                 &resolv_client, resolv_tp, 0);
384                 if (resolv_client == NULL)
385                         client_setup_failure = TRUE;
386         }
387 }
```

_____***unchanged_portion_omitted_***