

new/usr/src/cmd/bnu/uuglist.c

1

3425 Sun Jul 16 20:04:36 2017

new/usr/src/cmd/bnu/uuglist.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 2005 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

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

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

32 #include      "uucp.h"

34 #define MAXLENGTH 256
35 #define C_MAX      512

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

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

44 int
45 main(argc, argv)
46 int argc;
47 char **argv;
48 {
49     int c, i, uopt = 0;
50     char prev[2 * NAMESIZE];

52     if (eaccess(GRADES, 04) == -1) {
53         (void) fprintf(stderr, "No administrator defined service grades
54         exit(0);
55     }

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

new/usr/src/cmd/bnu/uuglist.c

2

```
60         uopt++;
61         break;
62     case 'x':
63         Debug = atoi(optarg);
64         if (Debug < 0)
65             Debug = 1;
66         break;
67     default:
68         (void) fprintf(stderr, "usage: uuglist [-u] [-xLEVEL]\n"
69         exit(-1);
70     }

72     if (uopt) {
73         Uid = getuid();

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

78         (void) guinfo(Uid, User);

80         uproc();
81     } else
82         rproc();

84     for (i = 0; i < Nnames; i++) {
86         if (EQUALS(Nptr[i], prev))
87             continue;

89         puts(Nptr[i]);
90         (void) strcpy(prev, Nptr[i]);
91     }
92     return (0);
93 }
94
95 _____ unchanged_portion_omitted _____

125 void
126 rproc()
127 {
128     FILE *cfd;
129     char line[BUFSIZ];
130     char *carray[C_MAX];
131     int na;

132     cfd = fopen(GRADES, "r");

134     while (rdfulline(cfd, line, BUFSIZ) != 0) {
136         (void) getargs(line, carray, C_MAX);
137         na = getargs(line, carray, C_MAX);
138         insert(carray[0]);
139     }

140     (void) fclose(cfd);
141     return;
142 }
143
144 _____ unchanged_portion_omitted _____
```

new/usr/src/cmd/bnu/uustat.c

1

```
*****
38453 Sun Jul 16 20:04:36 2017
new/usr/src/cmd/bnu/uustat.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 2005 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

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

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

34 #ifdef V7
35 #define O_RDONLY      0
36 #endif
37 #define KILLMSG "the system administrator has killed job"
38 #define USAGE1 "[-q] | [-m] | [-k JOB [-n]] | [-r JOB [-n]] | [-p]"
39 #define USAGE2 "[-a] [-s SYSTEM [-j]] [-u USER] [-S STATE]"
40 #define USAGE3 "-t SYSTEM [-d number] [-c]"
41 #define LOCK "LCK.."
42 #define STST_MAX      132
43 #define MAXDATE      12
44 #define MINTIME      60
45 #define MINUTES      60
46 #define CHAR      "a"
47 #define MAXSTATE      4
48 /* #include "logs.h" */
49 struct m {
50     char      mach[15];          /* machine name */
51     char      locked;
52     int      ccount, xcount;
53     int      count, type;
54     long      retrytime;
55     time_t      lasttime;
56     short      c_age;          /* age of oldest C. file */
57     short      x_age;          /* age of oldest X. file */
58     char      stst[STST_MAX];
59 } M[UUSTAT_TBL+2];
unchanged_portion_omitted
```

new/usr/src/cmd/bnu/uustat.c

2

```
1540 static void
1541 friendlytime(uplimit, lolimit)
1542 char *uplimit, *lolimit;
1543 {
1544
1545     char c;
1546
1547     c = *(uplimit+6);
1548     friendlyptr->uhour[0] = *(uplimit+6);
1549     friendlyptr->uhour[1] = *(uplimit+7);
1550     friendlyptr->lhour[0] = *(lolimit+6);
1551     friendlyptr->lhour[1] = *(lolimit+7);
1552     friendlyptr->umin[0] = *(uplimit+8);
1553     friendlyptr->umin[1] = *(uplimit+9);
1554     friendlyptr->lmin[0] = *(lolimit+8);
1555     friendlyptr->lmin[1] = *(lolimit+9);
1556
1557     friendlyptr->uhour[2] = '\0';
1558     friendlyptr->lhour[2] = '\0';
1559     friendlyptr->umin[2] = '\0';
1560     friendlyptr->lmin[2] = '\0';
1561     return;
1562 }
unchanged_portion_omitted
```

new/usr/src/cmd/cmd-inet/usr.bin/talk/get_names.c

1

```
*****
3573 Sun Jul 16 20:04:37 2017
new/usr/src/cmd/cmd-inet/usr.bin/talk/get_names.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 1997 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

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

31 /*
32  * University Copyright- Copyright (c) 1982, 1986, 1988
33  * The Regents of the University of California
34  * All Rights Reserved
35  *
36  * University Acknowledgment- Portions of this document are derived from
37  * software developed by the University of California, Berkeley, and its
38  * contributors.
39  */

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

41 #include "talk.h"
42 #include "ctl.h"
43 #include <locale.h>
44 #include <pwd.h>
45 #include <sys/systeminfo.h>

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

49 extern CTL_MSG msg;

51 /*
52  * Determine the local and remote user, tty, and machines
53  */

55 struct hostent *gethostbyname();

57 void
58 get_names(argc, argv)
59 int argc;
```

new/usr/src/cmd/cmd-inet/usr.bin/talk/get_names.c

2

```
60 char *argv[];
61 {
62     char hostname[HOST_NAME_LENGTH + 1];
63     char *rem_name;
64     char *my_name;
65     char *my_machine_name;
66     char *rem_machine_name;
67     char *my_tty;
68     char *rem_tty;
69     char *ptr;
70     int name_length;

71     if (argc < 2) {
72         fprintf(stderr,
73             "Usage: talk %s\n", gettext("address [terminal]"));
74         exit(1);
75     }
76     if (!isatty(0)) {
77         fprintf(stderr,
78             gettext("Standard input must be a tty, not a pipe or a file\n"));
79         exit(1);
80     }

82     if (!isatty(1)) {
83         fprintf(stderr,
84             gettext("Standard output must be a tty, not a pipe or a file\n"));
85         exit(1);
86     }

88     if ((my_name = getlogin()) == NULL) {
89         struct passwd *pass = getpwuid(getuid());
90         if (pass != NULL)
91             my_name = pass->pw_name;
92     }
93     if (my_name == NULL) {
94         fprintf(stderr,
95             gettext("Who are you? You have no entry in /etc/utmp! Aborting..\n"));
96         exit(1);
97     }

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

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

103 /*
104  * check for, and strip out, the machine name of the target
105  */

107     for (ptr = argv[1]; *ptr != '\0' &&
108         *ptr != '@' &&
109         *ptr != ':' &&
110         *ptr != '!' &&
111         *ptr != '.'; ptr++) {
112     }

114     if (*ptr == '\0') {
116         /* this is a local to local talk */

118         rem_name = argv[1];
119         rem_machine_name = my_machine_name;

121     } 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
```

new/usr/src/cmd/cmd-inet/usr.sbin/ipadm/ipadm.c

1

```
*****
56618 Sun Jul 16 20:04:37 2017
new/usr/src/cmd/cmd-inet/usr.sbin/ipadm/ipadm.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2017 Nexenta Systems, Inc.
25  * Copyright 2017 Gary Mills
26 */

28 #include <arpa/inet.h>
29 #include <errno.h>
30 #include <getopt.h>
31 #include <inet/ip.h>
32 #include <inet/iptun.h>
33 #include <inet/tunables.h>
34 #include <libdladm.h>
35 #include <libdliptun.h>
36 #include <libdlink.h>
37 #include <libinetutil.h>
38 #include <libipadm.h>
39 #include <locale.h>
40 #include <netdb.h>
41 #include <netinet/in.h>
42 #include <ofmt.h>
43 #include <stdarg.h>
44 #include <stddef.h>
45 #include <stdio.h>
46 #include <stdlib.h>
47 #include <string.h>
48 #include <strings.h>
49 #include <sys/stat.h>
50 #include <sys/types.h>
51 #include <zone.h>

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

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

new/usr/src/cmd/cmd-inet/usr.sbin/ipadm/ipadm.c

2

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

67 typedef struct cmd {
68     char          *c_name;
69     cmdfunc_t      *c_fn;
70     const char      *c_usage;
71 } cmd_t;
    unchanged_portion_omitted

500 /*
501  * Print individual columns for the show-*prop subcommands.
502  */
503 static void
504 print_prop(show_prop_state_t *statep, uint_t flags, char *buf, size_t bufsize)
505 {
506     const char      *prop_name = statep->sps_pname;
507     char             *ifname = statep->sps_ifname;
508     char             *propval = statep->sps_propval;
509     uint_t           proto = statep->sps_proto;
510     size_t           proptype = MAXPROPVALLEN;
511     char             *object;
512     ipadm_status_t   status;

513     if (statep->sps_ifprop) {
514         status = ipadm_get_ifprop(iph, ifname, prop_name, propval,
515             &proptype, proto, flags);
516         object = ifname;
517     } else if (statep->sps_modprop) {
518         status = ipadm_get_prop(iph, prop_name, propval, &proptype,
519             proto, flags);
520         object = ipadm_proto2str(proto);
521     } else {
522         status = ipadm_get_addrprop(iph, prop_name, propval, &proptype,
523             statep->sps_aobjname, flags);
524         object = statep->sps_aobjname;
525     }

526     if (status != IPADM_SUCCESS) {
527         if ((status == IPADM_NOTFOUND && (flags & IPADM_OPT_PERSIST)) ||
528             status == IPADM_ENXIO) {
529             propval[0] = '\0';
530             goto cont;
531         }
532         statep->sps_status = status;
533         statep->sps_retstatus = status;
534         return;
535     }
536     cont:
537     statep->sps_status = IPADM_SUCCESS;
538     (void) snprintf(buf, bufsize, "%s", propval);
539 }
    unchanged_portion_omitted
```

new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_dhcpv6.c

1

```
*****
28730 Sun Jul 16 20:04:37 2017
new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_dhcpv6.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2017 Gary Mills
24  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

28 /*
29  * Dynamic Host Configuration Protocol version 6, for IPv6. Supports
30  * RFCs 3315, 3319, 3646, 3898, 4075, 4242, 4280, 4580, 4649, and 4704.
31 */

33 #include <ctype.h>
34 #include <stdio.h>
35 #include <stdlib.h>
36 #include <string.h>
37 #include <time.h>
38 #include <sys/types.h>
39 #include <sys/socket.h>
40 #include <netinet/in.h>
41 #include <netinet/dhcp6.h>
42 #include <arpa/inet.h>
43 #include <dhcp_impl.h>
44 #include <dhcp_inittab.h>

46 #include "snoop.h"

48 static const char *mtype_to_str(uint8_t);
49 static const char *option_to_str(uint8_t);
50 static const char *duidtype_to_str(uint16_t);
51 static const char *status_to_str(uint16_t);
52 static const char *entr_to_str(uint32_t);
53 static const char *reconf_to_str(uint8_t);
54 static const char *authproto_to_str(uint8_t);
55 static const char *authalg_to_str(uint8_t, uint8_t);
56 static const char *authrdm_to_str(uint8_t);
57 static const char *cwhat_to_str(uint8_t);
58 static const char *catype_to_str(uint8_t);
59 static void show_hex(const uint8_t *, int, const char *);
60 static void show_ascii(const uint8_t *, int, const char *);
61 static void show_address(const char *, const void *);
```

new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_dhcpv6.c

2

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

64 int
65 interpret_dhcpv6(int flags, const uint8_t *data, int len)
66 {
67     int olen = len;
68     char *line, *lstart;
69     dhcpv6_relay_t d6r;
70     dhcpv6_message_t d6m;
71     uint_t optlen;
72     uint16_t statuscode;

74     if (len <= 0) {
75         (void) strlcpy(get_sum_line(), "DHCPv6?", MAXLINE);
76         return (0);
77     }
78     if (flags & F_SUM) {
79         uint_t ias;
80         dhcpv6_option_t *d6o;
81         in6_addr_t link, peer;
82         char linkstr[INET6_ADDRSTRLEN];
83         char peerstr[INET6_ADDRSTRLEN];

85         line = lstart = get_sum_line();
86         line += snprintf(line, MAXLINE, "DHCPv6 %s",
87             mtype_to_str(data[0]));
88         if (data[0] == DHCPV6_MSG_RELAY_FORW ||
89             data[0] == DHCPV6_MSG_RELAY_REPL) {
90             if (len < sizeof (d6r)) {
91                 (void) strlcpy(line, "?",
92                     MAXLINE - (line - lstart));
93                 return (olen);
94             }
95             /* Not much in DHCPv6 is aligned. */
96             (void) memcpy(&d6r, data, sizeof (d6r));
97             (void) memcpy(&link, d6r.d6r_linkaddr, sizeof (link));
98             (void) memcpy(&peer, d6r.d6r_peeraddr, sizeof (peer));
99             line += snprintf(line, MAXLINE - (line - lstart),
100                 " HC=%d link=%s peer=%s", d6r.d6r_hop_count,
101                 inet_ntop(AF_INET6, &link, linkstr,
102                     sizeof (linkstr)),
103                 inet_ntop(AF_INET6, &peer, peerstr,
104                     sizeof (peerstr)));
105             data += sizeof (d6r);
106             len -= sizeof (d6r);
107         } else {
108             if (len < sizeof (d6m)) {
109                 (void) strlcpy(line, "?",
110                     MAXLINE - (line - lstart));
111                 return (olen);
112             }
113             (void) memcpy(&d6m, data, sizeof (d6m));
114             line += snprintf(line, MAXLINE - (line - lstart),
115                 " xid=%x", DHCPV6_GET_TRANSID(&d6m));
116             data += sizeof (d6m);
117             len -= sizeof (d6m);
118         }
119         ias = 0;
120         d6o = NULL;
121         while ((d6o = dhcpv6_find_option(data, len, d6o,
122             DHCPV6_OPT_IA_NA, NULL)) != NULL)
123             ias++;
124         if (ias > 0)
125             line += snprintf(line, MAXLINE - (line - lstart),
126                 " IAs=%u", ias);
127         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     /*
598      * Be very careful with negative numbers; ANSI signed/unsigned
599      * comparison doesn't work as expected.
600      */
601     while (len >= (signed)sizeof (d6o)) {
602         (void) memcpy(&d6o, data, sizeof (d6o));
603         d6o.d6o_code = ntohs(d6o.d6o_code);
604         d6o.d6o_len = olen = ntohs(d6o.d6o_len);
605         (void) snprintf(get_line(0, 0), get_line_remain(),
606             "Option Code = %u (%s)", d6o.d6o_code,
607             option_to_str(d6o.d6o_code));
608         ostart = data + sizeof (d6o);
609         len -= sizeof (d6o);
610         if (olen > len) {
611             (void) strlcpy(get_line(0, 0), "Option truncated",
612                 get_line_remain());
613             olen = len;
614         }
615         switch (d6o.d6o_code) {
616             case DHCPV6_OPT_CLIENTID:
617             case DHCPV6_OPT_SERVERID:
618                 if (olen < sizeof (val16))
619                     break;
620                 (void) memcpy(&val16, data, sizeof (val16));
621                 data += sizeof (val16);
622                 olen -= sizeof (val16);
623                 type = ntohs(val16);
624                 (void) snprintf(get_line(0, 0), get_line_remain(),
625                     " DUID Type = %u (%s)", type,
626                     duidtype_to_str(type));
627                 if (type == DHCPV6_DUID_LL || type == DHCPV6_DUID_LL) {
628                     if (olen < sizeof (val16))
629                         break;
630                     (void) memcpy(&val16, data, sizeof (val16));
631                     data += sizeof (val16);
632                     olen -= sizeof (val16);
633                     val16 = ntohs(val16);
634                     (void) snprintf(get_line(0, 0),
635                         get_line_remain(),
636                         " Hardware Type = %u (%s)", val16,
637                         arp_h_type(val16));
638                 }
639                 if (type == DHCPV6_DUID_LL) {
640                     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;
684
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;
703
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);
771         retlen = interpret_dhcpv6(F_DTAIL, data, olen);
772         prot_prefix = prot_nest_prefix;
773         prot_nest_prefix = oldnest;
774     }
775     break;
776 case DHCPV6_OPT_AUTH: {
777     dhcpv6_auth_t d6a;
778
779     if (olen < DHCPV6_AUTH_SIZE - sizeof (d6o))
780         break;
781     (void) memcpy(&d6a, data - sizeof (d6o),
782         DHCPV6_AUTH_SIZE);
783     data += DHCPV6_AUTH_SIZE - sizeof (d6o);
784     olen += DHCPV6_AUTH_SIZE - sizeof (d6o);

```



```

884      (void) snprintf(get_line(0, 0), get_line_remain(),
885        " Protocol = %u (%s)", d6a.d6a_proto,
886        authproto_to_str(d6a.d6a_proto));
887      (void) snprintf(get_line(0, 0), get_line_remain(),
888        " Algorithm = %u (%s)", d6a.d6a_alg,
889        authalg_to_str(d6a.d6a_proto, d6a.d6a_alg));
890      (void) snprintf(get_line(0, 0), get_line_remain(),
891        " Replay Detection Method = %u (%s)", d6a.d6a_rdm,
892        authrdm_to_str(d6a.d6a_rdm));
893      show_hex(d6a.d6a_replay, sizeof (d6a.d6a_replay),
894        " RDM Data");
895      if (olen > 0)
896        show_hex(data, olen, " Auth Info");
897      break;
898    }
899    case DHCPV6_OPT_UNICAST:
900      if (olen >= sizeof (in6_addr_t))
901        show_address(" Server Address", data);
902      break;
903    case DHCPV6_OPT_STATUS_CODE:
904      if (olen < sizeof (val16))
905        break;
906      (void) memcpy(&val16, data, sizeof (val16));
907      val16 = ntohs(val16);
908      (void) snprintf(get_line(0, 0), get_line_remain(),
909        " Status Code = %u (%s)", val16,
910        status_to_str(val16));
911      data += sizeof (val16);
912      olen -= sizeof (val16);
913      if (olen > 0)
914        (void) snprintf(get_line(0, 0),
915          get_line_remain(), " Text = \"%s\"",
916          olen, data);
917      break;
918    case DHCPV6_OPT_VENDOR_CLASS:
919      if (olen < sizeof (val32))
920        break;
921      (void) memcpy(&val32, data, sizeof (val32));
922      data += sizeof (val32);
923      olen -= sizeof (val32);
924      val32 = ntohl(val32);
925      (void) snprintf(get_line(0, 0), get_line_remain(),
926        " Enterprise Number = %lu (%s)", val32,
927        entr_to_str(val32));
928      /* FALLTHROUGH */
929    case DHCPV6_OPT_USER_CLASS:
930      while (olen >= sizeof (val16)) {
931        (void) memcpy(&val16, data, sizeof (val16));
932        data += sizeof (val16);
933        olen -= sizeof (val16);
934        val16 = ntohs(val16);
935        if (val16 > olen) {
936          (void) strncpy(get_line(0, 0),
937            " Truncated class",
938            get_line_remain());
939          val16 = olen;
940        }
941        show_hex(data, olen, " Class");
942        data += val16;
943        olen -= val16;
944      }
945      break;
946    case DHCPV6_OPT_VENDOR_OPT: {
947      dhcpv6_option_t sd6o;
948      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) strncpy(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;
912
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) strncpy(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 }

```

new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_dhcpv6.c

11

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

_____unchanged_portion_omitted_____

new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_rpcsec.c

1

```
*****
9701 Sun Jul 16 20:04:37 2017
new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_rpcsec.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 2005 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  * Copyright 2012 Milan Jurik. All rights reserved.
26  * Copyright 2017 Gary Mills
27 */

29 #include <sys/types.h>
30 #include <sys/errno.h>
31 #include <sys/tiuser.h>
32 #include <setjmp.h>

34 #include <rpc/types.h>
35 #include <rpc/xdr.h>
36 #include <rpc/auth.h>
37 #include <rpc/clnt.h>
38 #include <rpc/rpc_msg.h>
39 #include <rpc/rpcsec_gss.h>
40 #include <string.h>
41 #include "snoop.h"

43 extern jmp_buf xdr_err;

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

50 char *
51 rpcsec_gss_proc_to_string(unsigned int proc)
52 {
53     switch (proc) {
54     case RPCSEC_GSS_DATA:    return "RPCSEC_GSS_DATA"; break;
55     case RPCSEC_GSS_INIT:   return "RPCSEC_GSS_INIT"; break;
56     case RPCSEC_GSS_CONTINUE_INIT:
57         return ("RPCSEC_GSS_CONTINUE_INIT");
58     case RPCSEC_GSS_DESTROY:
59         return ("RPCSEC_GSS_DESTROY");
60     default:
61         return ("unknown");
62     }
63 }
```

new/usr/src/cmd/cmd-inet/usr.sbin/snoop/snoop_rpcsec.c

2

```
62     }
63 }
    unchanged_portion_omitted

290 /*
291  * Skip the header RPCSEC_GSS cred data and
292  * put service and control type in the xid cache.
293 */
294 void
295 extract_rpcsec_gss_cred_info(int xid)
296 {
297     unsigned int seq_num;
298     unsigned int handle_len;
299     unsigned int flavor_len;
300     unsigned int rpcsec_gss_ver;
301     rpc_gss_service_t rpcsec_gss_service;
302     unsigned int rpcsec_gss_proc;
303     struct cache_struct *x;

304     (void) getxdr_u_long();
305     flavor_len = getxdr_u_long();
306     rpcsec_gss_ver = getxdr_u_long();
307     /* see if we know this version or not */
308     if (rpcsec_gss_ver != 1) {
309         longjmp(xdr_err, 1);
310     }
311     rpcsec_gss_proc = getxdr_u_long();
312     (void) getxdr_u_long();
313     seq_num = getxdr_u_long();
314     rpcsec_gss_service = getxdr_enum();
315     /* skip the handle */
316     xdr_skip(RNDUP(getxdr_u_long()));

317     if (x = find_xid(xid)) {
318         x->xid_gss_service = rpcsec_gss_service;
319         x->xid_gss_proc = rpcsec_gss_proc;
320     }

322 /*
323  * Print the argument data for the RPCSEC_GSS_INIT control procedure.
324 */
325 static void
326 print_rpc_gss_init_arg(int flags, struct cache_struct *x)
327 {
328
329     char *line;
330     char *token, *line;
331     unsigned int token_len;
332     int pos = 0;

333     /*
334      * see if we need to print out the rpc_gss_init_arg structure
335      * or not.
336      */

337     if (x->xid_gss_proc != RPCSEC_GSS_INIT &&
338         x->xid_gss_proc != RPCSEC_GSS_CONTINUE_INIT) {
339         return;
340     }

341     /* print it */

342     (void) sprintf(get_line(pos, getxdr_pos()),
343         "RPCSEC_GSS_INIT args:");
344 }
```

```

348     pos = getxdr_pos();
349     token_len = getxdr_u_long();
350     (void) getxdr_hex(token_len);
351     token = getxdr_hex(token_len);
352     line = get_line(pos, getxdr_pos());
353     sprintf(line, "    gss token: length = %d, data = [%d bytes]",
354             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 {
364     char *handle, *line;
365     char *handle, *token, *line;
366     unsigned int token_len, handle_len;
367     unsigned int major, minor, seq_window;
368
369     int pos = 0;
370     struct cache_struct *x;
371
372     /* print it */
373
374     (void) sprintf(get_line(pos, getxdr_pos()), "RPCSEC_GSS_INIT result:");
375
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();
386
387     (void) sprintf(get_line(pos, getxdr_pos()),
388                   "    gss_major status = %u", major);
389
390     (void) sprintf(get_line(pos, getxdr_pos()),
391                   "    gss_minor status = %u", minor);
392
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);
399     line = get_line(pos, getxdr_pos());
400     sprintf(line, "    gss token: length = %d, data = [%d bytes]",
401             token_len, token_len);
402     show_trailer();
403 }

```

unchanged_portion_omitted

new/usr/src/cmd/gss/gssd/gssdtest.c

1

```
*****
53182 Sun Jul 16 20:04:37 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
```

new/usr/src/cmd/gss/gssd/gssdtest.c

2

```
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 */
```

new/usr/src/cmd/gss/gssd/gssdtest.c

3

```
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 Sun Jul 16 20:04:38 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/socket.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 }
55
56 unchanged_portion_omitted

114 static char *
115 pseudo_serialno_from_addr(char *name)
116 {
117     int sd, errnum;
118     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);
145     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 }
166
167 unchanged_portion_omitted

```


new/usr/src/cmd/hal/addons/storage/addon-storage.c

1

```
*****
10679 Sun Jul 16 20:04:38 2017
new/usr/src/cmd/hal/addons/storage/addon-storage.c
8485 Remove set but unused variables in usr/src/cmd
*****
1  /*****
2  *
3  *  addon-storage.c : watch removable media state changes
4  *
5  *  Copyright 2017 Gary Mills
6  *  Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
7  *  Use is subject to license terms.
8  *
9  *  Licensed under the Academic Free License version 2.1
10 *
11 *****/

13 #ifdef HAVE_CONFIG_H
14 # include <config.h>
15 #endif

17 #include <errno.h>
18 #include <string.h>
19 #include <strings.h>
20 #include <stdlib.h>
21 #include <stdio.h>
22 #include <sys/ioctl.h>
23 #include <sys/types.h>
24 #include <sys/stat.h>
25 #include <sys/types.h>
26 #include <sys/wait.h>
27 #include <fcntl.h>
28 #include <unistd.h>
29 #include <sys/mnttab.h>
30 #include <sys/dkio.h>
31 #include <priv.h>
32 #include <libsysevent.h>
33 #include <sys/sysevent/dev.h>

35 #include <libhal.h>

37 #include "../hal/logger.h"

39 #define SLEEP_PERIOD    5

41 static char          *udi;
42 static char          *devfs_path;
43 LibHalContext        *ctx = NULL;
44 static sysevent_handle_t *shp = NULL;

46 static void          sysevent_dev_handler(sysevent_t *);

48 static void
49 my_dbus_error_free(DBusError *error)
50 {
51     if (dbus_error_is_set(error)) {
52         dbus_error_free(error);
53     }
54 }

55 unchanged_portion_omitted

314 int
315 main (int argc, char *argv[])
316 {
317     char *device_file, *raw_device_file;
318     DBusError error;
```

new/usr/src/cmd/hal/addons/storage/addon-storage.c

2

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

324     if ((udi = getenv ("UDI")) == NULL)
325         goto out;
326     if ((device_file = getenv ("HAL_PROP_BLOCK_DEVICE")) == NULL)
327         goto out;
328     if ((raw_device_file = getenv ("HAL_PROP_BLOCK_SOLARIS_RAW_DEVICE")) ==
329         goto out;
330     if ((bus = getenv ("HAL_PROP_STORAGE_BUS")) == NULL)
331         goto out;
332     if ((drive_type = getenv ("HAL_PROP_STORAGE_DRIVE_TYPE")) == NULL)
333         goto out;
334     if ((devfs_path = getenv ("HAL_PROP_SOLARIS_DEVFS_PATH")) == NULL)
335         goto out;

337     drop_privileges ();

339     setup_logger ();

341     sysevent_init ();

344     support_media_changed_str = getenv ("HAL_PROP_STORAGE_CDROM_SUPPORT_MEDI
345     if (support_media_changed_str != NULL && strcmp (support_media_changed_s
346         support_media_changed = TRUE;
347     else
348         support_media_changed = FALSE;

343     dbus_error_init (&error);

345     if ((ctx = libhal_ctx_init_direct (&error)) == NULL) {
346         goto out;
347     }
348     my_dbus_error_free (&error);

350     if (!libhal_device_addon_is_ready (ctx, udi, &error)) {
351         goto out;
352     }
353     my_dbus_error_free (&error);

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

357     last_state = state = DKIO_NONE;

359     /* Linux version of this addon attempts to re-open the device 0_EXCL
360     * every 2 seconds, trying to figure out if some other app,
361     * like a cd burner, is using the device. Aside from questionable
362     * value of this (apps should use HAL's locked property or/and
363     * Solaris in_use facility), but also frequent opens/closes
364     * keeps media constantly spun up. All this needs more thought.
365     */
366     for (;;) {
367         if (is_mounted (device_file)) {
368             close_device (&fd);
369             sleep (SLEEP_PERIOD);
370         } else if ((fd < 0) && ((fd = open (raw_device_file, O_RDONLY |
371             HAL_DEBUG ("open failed for %s: %s", raw_device_file, s
372             sleep (SLEEP_PERIOD);
373         } else {
374             /* Check if a disc is in the drive */
375             /* XXX initial call always returns inserted
376             * 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                 state, de));
382             continue;
383         } else {
384             HAL_DEBUG (("new state %d %s", state, de));
385         }
386     }
387     switch (state) {
388     case DKIO_EJECTED:
389         HAL_DEBUG (("Media removal detected on %s",
390             de));
391         libhal_device_set_property_bool (ctx, ud,
392             "media-removed", 1, &error);
393         /* attempt to unmount all childs */
394         unmount_childs (ctx, udi);
395
396         /* could have a fs on the main block dev */
397         libhal_device_rescan (ctx, udi, &error);
398         my_dbus_error_free (&error);
399         break;
400
401     case DKIO_INSERTED:
402         HAL_DEBUG (("Media insertion detected on %s",
403             de));
404         libhal_device_set_property_bool (ctx, ud,
405             "media-inserted", 1, &error);
406
407         /* could have a fs on the main block dev */
408         libhal_device_rescan (ctx, udi, &error);
409         my_dbus_error_free (&error);
410         break;
411
412     case DKIO_DEV_GONE:
413         HAL_DEBUG (("Device gone detected on %s",
414             de));
415         unmount_childs (ctx, udi);
416         close_device (&fd);
417         goto out;
418
419     case DKIO_NONE:
420     default:
421         break;
422     }
423     } else {
424         HAL_DEBUG (("DKIOCSTATE failed: %s\n", strerror(
425             errno)));
426         sleep (SLEEP_PERIOD);
427     }
428 }
429
430 out:
431 if (ctx != NULL) {
432     sysevent_fini ();
433     libhal_ctx_shutdown (ctx, &error);
434     libhal_ctx_free (ctx);
435 }
436
437 return 0;
438 }

```

```

*****
39988 Sun Jul 16 20:04:38 2017
new/usr/src/cmd/ipf/tools/ipfcomp.c
8485 Remove set but unused variables in usr/src/cmd
*****
1 /*
2  * Copyright (C) 1993-2001 by Darren Reed.
3  *
4  * See the IPFILTER.LICENCE file for details on licencing.
5  *
6  * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
7  * Use is subject to license terms.
8  * Copyright 2017 Gary Mills
9  */

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

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

16 #include "ipf.h"

19 typedef struct {
20     int c;
21     int e;
22     int n;
23     int p;
24     int s;
25 } mc_t;

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

31 int intcmp __P((const void *, const void *));
32 static void indent __P((FILE *, int));
33 static void printeq __P((FILE *, char *, int, int, int));
34 static void printipeq __P((FILE *, char *, int, int, int));
35 static void addrule __P((FILE *, frent_t *));
36 static void printhooks __P((FILE *, int, int, frgroup_t *));
37 static void emitheader __P((frgroup_t *, u_int, u_int));
38 static void emitGroup __P((int, int, void *, frent_t *, char *,
39     u_int, u_int));
40 static void emittail __P((void));
41 static void printCgroup __P((int, frent_t *, mc_t *, char *));

43 #define FRC_IFN 0
44 #define FRC_V 1
45 #define FRC_P 2
46 #define FRC_FL 3
47 #define FRC_TOS 4
48 #define FRC_TTL 5
49 #define FRC_SRC 6
50 #define FRC_DST 7
51 #define FRC_TCP 8
52 #define FRC_SP 9
53 #define FRC_DP 10
54 #define FRC_OPT 11
55 #define FRC_SEC 12
56 #define FRC_ATH 13
57 #define FRC_IST 14
58 #define FRC_ICC 15
59 #define FRC_MAX 16

```

```

62 static FILE *cfile = NULL;

64 /*
65  * This is called once per filter rule being loaded to emit data structures
66  * required.
67  */
68 void printc(fr)
69 frent_t *fr;
70 {
72     fripf_t *ipf;
73     u_long *ulp;
74     char *and;
75     FILE *fp;
76     int i;

77     if (fr->fr_v != 4)
78         return;
79     if ((fr->fr_type != FR_T_IPF) && (fr->fr_type != FR_T_NONE))
80         return;
81     if ((fr->fr_type == FR_T_IPF) &&
82         ((fr->fr_datatype != FRI_NORMAL) || (fr->fr_satype != FRI_NORMAL)))
83         return;
84     ipf = fr->fr_ipf;

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

98         fprintf(fp, "#include <sys/types.h>\n");
99         fprintf(fp, "#include <sys/time.h>\n");
100        fprintf(fp, "#include <sys/socket.h>\n");
101        fprintf(fp, "#if !defined(__FreeBSD__) && !defined(__OpenBSD__)\n");
102        fprintf(fp, "# include <sys/system.h>\n");
103        fprintf(fp, "#endif\n");
104        fprintf(fp, "#include <sys/errno.h>\n");
105        fprintf(fp, "#include <sys/param.h>\n");
106        fprintf(fp, "#include <sys/rnd.h>\n");
107        fprintf(fp, "#if !defined(__SVR4) && !defined(__svr4__) && !defined(__hpux)\n");
108        fprintf(fp, "# include <sys/mbuf.h>\n");
109        fprintf(fp, "#endif\n");
110        fprintf(fp, "#include <sys/rnd.h>\n");
111        fprintf(fp, "#if defined(__FreeBSD__) && (__FreeBSD_version > 220000)\n");
112        fprintf(fp, "# include <sys/sockio.h>\n");
113        fprintf(fp, "#else\n");
114        fprintf(fp, "# include <sys/ioctl.h>\n");
115        fprintf(fp, "#endif /* FreeBSD */\n");
116        fprintf(fp, "#include <net/if.h>\n");
117        fprintf(fp, "#include <netinet/in.h>\n");
118        fprintf(fp, "#include <netinet/in_system.h>\n");
119        fprintf(fp, "#include <netinet/ip.h>\n");
120        fprintf(fp, "#include <netinet/tcp.h>\n");
121        fprintf(fp, "#include \"netinet/ip_compat.h\"\n");
122        fprintf(fp, "#include \"netinet/ip_fil.h\"\n");
123        fprintf(fp, "#include \"netinet/ip_rules.h\"\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

new/usr/src/cmd/ipf/tools/ippool.c

1

18031 Sun Jul 16 20:04:38 2017

new/usr/src/cmd/ipf/tools/ippool.c

8485 Remove set but unused variables in usr/src/cmd

```
1 /*
2  * Copyright (C) 2003 by Darren Reed.
3  *
4  * See the IPFILTER.LICENCE file for details on licencing.
5  *
6  * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
7  * Use is subject to license terms.
8  *
9  * Copyright (c) 2014, Joyent, Inc. All rights reserved.
10 * Copyright 2017 Gary Mills
11 */

13 #include <sys/types.h>
14 #include <sys/time.h>
15 #include <sys/param.h>
16 #include <sys/socket.h>
17 #if defined(BSD) && (BSD >= 199306)
18 # include <sys/cdefs.h>
19 #endif
20 #include <sys/ioctl.h>

22 #include <net/if.h>
23 #if __FreeBSD_version >= 300000
24 # include <net/if_var.h>
25 #endif
26 #include <netinet/in.h>

28 #include <arpa/inet.h>

30 #include <stdio.h>
31 #include <fcntl.h>
32 #include <stdlib.h>
33 #include <string.h>
34 #include <netdb.h>
35 #include <ctype.h>
36 #include <unistd.h>
37 #include <nlist.h>

39 #include "ipf.h"
40 #include "netinet/ipl.h"
41 #include "netinet/ip_lookup.h"
42 #include "netinet/ip_pool.h"
43 #include "netinet/ip_htable.h"
44 #include "kmem.h"
45 #include "ipfzone.h"

47 extern int ippool_yyparse __P((void));
48 extern int ippool_yydebug;
49 extern FILE *ippool_yyin;
50 extern char *optarg;
51 extern int lineNum;

53 void showpools __P((ip_pool_stat_t *));
54 void usage __P((char *));
55 int main __P((int, char **));
56 int poolcommand __P((int, int, char *[]));
57 int poolnodecommand __P((int, int, char *[]));
58 int loadpoolfile __P((int, char *[], char *));
59 int poollist __P((int, char *[]));
60 int poolflush __P((int, char *[]));
61 int poolstats __P((int, char *[]));
```

new/usr/src/cmd/ipf/tools/ippool.c

2

```
62 int gettype __P((char *, u_int *));
63 int getrole __P((char *));
64 void poollist_dead __P((int, char *, int, char *, char *));
65 void showpools_live(int, int, ip_pool_stat_t *, char *, int);
66 void showhashs_live(int, int, iphtstat_t *, char *, int);

68 int opts = 0;
69 int fd = -1;
70 int use_inet6 = 0;

73 void usage(prog)
74 char *prog;
75 {
76     const char *zoneopt = "[-G|-z zonename] ";
77     fprintf(stderr, "Usage: %s\n", prog);
78     fprintf(stderr, "\t\t\t-a [-dnv] %s[-m <name>] [-o <role>] -i <ipaddr>[/
79     zoneopt);
80     fprintf(stderr, "\t\t\t-A [-dnv] %s[-m <name>] [-o <role>] [-S <seed>] [
81     zoneopt);
82     fprintf(stderr, "\t\t\t-f <file> %s[-dnuv]\n", zoneopt);
83     fprintf(stderr, "\t\t\t-F [-dv] %s[-o <role>] [-t <type>]\n", zoneopt);
84     fprintf(stderr, "\t\t\t-l [-dv] %s[-m <name>] [-t <type>]\n", zoneopt);
85     fprintf(stderr, "\t\t\t-r [-dnv] %s[-m <name>] [-o <role>] -i <ipaddr>[/
86     zoneopt);
87     fprintf(stderr, "\t\t\t-R [-dnv] %s[-m <name>] [-o <role>] [-t <type>]\n
88     zoneopt);
89     fprintf(stderr, "\t\t\t-s [-dtv] %s[-M <core>] [-N <namelist>]\n",
90     zoneopt);
91     exit(1);
92 }

unchanged_portion_omitted

380 int poollist(argc, argv)
381 int argc;
382 char *argv[];
383 {
384     char *kernel, *core, *poolname;
385     int c, role, type, live_kernel;
386     ip_pool_stat_t *plstp;
387     ip_pool_stat_t plstat;
388     iphtstat_t htstat;
389     ip_pool_stat_t *plstp, plstat;
390     iphtstat_t *htstp, htstat;
391     iphtable_t *hptr;
392     iplookupop_t op;
393     ip_pool_t *ptr;

393     core = NULL;
394     kernel = NULL;
395     live_kernel = 1;
396     type = IPLT_ALL;
397     poolname = NULL;
398     role = IPL_LOGALL;

400     while ((c = getopt(argc, argv, "dG:m:M:N:o:Rt:vz:")) != -1)
401     switch (c)
402     {
403     case 'd' :
404         opts |= OPT_DEBUG;
405         break;
406     case 'G' :
407         setzonename_global(optarg);
408         break;
409     case 'm' :
```

```

410         poolname = optarg;
411         break;
412     case 'M' :
413         live_kernel = 0;
414         core = optarg;
415         break;
416     case 'N' :
417         live_kernel = 0;
418         kernel = optarg;
419         break;
420     case 'o' :
421         role = getrole(optarg);
422         if (role == IPL_LOGNONE) {
423             fprintf(stderr, "unknown role '%s'\n", optarg);
424             return -1;
425         }
426         break;
427     case 'R' :
428         opts |= OPT_NORESOLVE;
429         break;
430     case 't' :
431         type = gettype(optarg, NULL);
432         if (type == IPLT_NONE) {
433             fprintf(stderr, "unknown type '%s'\n", optarg);
434             return -1;
435         }
436         break;
437     case 'v' :
438         opts |= OPT_VERBOSE;
439         break;
440     case 'z' :
441         setzone(optarg);
442         break;
443     }
444
445     if (opts & OPT_DEBUG)
446         fprintf(stderr, "poollist: opts = %#x\n", opts);
447
448     if (!(opts & OPT_DONOTHING) && (fd == -1)) {
449         fd = open(IPLLOOKUP_NAME, O_RDWR);
450         if (fd == -1) {
451             perror("open(IPLLOOKUP_NAME)");
452             exit(1);
453         }
454     }
455     if (setzone(fd) != 0) {
456         close(fd);
457         exit(1);
458     }
459 }
460
461 bzero((char *)&op, sizeof(op));
462 if (poolname != NULL) {
463     strncpy(op.iplo_name, poolname, sizeof(op.iplo_name));
464     op.iplo_name[sizeof(op.iplo_name) - 1] = '\0';
465 }
466 op.iplo_unit = role;
467
468 if (live_kernel == 0) {
469     poollist_dead(role, poolname, type, kernel, core);
470     return (0);
471 }
472
473 if (type == IPLT_ALL || type == IPLT_POOL) {
474     plstp = &plstat;

```

```

475     op.iplo_size = sizeof(plstat);
476     op.iplo_struct = &plstat;
477     op.iplo_name[0] = '\0';
478     op.iplo_arg = 0;
479
480     if (role != IPL_LOGALL) {
481         op.iplo_unit = role;
482
483         c = ioctl(fd, SIOCLLOOKUPSTAT, &op);
484         if (c == -1) {
485             perror("ioctl(SIOCLLOOKUPSTAT)");
486             return -1;
487         }
488     }
489     showpools_live(fd, role, &plstat, poolname, opts);
490 } else {
491     for (role = 0; role <= IPL_LOGMAX; role++) {
492         op.iplo_unit = role;
493
494         c = ioctl(fd, SIOCLLOOKUPSTAT, &op);
495         if (c == -1) {
496             perror("ioctl(SIOCLLOOKUPSTAT)");
497             return -1;
498         }
499         showpools_live(fd, role, &plstat, poolname, opts);
500     }
501     role = IPL_LOGALL;
502 }
503
504 if (type == IPLT_ALL || type == IPLT_HASH) {
505     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;
511
512     if (role != IPL_LOGALL) {
513         op.iplo_unit = role;
514
515         c = ioctl(fd, SIOCLLOOKUPSTAT, &op);
516         if (c == -1) {
517             perror("ioctl(SIOCLLOOKUPSTAT)");
518             return -1;
519         }
520     }
521     showhashs_live(fd, role, &htstat, poolname, opts);
522 } else {
523     for (role = 0; role <= IPL_LOGMAX; role++) {
524         op.iplo_unit = role;
525         c = ioctl(fd, SIOCLLOOKUPSTAT, &op);
526         if (c == -1) {
527             perror("ioctl(SIOCLLOOKUPSTAT)");
528             return -1;
529         }
530     }
531     showhashs_live(fd, role, &htstat, poolname, opts);
532 }
533
534 }
535 return 0;
536 }
537 }

```

unchanged portion omitted

```

641 int poolstats(argc, argv)
642 int argc;
643 char *argv[];
644 {
645     int c, type, role;
646     int c, type, role, live_kernel;
647     ip_pool_stat_t plstat;
648     char *kernel, *core;
649     iphtstat_t htstat;
650     iplookupop_t op;
651
652     core = NULL;
653     kernel = NULL;
654     live_kernel = 1;
655     type = IPLT_ALL;
656     role = IPL_LOGALL;
657
658     bzero((char *)&op, sizeof(op));
659
660     while ((c = getopt(argc, argv, "dG:M:N:o:t:vz:")) != -1)
661     {
662         switch (c)
663         {
664             case 'd' :
665                 opts |= OPT_DEBUG;
666                 break;
667             case 'G' :
668                 setzonename_global(optarg);
669                 break;
670             case 'M' :
671                 live_kernel = 0;
672                 core = optarg;
673                 break;
674             case 'N' :
675                 live_kernel = 0;
676                 kernel = optarg;
677                 break;
678             case 'o' :
679                 role = getrole(optarg);
680                 if (role == IPL_LOGNONE) {
681                     fprintf(stderr, "unknown role '%s'\n", optarg);
682                     return -1;
683                 }
684                 break;
685             case 't' :
686                 type = gettype(optarg, NULL);
687                 if (type != IPLT_POOL) {
688                     fprintf(stderr,
689                         "-s not supported for this type yet\n");
690                     return -1;
691                 }
692                 break;
693             case 'v' :
694                 opts |= OPT_VERBOSE;
695                 break;
696             case 'z' :
697                 setzonename(optarg);
698                 break;
699         }
700     }
701
702     if (opts & OPT_DEBUG)
703         fprintf(stderr, "poolstats: opts = %x\n", opts);
704
705     if (!(opts & OPT_DONOTHING) && (fd == -1)) {
706         fd = open(IPLOOKUP_NAME, O_RDWR);
707         if (fd == -1) {
708             perror("open(IPLOOKUP_NAME)");
709         }
710     }

```

```

698         exit(1);
699     }
700
701     if (setzone(fd) != 0) {
702         close(fd);
703         exit(1);
704     }
705
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, SIOCLLOOKUPSTAT, &op);
712             if (c == -1) {
713                 perror("ioctl(SIOCLLOOKUPSTAT)");
714                 return -1;
715             }
716             printf("Pools:\t%lu\n", plstat.ipls_pools);
717             printf("Nodes:\t%lu\n", plstat.ipls_nodes);
718         }
719     }
720
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, SIOCLLOOKUPSTAT, &op);
727             if (c == -1) {
728                 perror("ioctl(SIOCLLOOKUPSTAT)");
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 }
738 }

```

unchanged_portion_omitted

7480 Sun Jul 16 20:04:38 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"
2 /*
3 */
4
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 */
31
32 /*
33 * Miscellaneous functions for managing the string and integer lists
34 */
35
36 #include <k5-int.h>
37 #include "kdb5_ldap_list.h"
38
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;
45
46     if (list == NULL)
47         return 0;
48
49     for (i = 0; *list != NULL; list++) {
50         i++;
51     }
52
53     return i;
54 }

```

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;
165     int found = 0;
166
167     if ((destlist == NULL) || (*destlist == NULL) || (sourcelist == NULL))
168         return;
169
170     /* We need to add every entry present in the source list to
171      * the destination list */
172     if (mode == LIST_MODE_ADD) {
173         /* Traverse through the end of destlist for appending */
174         for (dlist = *destlist, dcount = 0; *dlist != NULL;
175             dlist++, dcount++) {
176             ; /* NULL statement */
177         }
178         /* Count the number of entries in the source list */
179         for (slist = sourcelist, scount = 0; *slist != NULL;
180             slist++, scount++) {
181             ; /* NULL statement */
182         }
183         /* Reset the slist pointer to the start of source list */
184         slist = sourcelist;
185
186         /* Now append the source list to the existing destlist */
187         if ((dcount + scount) < MAX_LIST_ENTRIES)
188             copycount = scount;
189         else
190             /* Leave the last entry for list terminator(=NULL) */
191             copycount = (MAX_LIST_ENTRIES - 1) - dcount;
192
193         memcpy(dlist, slist, (sizeof(char *) * copycount));
194         dlist += copycount;
195         *dlist = NULL;
196     } else if (mode == LIST_MODE_DELETE) {
197         /* We need to delete every entry present in the source list
198          * from the destination list */
199         for (slist = sourcelist; *slist != NULL; slist++) {
200             for (dlist = *destlist; *dlist != NULL; dlist++) {
201                 found = 0; /* value not found */
202                 /* DN is case insensitive string */
203                 if (strcasecmp(*dlist, *slist) == 0) {
204                     found = 1;
205                     free(*dlist);
206                     /* Advance the rest of the entries by one */
207                     for (tmplist = dlist; *tmplist != NULL; tmplist++) {
208                         *tmplist = *(tmplist+1);
209                     }
210                     break;
211                 }
212             }
213         }
214     }
215
216     return;
217 }

```

unchanged_portion_omitted_

new/usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.c

1

```
*****
80663 Sun Jul 16 20:04:38 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 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
57 */

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

new/usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.c

2

```
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 *progrname;
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 CHANGE_PW_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_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_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->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_POSTDATED);
194         else if (*argv[*i] == '-')
195             rparams->tktf_flags |= KRB5_KDB_DISALLOW_POSTDATED;
196         else
197             goto err_usage;

199         mask |= LDAP_REALM_KRB5TICKETFLAGS;
200     } else if (!strcmp((argv[*i] + 1), "allow_forwardable")) {
201         if (*argv[*i] == '+')
202             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_FORWARDABLE);

204         else if (*argv[*i] == '-')
205             rparams->tktf_flags |= KRB5_KDB_DISALLOW_FORWARDABLE;
206         else

```

```

207         goto err_usage;

209         mask |= LDAP_REALM_KRB5TICKETFLAGS;
210     } else if (!strcmp((argv[*i] + 1), "allow_renewable")) {
211         if (*argv[*i] == '+')
212             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_RENEWABLE);
213         else if (*argv[*i] == '-')
214             rparams->tktf_flags |= KRB5_KDB_DISALLOW_RENEWABLE;
215         else
216             goto err_usage;

218         mask |= LDAP_REALM_KRB5TICKETFLAGS;
219     } else if (!strcmp((argv[*i] + 1), "allow_proxiable")) {
220         if (*argv[*i] == '+')
221             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_PROXIABLE);
222         else if (*argv[*i] == '-')
223             rparams->tktf_flags |= KRB5_KDB_DISALLOW_PROXIABLE;
224         else
225             goto err_usage;

227         mask |= LDAP_REALM_KRB5TICKETFLAGS;
228     } else if (!strcmp((argv[*i] + 1), "allow_dup_skey")) {
229         if (*argv[*i] == '+')
230             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_DUP_SKEY);
231         else if (*argv[*i] == '-')
232             rparams->tktf_flags |= KRB5_KDB_DISALLOW_DUP_SKEY;
233         else
234             goto err_usage;

236         mask |= LDAP_REALM_KRB5TICKETFLAGS;
237     }

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

247         mask |= LDAP_REALM_KRB5TICKETFLAGS;
248     } else if (!strcmp((argv[*i] + 1), "requires_hwauth")) {
249         if (*argv[*i] == '+')
250             rparams->tktf_flags |= KRB5_KDB_REQUIRES_HW_AUTH;
251         else if (*argv[*i] == '-')
252             rparams->tktf_flags |= (int)(~KRB5_KDB_REQUIRES_HW_AUTH);
253         else
254             goto err_usage;

256         mask |= LDAP_REALM_KRB5TICKETFLAGS;
257     } else if (!strcmp((argv[*i] + 1), "allow_svr")) {
258         if (*argv[*i] == '+')
259             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_SVR);
260         else if (*argv[*i] == '-')
261             rparams->tktf_flags |= KRB5_KDB_DISALLOW_SVR;
262         else
263             goto err_usage;

265         mask |= LDAP_REALM_KRB5TICKETFLAGS;
266     } else if (!strcmp((argv[*i] + 1), "allow_tgs_req")) {
267         if (*argv[*i] == '+')
268             rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_TGT_BASED);
269         else if (*argv[*i] == '-')
270             rparams->tktf_flags |= KRB5_KDB_DISALLOW_TGT_BASED;
271         else
272             goto err_usage;

```

```
274     mask |= LDAP_REALM_KRB5_TICKETFLAGS;
275 } else if (!strcmp((argv[*i] + 1), "allow_tix")) {
276     if (*(argv[*i]) == '+')
277         rparams->tktf_flags |= (int)(~KRB5_KDB_DISALLOW_ALL_TIX);
278     else if (*(argv[*i]) == '-')
279         rparams->tktf_flags |= KRB5_KDB_DISALLOW_ALL_TIX;
280     else
281         goto err_usage;

283     mask |= LDAP_REALM_KRB5_TICKETFLAGS;
284 } else if (!strcmp((argv[*i] + 1), "needchange")) {
285     if (*(argv[*i]) == '+')
286         rparams->tktf_flags |= KRB5_KDB_REQUIRES_PWCHANGE;
287     else if (*(argv[*i]) == '-')
288         rparams->tktf_flags |= (int)(~KRB5_KDB_REQUIRES_PWCHANGE);
289     else
290         goto err_usage;

292     mask |= LDAP_REALM_KRB5_TICKETFLAGS;
293 } else if (!strcmp((argv[*i] + 1), "password_changing_service")) {
294     if (*(argv[*i]) == '+')
295         rparams->tktf_flags |= KRB5_KDB_PWCHANGE_SERVICE;
296     else if (*(argv[*i]) == '-')
297         rparams->tktf_flags |= (int)(~KRB5_KDB_PWCHANGE_SERVICE);
298     else
299         goto err_usage;

301     mask |= LDAP_REALM_KRB5_TICKETFLAGS;
302 }
303 err_usage:
304     print_usage = TRUE;

305 err_nomsg:
306     no_msg = TRUE;

307     return mask;
308 }
_____unchanged_portion_omitted_____
```

new/usr/src/cmd/ldap/common/fileurl.c

1

```
*****
13040 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/ldap/common/fileurl.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  * The contents of this file are subject to the Netscape Public
9  * License Version 1.1 (the "License"); you may not use this file
10 * except in compliance with the License. You may obtain a copy of
11 * the License at http://www.mozilla.org/NPL/
12 *
13 * Software distributed under the License is distributed on an "AS
14 * IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or
15 * implied. See the License for the specific language governing
16 * rights and limitations under the License.
17 *
18 * The Original Code is Mozilla Communicator client code, released
19 * March 31, 1998.
20 *
21 * The Initial Developer of the Original Code is Netscape
22 * Communications Corporation. Portions created by Netscape are
23 * Copyright (C) 1998-1999 Netscape Communications Corporation. All
24 * Rights Reserved.
25 *
26 * Contributor(s):
27 */

29 /*
30 * LDAP tools fileurl.c -- functions for handling file URLs.
31 * Used by ldapmodify.
32 */

34 #include "ldaptool.h"
35 #include "fileurl.h"
36 #include <ctype.h> /* for isalpha() */
37 #ifdef SOLARIS_LDAP_CMD
38 #include <locale.h>
39 #endif /* SOLARIS_LDAP_CMD */

41 #ifndef SOLARIS_LDAP_CMD
42 #define gettext(s) s
43 #endif

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

52 /*
53  * Convert a file URL to a local path.
54  *
55  * If successful, LDAPTOOL_FILEURL_SUCCESS is returned and *localpathp is
56  * set point to an allocated string. If not, an different LDAPTOOL_FILEURL_
57  * error code is returned.
58  *
59  * See RFCs 1738 and 2396 for a specification for file URLs... but
60  * Netscape Navigator seems to be a bit more lenient in what it will
61  * accept, especially on Windows).
```

new/usr/src/cmd/ldap/common/fileurl.c

2

```
62 *
63 * This function parses file URLs of these three forms:
64 *
65 *   file:///path
66 *   file:/path
67 *   file://localhost/path
68 *   file://host/path           (rejected with a ...NONLOCAL error)
69 *
70 * On Windows, we convert leading drive letters of the form C| to C:
71 * and if a drive letter is present we strip off the slash that precedes
72 * path. Otherwise, the leading slash is returned.
73 */
74 int
75 ldaptool_fileurl2path( const char *fileurl, char **localpathp )
76 {
77     const char *path;
78     char *pathcopy;
79
81     /*
82      * Make sure this is a file URL we can handle.
83      */
84     if ( !str_starts_with( fileurl, "file:" ) ) {
85         return( LDAPTOOL_FILEURL_NOTAFILEURL );
86     }
87
88     path = fileurl + 5; /* skip past "file:" scheme prefix */
89
90     if ( *path != '/' ) {
91         return( LDAPTOOL_FILEURL_MISSINGPATH );
92     }
93
94     ++path; /* skip past '/' at end of "file:/" */
95
96     if ( *path == '/' ) {
97         ++path; /* remainder is now host/path or /path */
98         if ( *path != '/' ) {
99             /*
100              * Make sure it is for the local host.
101              */
102             if ( str_starts_with( path, "localhost/" ) ) {
103                 path += 9;
104             } else {
105                 return( LDAPTOOL_FILEURL_NONLOCAL );
106             }
107         }
108     } else { /* URL is of the form file:/path */
109         --path;
110     }
111
112     /*
113      * The remainder is now of the form /path. On Windows, skip past the
114      * leading slash if a drive letter is present.
115      */
116     #ifndef _WINDOWS
117     if ( isalpha( path[1] ) && ( path[2] == '|' || path[2] == ':' ) ) {
118         ++path;
119     }
120     #endif /* _WINDOWS */
121
122     /*
123      * Duplicate the path so we can safely alter it.
124      * Unescape any %HH sequences.
125      */
126     if ( ( pathcopy = strdup( path ) ) == NULL ) {
127         return( LDAPTOOL_FILEURL_NOMEMORY );
128     }
```

```

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;
139
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

```

```
new/usr/src/cmd/lp/lib/papi/ppd.c
```

4114 Sun Jul 16 20:04:39 2017

```
new/usr/src/cmd/lp/lib/papi/ppd.c
```

```
new/usr/src/cmd; cp -tlib; pap1; ppap1
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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22  * Copyright 2017 Gary Mills
23  * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

```

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

```

27 /*
28  * This file contains an extremely rudimentary implementation of PPD file
29  * parsing support. The parsing done here converts the contents of a PPD
30  * file into a set of PAPI attributes that applications can use to build
31  * print panels.
32  */

```

```
34 #include <stdio.h>
35 #include <ctype.h>
36 #include <string.h>
37 #include <papi.h>
```

```
39 static void
40 process_line(char *line, char **key, char **value, char **comment)
```

```
42     char *ptr, *ptr2;
```

```

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

```

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

```

51      /*
52      *   line is in the form:
53      *   *key: value/comment
54      *   or
55      *   *key value/comment: data
56      */
57      *ptr++ = NULL;
58      while (isspace(*ptr) != 0)
59          ptr++;

```

1

```
new/usr/src/cmd/lp/lib/papi/ppd.c
```

```

61     if ((ptr2 = strchr(line, ' ')) != NULL) {
62         ptr = ptr2;
63         /*
64          * line is in the form:
65          *      *key value/comment: data
66          */
67         *ptr++ = NULL;
68         while (*ptr == ' ')
69             ptr++;
70     }

```

```
72         if (*ptr == '*')
73             ptr++;
```

```
75      *value = ptr;
```

```

77     if ((ptr = strchr(ptr, '/')) != NULL) {
78         *ptr++ = NULL;
79         *comment = ptr;
80     }
81 }

```

```

83 papi_status_t
84 PPDFileToAttributesList(papi_attribute_t ***attributes, char *filename)
85 {
86     papi_status_t status = PAPI_OK;
87     FILE *fp;
88     char line[256];
89     char capability[256];
90     char def[256];
91     char supported[256];
92     char *current_group_name = NULL;

```

```
93     int ui = 0;
```

```

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

```

```

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

```

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

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

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

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

```
114         if ((strcasecmp(key, "PageSize") == 0) ||
115             (strcasecmp(key, "InputSlot") == 0))
116             key = "media";
```

```
118         if (strcasecmp(key, "OpenGroup") == 0) {
119             if (value == NULL)
120                 value = "unknown":
```

```

120         current_group_name = group_name(value),
121     } else if (strcasecmp(key, "OpenUI") == 0) {
122         if ((strcasecmp(value, "PageSize") == 0) ||
123             (strcasecmp(value, "InputSlot") == 0))

```

2

```
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);
159
160 return (status);
161 }
```

unchanged_portion_omitted

new/usr/src/cmd/mail/copylet.c

1

```
*****
9763 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/mail/copylet.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 2005 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

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

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

31 #include "mail.h"

32 /*
33  NAME
34      copylet - copy a given letter to a file pointer
35
36  SYNOPSIS
37      int copylet(int letnum, FILE *f, int type)
38
39  DESCRIPTION
40      Copylet() will copy the letter "letnum" to the
41      given file pointer.
42
43      letnum  -> index into: letter table
44      f       -> file pointer to copy file to
45      type    -> copy type
46
47      Returns TRUE on a completely successful copy.
48 */
49
50 int
51 copylet(int letnum, FILE *f, int type)
52 {
53     int        pos = ftell(f);
54     int        rc = xxxcopylet(letnum, f, type);
55
56     if (fflush(f) != 0)
57         rc = FALSE;
58
59 }
```

new/usr/src/cmd/mail/copylet.c

2

```
60 /*
61  * On error, truncate the file to its original position so that a
62  * partial message is not left in the mailbox.
63  */
64 if (rc == FALSE)
65     ftruncate(fileno(f), pos);
66
67 return(rc);
68 }

69 int
70 xxxcopylet(int letnum, FILE *f, int type)
71 {
72     static char    pn[] = "copylet";
73     char          buf[LSIZE], lastc;
74     char          wbuf[LSIZE];
75     int           n;
76     long          l, k;
77     int           num;
78     int           rtncont = 1; /* True: nondelivery&content included, or regula
79     int           suppress = FALSE;
80     int           sav_suppress = FALSE; /* Did we suppress previous hdr line? */
81     int           print_from_struct = FALSE; /* print from hdrlines struct */
82     int           /* rather than fgets() buffer */
83
84     int           pushrest = FALSE;
85     int           ctf = FALSE;
86     int           didafflines = FALSE; /* Did we already put out any */
87     /* H_AFWDFROM lines? */
88     int           didrcvlines = FALSE; /* Did we already put out any */
89     /* H_RECEIVED lines? */
90     long          clen = -1L;
91     int           htype; /* header type */
92     int           sav_htype; /* Header type of last non-H_CONT header line */
93     struct hdrs *hptr;
94
95     if (!sending) {
96         /* Clear out any saved header info from previous message */
97         clr_hinfo();
98     }
99
100     fseek(tmpf, let[letnum].adr, 0);
101     /* Get size of message as stored into tempfile by copymt() */
102     k = let[letnum+1].adr - let[letnum].adr;
103     Dout(pn, 1, "(letnum = %d, type = %d), k = %ld\n", letnum, type, k);
104     while (k > 0) { /* process header */
105         num = ((k < sizeof(buf)) ? k+1 : sizeof(buf));
106         if (fgets(buf, num, tmpf) == NULL) {
107             return (FALSE);
108         }
109         if ((n = strlen(buf)) == 0) {
110             k = 0;
111             break;
112         }
113         k -= n;
114         lastc = buf[n-1];
115         if (pushrest) {
116             pushrest = (lastc != '\n');
117             continue;
118         }
119         htype = isheader(buf, &ctf);
120         Dout(pn, 5, "loop 1: buf = %s, htype= %d/%s\n", buf, htype, head
121         if (htype == H_CLEN) {
122             if (!sending) {
123                 savehdrs(buf, htype);
124             }
125             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             break;
166
167         case TTY:
168         case ORDINARY:
169         default:
170             if (fprintf(f, "\n") < 0)
171             {
172                 sav_errno = errno;
173                 return(FALSE);
174             }
175             break;
176     }
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     }
192 }
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];
289
290 if (pushrest) {
291     pushrest = (lastc != '\n');
292     continue;
293 }
294 sav_suppress = suppress;
295 suppress = FALSE;
296 print_from_struct = FALSE;
297 sav_htype = htype;
298 htype = isheader (buf, &ctf);
299 Dout(pn, 5, "loop 2: buf = %s, htype= %d/%s\n", buf, hty
300 /* The following order is defined in the MTA documents.
301 switch (htype) {
302 case H_CONT:
303     if (sending) {
304         suppress = sav_suppress;
305     }
306     continue;
307 case H_TCOPY:
308 case H_MIMEVERS:
309 case H_CTYPE:
310 case H_CLEN:
311     if (!sending) {
312         savehdrs(buf, htype);
313     }
314     hptra = hdrlines[htype].head;
315     if (htype == H_CLEN) {
316         clen = atol (hptra->value);
317     }
318     /*
319     * Use values saved in hdrlines[] structure
320     * rather than what was read from tmp file.
321     */
322     print_from_struct = TRUE;

```

```

322         /* FALLTHROUGH */
323     case H_EOH:
324     case H_AFWDFROM:
325     case H_AFWDCT:
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 }
336
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 }
344
345 Dout(pn, 3, "let[%d].text = %s\n", letnum, (let[letnum].text ? "TRUE" : "FALSE"));
346
347 if ((type == TTY) && (let[letnum].text == FALSE) && !pfl
348     if (fprintf (f, "\n%s\n", binmsg) < 0)
349     {
350         sav_errno = errno;
351         return(FALSE);
352     }
353     return (TRUE);
354 }
355
356 if (n == 1 && buf[0] == '\n') {
357     n = 0;
358 }
359 break;
360 }
361
362 Dout(pn, 1, "header processed, clen/k/n = %ld/%ld/%d\n", clen, k, n);
363
364 if (clen >= 0) {
365     if (((clen - n) == k) || ((clen - n) == (k - 1))) {
366         k = clen - n;
367     } else {
368         /* probable content-length mismatch. show it ALL! */
369         Dout(pn, 1, "clen conflict. using k = %ld\n", k);
370     }
371 }
372
373 /* copy balance of message */
374 if (rtrncont)
375     while (k > 0) {
376         num = ((k < sizeof(buf)) ? k : sizeof(buf));
377         if ((n = fread (buf, 1, num, tmpf)) <= 0) {
378             Dout(pn, 1, "content-length mismatch. return(FAL
379             return(FALSE);
380         }
381         k -= n;
382         if (fwrite(buf, 1, n, f) != n) {
383             sav_errno = errno;
384             return(FALSE);
385         }
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
```

new/usr/src/cmd/print/bsd-sysv-commands/disable.c

1

```
*****
3827 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/print/bsd-sysv-commands/disable.c
8485 Remove set but unused variables in usr/src/cmd
*****

2 /*
3  * CDDL HEADER START
4  *
5  * The contents of this file are subject to the terms of the
6  * Common Development and Distribution License (the "License").
7  * You may not use this file except in compliance 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  */

23 /*
24  * Copyright 2017 Gary Mills
25  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
26  * Use is subject to license terms.
27  *
28  */

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

33 #include <stdio.h>
34 #include <stdlib.h>
35 #include <unistd.h>
36 #include <string.h>
37 #include <locale.h>
38 #include <libintl.h>
39 #include <papi.h>
40 #include "common.h"

42 static void
43 usage(char *program)
44 {
45     char *name;

47     if ((name = strrchr(program, '/')) == NULL)
48         name = program;
49     else
50         name++;

52     fprintf(stdout,
53             gettext("Usage: %s [-c] [-W] [-r reason] destination ...\n"),
54             name);
55     exit(1);
56 }

unchanged_portion_omitted

88 int
89 main(int ac, char *av[])
90 {
```

new/usr/src/cmd/print/bsd-sysv-commands/disable.c

2

```
91     papi_status_t status;
92     papi_service_t svc = NULL;
93     papi_encryption_t encryption = PAPI_ENCRYPT_NEVER;
94     int exit_status = 0;
95     int cancel = 0;
96     int pending = 0; /* not implemented */
97     char *reason = NULL;
98     int c;

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

102    while ((c = getopt(ac, av, "EcWr:")) != EOF)
103        switch (c) {
104            case 'c': /* cancel active job first */
105                cancel = 1;
106                break;
107            case 'W': /* wait for active request, not implemented */
108                pending = 1;
109                break;
110            case 'r': /* reason */
111                reason = optarg;
112                break;
113            case 'E':
114                encryption = PAPI_ENCRYPT_NEVER;
115                break;
116            default:
117                usage(av[0]);
118        }

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

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

125        status = papiServiceCreate(&svc, printer, NULL, NULL,
126                                  cli_auth_callback, encryption, NULL);
127        if (status != PAPI_OK) {
128            fprintf(stderr, gettext(
129                "Failed to contact service for %s: %s\n"),
130                printer, verbose_papi_message(svc, status));
131            exit_status = 1;
132        }

134        status = papiPrinterDisable(svc, printer, reason);
135        if (status == PAPI_OK) {
136            printf(gettext("printer \"%s\" now disabled\n"),
137                  printer);
138        } else if (status == PAPI_NOT_ACCEPTING) {
139            fprintf(stderr, gettext(
140                "Destination \"%s\" was already disabled.\n"),
141                printer);
142            exit_status = 1;
143        } else {
144            /* The operation is not supported in lpd protocol */
145            if (status == PAPI_OPERATION_NOT_SUPPORTED) {
146                fprintf(stderr,
147                        verbose_papi_message(svc, status));
148            } else {
149                fprintf(stderr, gettext("disable: %s: %s\n"),
150                        printer, verbose_papi_message(svc, status));
151            }
152            exit_status = 1;
153        }
154    }
```

new/usr/src/cmd/print/bsd-sysv-commands/disable.c

3

```
155         if (cancel != 0)
156             cancel_active_job(svc, printer);
158         papiServiceDestroy(svc);
159     }
161     return (exit_status);
162 }
```

unchanged_portion_omitted_

new/usr/src/cmd/print/bsd-sysv-commands/lpr.c

1

```
*****
7223 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/print/bsd-sysv-commands/lpr.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21
22 /*
23  * Copyright 2017 Gary Mills
24  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26  *
27  */
28
29 /* $Id: lpr.c 146 2006-03-24 00:26:54Z njacobs $ */
30 #pragma ident "%Z%M% %I% %E% SMI"
31
32 #include <stdio.h>
33 #include <stdlib.h>
34 #include <unistd.h>
35 #include <string.h>
36 #include <locale.h>
37 #include <libintl.h>
38 #include <papi.h>
39 #include "common.h"
40
41 #ifdef HAVE_LIBMAGIC /* for mimetype auto-detection */
42 #include <magic.h>
43 #endif
44
45 static void
46 usage(char *program)
47 {
48     char *name;
49
50     if ((name = strrchr(program, '/')) == NULL)
51         name = program;
52     else
53         name++;
54
55     fprintf(stdout,
56             gettext("Usage: %s [-P printer] [-# copies] [-C class] "
57                     "[-J job] [-T title] "
58                     "[-p [-i indent] [-w width]] "
59                     "[-1|-2|-3|-4 font] [-m] [-h] [-s] "
60                     "[-filter_option] [file ..]\n"), name);
61 }
```

new/usr/src/cmd/print/bsd-sysv-commands/lpr.c

2

```
60     exit(1);
61 }
62
63 int
64 main(int ac, char *av[])
65 {
66     papi_status_t status;
67     papi_service_t svc = NULL;
68     papi_attribute_t **list = NULL;
69     papi_job_t job = NULL;
70     int exit_code = 0;
71     char *printer = NULL;
72     char prefetch[3];
73     int prefetch_len = sizeof (prefetch);
74     papi_encryption_t encryption = PAPI_ENCRYPT_NEVER;
75     int dump = 0;
76     int validate = 0;
77     int remove = 0;
78     int copy = 1; /* default is to copy the data */
79     char *document_format = "text/plain";
80     int c;
81
82     (void) setlocale(LC_ALL, "");
83     (void) textdomain("SUNW_OST_OSCMD");
84
85     while ((c = getopt(ac, av,
86                       "EP:#:C:DVJ:T:w:i:hplrstdgvcfmn1:2:3:4:")) != EOF)
87     {
88         switch (c) {
89             case 'E':
90                 encryption = PAPI_ENCRYPT_REQUIRED;
91                 break;
92             case 'P':
93                 printer = optarg;
94                 break;
95             case '#':
96                 papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
97                                             "copies", atoi(optarg));
98                 break;
99             case 'C':
100                 papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
101                                             "rfc-1179-class", optarg);
102                 break;
103             case 'D':
104                 dump = 1;
105                 break;
106             case 'J':
107                 papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
108                                             "job-name", optarg);
109                 break;
110             case 'T':
111                 papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
112                                             "pr-title", optarg);
113                 break;
114             case 'p':
115                 papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
116                                             "document-format", "application/x-pr");
117                 papiAttributeListAddBoolean(&list, PAPI_ATTR_EXCL,
118                                             "pr-filter", 1);
119                 break;
120             case 'i':
121                 papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
122                                             "pr-indent", atoi(optarg));
123                 break;
124             case 'w':
125                 papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
126                                             "pr-width", atoi(optarg));
127                 break;
128             default:
129                 break;
130         }
131     }
132
133     if (printer == NULL)
134         printer = getenv("PRINTER");
135
136     if (printer == NULL)
137         printer = "lpr";
138
139     if (document_format == NULL)
140         document_format = "text/plain";
141
142     if (encryption == PAPI_ENCRYPT_REQUIRED)
143         document_format = "text/plain; charset=utf-8; encryption=required";
144
145     if (dump)
146         document_format = "text/plain; charset=utf-8; encryption=required; dump=1";
147
148     if (validate)
149         document_format = "text/plain; charset=utf-8; encryption=required; validate=1";
150
151     if (remove)
152         document_format = "text/plain; charset=utf-8; encryption=required; remove=1";
153
154     if (copy)
155         document_format = "text/plain; charset=utf-8; encryption=required; copy=1";
156
157     if (list != NULL)
158         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
159                                     "pr-copies", atoi(optarg));
160
161     if (list != NULL)
162         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
163                                     "pr-format", document_format);
164
165     if (list != NULL)
166         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
167                                     "pr-width", atoi(optarg));
168
169     if (list != NULL)
170         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
171                                     "pr-height", atoi(optarg));
172
173     if (list != NULL)
174         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
175                                     "pr-resolution", atoi(optarg));
176
177     if (list != NULL)
178         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
179                                     "pr-dpi", atoi(optarg));
180
181     if (list != NULL)
182         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
183                                     "pr-pages", atoi(optarg));
184
185     if (list != NULL)
186         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
187                                     "pr-size", atoi(optarg));
188
189     if (list != NULL)
190         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
191                                     "pr-timeout", atoi(optarg));
192
193     if (list != NULL)
194         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
195                                     "pr-verbose", atoi(optarg));
196
197     if (list != NULL)
198         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
199                                     "pr-quiet", atoi(optarg));
200
201     if (list != NULL)
202         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
203                                     "pr-silent", atoi(optarg));
204
205     if (list != NULL)
206         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
207                                     "pr-verbose", atoi(optarg));
208
209     if (list != NULL)
210         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
211                                     "pr-quiet", atoi(optarg));
212
213     if (list != NULL)
214         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
215                                     "pr-silent", atoi(optarg));
216
217     if (list != NULL)
218         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
219                                     "pr-verbose", atoi(optarg));
220
221     if (list != NULL)
222         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
223                                     "pr-quiet", atoi(optarg));
224
225     if (list != NULL)
226         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
227                                     "pr-silent", atoi(optarg));
228
229     if (list != NULL)
230         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
231                                     "pr-verbose", atoi(optarg));
232
233     if (list != NULL)
234         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
235                                     "pr-quiet", atoi(optarg));
236
237     if (list != NULL)
238         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
239                                     "pr-silent", atoi(optarg));
240
241     if (list != NULL)
242         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
243                                     "pr-verbose", atoi(optarg));
244
245     if (list != NULL)
246         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
247                                     "pr-quiet", atoi(optarg));
248
249     if (list != NULL)
250         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
251                                     "pr-silent", atoi(optarg));
252
253     if (list != NULL)
254         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
255                                     "pr-verbose", atoi(optarg));
256
257     if (list != NULL)
258         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
259                                     "pr-quiet", atoi(optarg));
260
261     if (list != NULL)
262         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
263                                     "pr-silent", atoi(optarg));
264
265     if (list != NULL)
266         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
267                                     "pr-verbose", atoi(optarg));
268
269     if (list != NULL)
270         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
271                                     "pr-quiet", atoi(optarg));
272
273     if (list != NULL)
274         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
275                                     "pr-silent", atoi(optarg));
276
277     if (list != NULL)
278         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
279                                     "pr-verbose", atoi(optarg));
280
281     if (list != NULL)
282         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
283                                     "pr-quiet", atoi(optarg));
284
285     if (list != NULL)
286         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
287                                     "pr-silent", atoi(optarg));
288
289     if (list != NULL)
290         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
291                                     "pr-verbose", atoi(optarg));
292
293     if (list != NULL)
294         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
295                                     "pr-quiet", atoi(optarg));
296
297     if (list != NULL)
298         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
299                                     "pr-silent", atoi(optarg));
300
301     if (list != NULL)
302         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
303                                     "pr-verbose", atoi(optarg));
304
305     if (list != NULL)
306         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
307                                     "pr-quiet", atoi(optarg));
308
309     if (list != NULL)
310         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
311                                     "pr-silent", atoi(optarg));
312
313     if (list != NULL)
314         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
315                                     "pr-verbose", atoi(optarg));
316
317     if (list != NULL)
318         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
319                                     "pr-quiet", atoi(optarg));
320
321     if (list != NULL)
322         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
323                                     "pr-silent", atoi(optarg));
324
325     if (list != NULL)
326         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
327                                     "pr-verbose", atoi(optarg));
328
329     if (list != NULL)
330         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
331                                     "pr-quiet", atoi(optarg));
332
333     if (list != NULL)
334         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
335                                     "pr-silent", atoi(optarg));
336
337     if (list != NULL)
338         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
339                                     "pr-verbose", atoi(optarg));
340
341     if (list != NULL)
342         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
343                                     "pr-quiet", atoi(optarg));
344
345     if (list != NULL)
346         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
347                                     "pr-silent", atoi(optarg));
348
349     if (list != NULL)
350         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
351                                     "pr-verbose", atoi(optarg));
352
353     if (list != NULL)
354         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
355                                     "pr-quiet", atoi(optarg));
356
357     if (list != NULL)
358         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
359                                     "pr-silent", atoi(optarg));
360
361     if (list != NULL)
362         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
363                                     "pr-verbose", atoi(optarg));
364
365     if (list != NULL)
366         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
367                                     "pr-quiet", atoi(optarg));
368
369     if (list != NULL)
370         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
371                                     "pr-silent", atoi(optarg));
372
373     if (list != NULL)
374         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
375                                     "pr-verbose", atoi(optarg));
376
377     if (list != NULL)
378         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
379                                     "pr-quiet", atoi(optarg));
380
381     if (list != NULL)
382         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
383                                     "pr-silent", atoi(optarg));
384
385     if (list != NULL)
386         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
387                                     "pr-verbose", atoi(optarg));
388
389     if (list != NULL)
390         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
391                                     "pr-quiet", atoi(optarg));
392
393     if (list != NULL)
394         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
395                                     "pr-silent", atoi(optarg));
396
397     if (list != NULL)
398         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
399                                     "pr-verbose", atoi(optarg));
400
401     if (list != NULL)
402         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
403                                     "pr-quiet", atoi(optarg));
404
405     if (list != NULL)
406         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
407                                     "pr-silent", atoi(optarg));
408
409     if (list != NULL)
410         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
411                                     "pr-verbose", atoi(optarg));
412
413     if (list != NULL)
414         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
415                                     "pr-quiet", atoi(optarg));
416
417     if (list != NULL)
418         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
419                                     "pr-silent", atoi(optarg));
420
421     if (list != NULL)
422         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
423                                     "pr-verbose", atoi(optarg));
424
425     if (list != NULL)
426         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
427                                     "pr-quiet", atoi(optarg));
428
429     if (list != NULL)
430         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
431                                     "pr-silent", atoi(optarg));
432
433     if (list != NULL)
434         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
435                                     "pr-verbose", atoi(optarg));
436
437     if (list != NULL)
438         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
439                                     "pr-quiet", atoi(optarg));
440
441     if (list != NULL)
442         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
443                                     "pr-silent", atoi(optarg));
444
445     if (list != NULL)
446         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
447                                     "pr-verbose", atoi(optarg));
448
449     if (list != NULL)
450         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
451                                     "pr-quiet", atoi(optarg));
452
453     if (list != NULL)
454         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
455                                     "pr-silent", atoi(optarg));
456
457     if (list != NULL)
458         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
459                                     "pr-verbose", atoi(optarg));
460
461     if (list != NULL)
462         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
463                                     "pr-quiet", atoi(optarg));
464
465     if (list != NULL)
466         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
467                                     "pr-silent", atoi(optarg));
468
469     if (list != NULL)
470         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
471                                     "pr-verbose", atoi(optarg));
472
473     if (list != NULL)
474         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
475                                     "pr-quiet", atoi(optarg));
476
477     if (list != NULL)
478         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
479                                     "pr-silent", atoi(optarg));
480
481     if (list != NULL)
482         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
483                                     "pr-verbose", atoi(optarg));
484
485     if (list != NULL)
486         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
487                                     "pr-quiet", atoi(optarg));
488
489     if (list != NULL)
490         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
491                                     "pr-silent", atoi(optarg));
492
493     if (list != NULL)
494         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
495                                     "pr-verbose", atoi(optarg));
496
497     if (list != NULL)
498         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
499                                     "pr-quiet", atoi(optarg));
500
501     if (list != NULL)
502         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
503                                     "pr-silent", atoi(optarg));
504
505     if (list != NULL)
506         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
507                                     "pr-verbose", atoi(optarg));
508
509     if (list != NULL)
510         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
511                                     "pr-quiet", atoi(optarg));
512
513     if (list != NULL)
514         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
515                                     "pr-silent", atoi(optarg));
516
517     if (list != NULL)
518         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
519                                     "pr-verbose", atoi(optarg));
520
521     if (list != NULL)
522         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
523                                     "pr-quiet", atoi(optarg));
524
525     if (list != NULL)
526         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
527                                     "pr-silent", atoi(optarg));
528
529     if (list != NULL)
530         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
531                                     "pr-verbose", atoi(optarg));
532
533     if (list != NULL)
534         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
535                                     "pr-quiet", atoi(optarg));
536
537     if (list != NULL)
538         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
539                                     "pr-silent", atoi(optarg));
540
541     if (list != NULL)
542         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
543                                     "pr-verbose", atoi(optarg));
544
545     if (list != NULL)
546         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
547                                     "pr-quiet", atoi(optarg));
548
549     if (list != NULL)
550         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
551                                     "pr-silent", atoi(optarg));
552
553     if (list != NULL)
554         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
555                                     "pr-verbose", atoi(optarg));
556
557     if (list != NULL)
558         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
559                                     "pr-quiet", atoi(optarg));
560
561     if (list != NULL)
562         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
563                                     "pr-silent", atoi(optarg));
564
565     if (list != NULL)
566         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
567                                     "pr-verbose", atoi(optarg));
568
569     if (list != NULL)
570         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
571                                     "pr-quiet", atoi(optarg));
572
573     if (list != NULL)
574         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
575                                     "pr-silent", atoi(optarg));
576
577     if (list != NULL)
578         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
579                                     "pr-verbose", atoi(optarg));
580
581     if (list != NULL)
582         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
583                                     "pr-quiet", atoi(optarg));
584
585     if (list != NULL)
586         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
587                                     "pr-silent", atoi(optarg));
588
589     if (list != NULL)
590         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
591                                     "pr-verbose", atoi(optarg));
592
593     if (list != NULL)
594         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
595                                     "pr-quiet", atoi(optarg));
596
597     if (list != NULL)
598         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
599                                     "pr-silent", atoi(optarg));
600
601     if (list != NULL)
602         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
603                                     "pr-verbose", atoi(optarg));
604
605     if (list != NULL)
606         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
607                                     "pr-quiet", atoi(optarg));
608
609     if (list != NULL)
610         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
611                                     "pr-silent", atoi(optarg));
612
613     if (list != NULL)
614         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
615                                     "pr-verbose", atoi(optarg));
616
617     if (list != NULL)
618         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
619                                     "pr-quiet", atoi(optarg));
620
621     if (list != NULL)
622         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
623                                     "pr-silent", atoi(optarg));
624
625     if (list != NULL)
626         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
627                                     "pr-verbose", atoi(optarg));
628
629     if (list != NULL)
630         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
631                                     "pr-quiet", atoi(optarg));
632
633     if (list != NULL)
634         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
635                                     "pr-silent", atoi(optarg));
636
637     if (list != NULL)
638         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
639                                     "pr-verbose", atoi(optarg));
640
641     if (list != NULL)
642         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
643                                     "pr-quiet", atoi(optarg));
644
645     if (list != NULL)
646         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
647                                     "pr-silent", atoi(optarg));
648
649     if (list != NULL)
650         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
651                                     "pr-verbose", atoi(optarg));
652
653     if (list != NULL)
654         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
655                                     "pr-quiet", atoi(optarg));
656
657     if (list != NULL)
658         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
659                                     "pr-silent", atoi(optarg));
660
661     if (list != NULL)
662         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
663                                     "pr-verbose", atoi(optarg));
664
665     if (list != NULL)
666         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
667                                     "pr-quiet", atoi(optarg));
668
669     if (list != NULL)
670         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
671                                     "pr-silent", atoi(optarg));
672
673     if (list != NULL)
674         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
675                                     "pr-verbose", atoi(optarg));
676
677     if (list != NULL)
678         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
679                                     "pr-quiet", atoi(optarg));
680
681     if (list != NULL)
682         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
683                                     "pr-silent", atoi(optarg));
684
685     if (list != NULL)
686         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
687                                     "pr-verbose", atoi(optarg));
688
689     if (list != NULL)
690         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
691                                     "pr-quiet", atoi(optarg));
692
693     if (list != NULL)
694         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
695                                     "pr-silent", atoi(optarg));
696
697     if (list != NULL)
698         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
699                                     "pr-verbose", atoi(optarg));
700
701     if (list != NULL)
702         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
703                                     "pr-quiet", atoi(optarg));
704
705     if (list != NULL)
706         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
707                                     "pr-silent", atoi(optarg));
708
709     if (list != NULL)
710         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
711                                     "pr-verbose", atoi(optarg));
712
713     if (list != NULL)
714         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
715                                     "pr-quiet", atoi(optarg));
716
717     if (list != NULL)
718         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
719                                     "pr-silent", atoi(optarg));
720
721     if (list != NULL)
722         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
723                                     "pr-verbose", atoi(optarg));
724
725     if (list != NULL)
726         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
727                                     "pr-quiet", atoi(optarg));
728
729     if (list != NULL)
730         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
731                                     "pr-silent", atoi(optarg));
732
733     if (list != NULL)
734         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
735                                     "pr-verbose", atoi(optarg));
736
737     if (list != NULL)
738         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
739                                     "pr-quiet", atoi(optarg));
740
741     if (list != NULL)
742         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
743                                     "pr-silent", atoi(optarg));
744
745     if (list != NULL)
746         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
747                                     "pr-verbose", atoi(optarg));
748
749     if (list != NULL)
750         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
751                                     "pr-quiet", atoi(optarg));
752
753     if (list != NULL)
754         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
755                                     "pr-silent", atoi(optarg));
756
757     if (list != NULL)
758         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
759                                     "pr-verbose", atoi(optarg));
760
761     if (list != NULL)
762         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
763                                     "pr-quiet", atoi(optarg));
764
765     if (list != NULL)
766         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
767                                     "pr-silent", atoi(optarg));
768
769     if (list != NULL)
770         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
771                                     "pr-verbose", atoi(optarg));
772
773     if (list != NULL)
774         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
775                                     "pr-quiet", atoi(optarg));
776
777     if (list != NULL)
778         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
779                                     "pr-silent", atoi(optarg));
780
781     if (list != NULL)
782         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
783                                     "pr-verbose", atoi(optarg));
784
785     if (list != NULL)
786         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
787                                     "pr-quiet", atoi(optarg));
788
789     if (list != NULL)
790         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
791                                     "pr-silent", atoi(optarg));
792
793     if (list != NULL)
794         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
795                                     "pr-verbose", atoi(optarg));
796
797     if (list != NULL)
798         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
799                                     "pr-quiet", atoi(optarg));
800
801     if (list != NULL)
802         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
803                                     "pr-silent", atoi(optarg));
804
805     if (list != NULL)
806         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
807                                     "pr-verbose", atoi(optarg));
808
809     if (list != NULL)
810         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
811                                     "pr-quiet", atoi(optarg));
812
813     if (list != NULL)
814         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
815                                     "pr-silent", atoi(optarg));
816
817     if (list != NULL)
818         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
819                                     "pr-verbose", atoi(optarg));
820
821     if (list != NULL)
822         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
823                                     "pr-quiet", atoi(optarg));
824
825     if (list != NULL)
826         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
827                                     "pr-silent", atoi(optarg));
828
829     if (list != NULL)
830         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
831                                     "pr-verbose", atoi(optarg));
832
833     if (list != NULL)
834         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
835                                     "pr-quiet", atoi(optarg));
836
837     if (list != NULL)
838         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
839                                     "pr-silent", atoi(optarg));
840
841     if (list != NULL)
842         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
843                                     "pr-verbose", atoi(optarg));
844
845     if (list != NULL)
846         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
847                                     "pr-quiet", atoi(optarg));
848
849     if (list != NULL)
850         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
851                                     "pr-silent", atoi(optarg));
852
853     if (list != NULL)
854         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
855                                     "pr-verbose", atoi(optarg));
856
857     if (list != NULL)
858         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
859                                     "pr-quiet", atoi(optarg));
860
861     if (list != NULL)
862         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
863                                     "pr-silent", atoi(optarg));
864
865     if (list != NULL)
866         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
867                                     "pr-verbose", atoi(optarg));
868
869     if (list != NULL)
870         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
871                                     "pr-quiet", atoi(optarg));
872
873     if (list != NULL)
874         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
875                                     "pr-silent", atoi(optarg));
876
877     if (list != NULL)
878         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
879                                     "pr-verbose", atoi(optarg));
880
881     if (list != NULL)
882         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
883                                     "pr-quiet", atoi(optarg));
884
885     if (list != NULL)
886         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
887                                     "pr-silent", atoi(optarg));
888
889     if (list != NULL)
890         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
891                                     "pr-verbose", atoi(optarg));
892
893     if (list != NULL)
894         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
895                                     "pr-quiet", atoi(optarg));
896
897     if (list != NULL)
898         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
899                                     "pr-silent", atoi(optarg));
900
901     if (list != NULL)
902         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
903                                     "pr-verbose", atoi(optarg));
904
905     if (list != NULL)
906         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
907                                     "pr-quiet", atoi(optarg));
908
909     if (list != NULL)
910         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
911                                     "pr-silent", atoi(optarg));
912
913     if (list != NULL)
914         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
915                                     "pr-verbose", atoi(optarg));
916
917     if (list != NULL)
918         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
919                                     "pr-quiet", atoi(optarg));
920
921     if (list != NULL)
922         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
923                                     "pr-silent", atoi(optarg));
924
925     if (list != NULL)
926         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
927                                     "pr-verbose", atoi(optarg));
928
929     if (list != NULL)
930         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
931                                     "pr-quiet", atoi(optarg));
932
933     if (list != NULL)
934         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
935                                     "pr-silent", atoi(optarg));
936
937     if (list != NULL)
938         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
939                                     "pr-verbose", atoi(optarg));
940
941     if (list != NULL)
942         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
943                                     "pr-quiet", atoi(optarg));
944
945     if (list != NULL)
946         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
947                                     "pr-silent", atoi(optarg));
948
949     if (list != NULL)
950         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
951                                     "pr-verbose", atoi(optarg));
952
953     if (list != NULL)
954         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
955                                     "pr-quiet", atoi(optarg));
956
957     if (list != NULL)
958         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
959                                     "pr-silent", atoi(optarg));
960
961     if (list != NULL)
962         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
963                                     "pr-verbose", atoi(optarg));
964
965     if (list != NULL)
966         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
967                                     "pr-quiet", atoi(optarg));
968
969     if (list != NULL)
970         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
971                                     "pr-silent", atoi(optarg));
972
973     if (list != NULL)
974         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
975                                     "pr-verbose", atoi(optarg));
976
977     if (list != NULL)
978         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
979                                     "pr-quiet", atoi(optarg));
980
981     if (list != NULL)
982         papiAttributeListAddInteger(&list, PAPI_ATTR_EXCL,
983                                     "pr-silent", atoi(optarg));
984
985     if (list != NULL)
986         papiAttributeListAddInteger(&list
```

```

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':
171         remove = 1;
172         break;
173     case 's':
174         copy = 0;
175         break;
176     case 'V': /* validate */
177         validate = 1;
178         break;
179     case '1':
180         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
181             "rfc-1179-font-r", optarg);
182         break;
183     case '2':
184         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
185             "rfc-1179-font-i", optarg);
186         break;
187     case '3':
188         papiAttributeListAddString(&list, PAPI_ATTR_EXCL,
189             "rfc-1179-font-b", optarg);
190         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     }
197
198     if ((printer == NULL) &&
199         ((printer = getenv("PRINTER")) == NULL) &&
200         ((printer = getenv("LPDEST")) == NULL))
201         printer = DEFAULT_DEST;
202
203     if (((optind + 1) == ac) && (strcmp(av[optind], "-") == 0))
204         optind = ac;
205
206     if (optind != ac) {
207         /* get the mime type of the file data */
208 #ifdef MAGIC_MIME
209         magic_t ms;
210
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     }
223
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");
229
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     }
238
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);
251
252     papiAttributeListFree(list);
253
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     }
261
262     if (dump != 0) {
263         list = papiJobGetAttributeList(job);
264         printf("job attributes:\n");
265         papiAttributeListPrint(stdout, list, "\t");
266         printf("\n");
267     }
268
269     papiJobFree(job);
270     papiServiceDestroy(svc);
271
272     return (exit_code);
273 }
```

unchanged_portion_omitted

new/usr/src/cmd/refer/glue4.c

1

```
*****
2296 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/refer/glue4.c
8485 Remove set but unused variables in usr/src/cmd
*****
1 /*
2  * Copyright 2017 Gary Mills
3  * Copyright 2005 Sun Microsystems, Inc. All rights reserved.
4  * Use is subject to license terms.
5  */

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

10 /*
11  * Copyright (c) 1980 Regents of the University of California.
12  * All rights reserved. The Berkeley software license Agreement
13  * specifies the terms and conditions for redistribution.
14  */

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

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

20 extern char gfile[];

22 void
23 grepcall(char *in, char *out, char *arg)
24 {
25     char line[200], *s, argig[100], *cv[50];
26     char *inp, inb[500];
27     FILE *qf, *gf;
28     int c, alph = 0, nv = 0;
29     int c, oldc = 0, alph = 0, nv = 0;
30     int sv0, svi;
31     strcpy(argig, arg);
32     strcat(argig, ".ig");
33     strcpy(inp = inb, in);
34     if (gfile[0] == 0)
35         sprintf(gfile, "/tmp/rj%dg", getpid());
36     #if D1
37         fprintf(stderr, "in grepcall, gfile %s in %o out %o\n",
38             gfile, in, out);
39     #endif
40     for (cv[nv++] = "fgrep"; c = *inp; inp++) {
41         if (c == ',')
42             c = *inp = 0;
43         else if (isupper(c))
44             *inp = tolower(c);
45         alph = (c == 0) ? 0 : alph+1;
46         if (alph == 1)
47             cv[nv++] = inp;
48         if (alph > 6)
49             *inp = 0;
50         oldc = c;
51     }
52     #if D1
53         fprintf(stderr, "%d args set up\n", nv);
54     #endif
55     {
56         sv0 = dup(0);
57         close(0);
58         if (open(argig, 0) != 0)
59             err("Can't read fgrep index %s", argig);
```

new/usr/src/cmd/refer/glue4.c

2

```
58         sv1 = dup(1);
59         close(1);
60         if (creat(gfile, 0666) != 1)
61             err("Can't write fgrep output %s", gfile);
62         fgrep(nv, cv);
63     #if D1
64         fprintf(stderr, "fgrep returned, output is..\\n");
65     #endif
66         close(0);
67         dup(sv0);
68         close(sv0);
69         close(1);
70         dup(sv1);
71         close(sv1);
72     }

74 #if D1
75     fprintf(stderr, "back from fgrep\\n");
76 #endif
77     gf = fopen(gfile, "r");
78     if (gf == NULL)
79         err("can't read fgrep output %s", gfile);
80     while (fgets(line, 100, gf) == line) {
81         line[100] = 0;
82     #if D1
83         fprintf(stderr, "read line as //s//\\n", line);
84     #endif
85         for (s = line; *s && (*s != '\\t'); s++)
86             ;
87         if (*s == '\\t') {
88             *s++ = '\\n';
89             *s++ = 0;
90         }
91         if (line[0])
92             strcat(out, line);
93     #if D1
94         fprintf(stderr, "out now /%s//\\n", out);
95     #endif
96         while (*s) s++;
97     #if D1
98         fprintf(stderr, "line %o s %o s-1 %o\\n", line, s, s[-1]);
99     #endif
100         if (s[-1] != '\\n')
101             while (!feof(gf) && getc(gf) != '\\n')
102                 ;
103     }
104     fclose(gf);
105 #if D1
106     fprintf(stderr, "back from reading %, out %s\\n", out);
107 #else
108     unlink(gfile);
109 #endif
110 }
```

unchanged_portion_omitted

new/usr/src/cmd/refer/hunt7.c

1

```
*****
2681 Sun Jul 16 20:04:39 2017
new/usr/src/cmd/refer/hunt7.c
8485 Remove set but unused variables in usr/src/cmd
*****
1 /*
2  * Copyright 2017 Gary Mills
3  * Copyright 2005 Sun Microsystems, Inc. All rights reserved.
4  * Use is subject to license terms.
5  */

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

10 /*
11  * Copyright (c) 1980 Regents of the University of California.
12  * All rights reserved. The Berkeley software license Agreement
13  * specifies the terms and conditions for redistribution.
14  */

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

16 #include <stdio.h>
17 #include <locale.h>
18 #include <assert.h>
19 #define SAME 0
20 #define FGCT 10
21 #define FGSIZE 150

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

29 extern char *mindex();

31 long
32 findline(char *in, char **out, int outlen, long indexdate)
33 {
34     static char name[100] = "";
35     char *p, **ftp;
36     extern long gdate();
37     static FILE *fa = NULL;
38     long lp, llen;
39     int k, nofil;
40     int len, k, nofil;

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

47     nofil = in[0] == 0;
48     for (p = in; *p && *p != ':' && *p != ';' && *p != ' '; p++)
49         ;
50     if (*p) *p++ = 0;
51     else p = in;
52     k = sscanf(p, "%ld,%ld", &lp, &llen);
53 #ifdef D1
54     fprintf(stderr, "p %s k %d lp %ld llen %ld\n", p, k, lp, llen);
55 #endif
56     if (k < 2) {
57         lp = 0;
58         llen = outlen;
```

new/usr/src/cmd/refer/hunt7.c

2

```
59     }
60 #ifdef D1
61     fprintf(stderr, "lp %ld llen %ld\n", lp, llen);
62 #endif
63 #ifdef D1
64     fprintf(stderr, "fa now %o, p %o in %o %s\n", fa, p, in, in);
65 #endif
66     if (nofil) {
67 #if D1
68         fprintf(stderr, "set fa to stdin\n");
69 #endif
70         fa = stdin;
71     } else
72         if (strcmp(name, in) != 0 || 1) {
73 #if D1
74             fprintf(stderr, "old: %s new %s not equal\n", name, in);
75 #endif
76             if (fa != NULL)
77                 fa = freopen(in, "r", fa);
78             else
79                 fa = fopen(in, "r");
80 #if D1
81             if (fa == NULL)
82                 fprintf(stderr, "failed to (re)open %s*\n",
83                     in);
84 #endif
85             if (fa == NULL)
86                 return (0);
87             /* err("Can't open %s", in); */
88             strcpy(name, in);
89             if (gdate(fa) > indexdate && indexdate != 0) {
90                 if (keepold) {
91                     for (ftp = fgnames; ftp < fgnamp; ftp++)
92                         if (strcmp(*ftp, name) == SAME)
93                             return (0);
94                     strcpy(*fgnamp++ = fgp, name);
95                     assert(fgnamp < fgnames+FGCT);
96                     while (*fgp && *fgp != ':')
97                         fgp++;
98                     *fgp++ = 0;
99                     assert(fgp < fgspace+FGSIZE);
100                     return (0);
101                 }
102                 fprintf(stderr, gettext(
103                     "Warning: index predates file '%s'\n"),
104                     name);
105             }
106         }
107 #if D1
108         else
109             fprintf(stderr, "old %s new %s same fa %o\n",
110                 name, in, fa);
111 #endif
112     if (fa != NULL) {
113         fseek(fa, lp, 0);
114         *out = (char *)malloc(llen + 1);
115         if (*out == NULL) {
116             return (0);
117         }
118         (void) fread(*out, 1, llen, fa);
119         len = fread(*out, 1, llen, fa);
120         *out = (*out + llen) = 0;
121 #ifdef D1
122         fprintf(stderr, "length as read is %d\n", len);
123 #endif
124     }
```

new/usr/src/cmd/refer/hunt7.c

```
124         return (llen);
```

```
125     }
```

_____unchanged_portion_omitted_

new/usr/src/cmd/sh/xec.c

19815 Sun Jul 16 20:04:40 2017

new/usr/src/cmd/sh/xec.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
22 /*
23  * Copyright 2017 Gary Mills
24  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */
28 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
29 /*      All Rights Reserved */
31 /*
32  *
33  * UNIX shell
34  *
35 */
38 #include      "defs.h"
39 #include      <errno.h>
40 #include      "sym.h"
41 #include      "hash.h"
42 #include      <sys/types.h>
43 #include      <sys/times.h>
45 pid_t parent;
47 void execprint(unsigned char **);
49 /* =====      command execution      ===== */
51 /*VARARGS3*/
52 int
53 execute(argt, xflags, errorflg, pf1, pf2)
54 struct trenod *argt;
55 int xflags, errorflg;
56 int *pf1, *pf2;
57 {
58     /*
59      * 'stakbot' is preserved by this routine
60      */
61     struct trenod *t;
```

1

new/usr/src/cmd/sh/xec.c

```
62     unsigned char      *sav = savstak();
64     sigchk();
65     if (!errorflg)
66         flags &= ~errflg;
68     if ((t = argt) && execbrk == 0) {
69         int treeflgs;
70         unsigned char **com;
71         int type;
72         short pos;
74         treeflgs = t->tretyp;
75         type = treeflgs & COMMSK;
77         switch (type)
78         {
79             case TFND:
80                 {
81                     struct fndnod *f = fndptr(t);
82                     struct namnod *n = lookup(f->fndnam);
84                     exitval = 0;
86                     if (n->namflg & N_RDONLY)
87                         failed(n->namid, wtfailed);
89                     if (flags & rshflg && (n == &pathnod ||
90                        eq(n->namid, "SHELL")))
91                         failed(n->namid, restricted);
92                     /*
93                      * If function of same name is previously
94                      * defined, it will no longer be used.
95                      */
96                     if (n->namflg & N_FUNCNTN) {
97                         freefunc(n);
98                     } else {
99                         free(n->namval);
100                        free(n->namenv);
102                        n->namval = 0;
103                        n->namflg &= ~(N_EXPORT | N_ENVCHG);
104                    }
105                    /*
106                     * If function is defined within function,
107                     * we don't want to free it along with the
108                     * free of the defining function. If we are
109                     * in a loop, fndnod may be reused, so it
110                     * should never be freed.
111                     */
112                    if (funcnt != 0 || loopcnt != 0)
113                        f->fndref++;
115                    /*
116                     * We hang a fndnod on the namenv so that
117                     * ref cnt(fndref) can be increased while
118                     * running in the function.
119                     */
120                    n->namenv = (unsigned char *)f;
121                    attrib(n, N_FUNCNTN);
122                    hash_func(n->namid);
123                    break;
124                }
126                case TCOM:
127                {
```

2

```

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|jcf|g|jcoff))
246                 == (monitorflg|jcf|g));
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         rmtmp(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];
379
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
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      }
404      case TFOR:
405      {
406          struct namnod *n = lookup(forptr(t)->fornam);
407          unsigned char **args;
408          struct dolnod *argsav = 0;
410          if (forptr(t)->forlst == 0)
411          {
412              args = dolv + 1;
413              argsav = useargs();
414          }
415          else
416          {
417              struct argnod *schain = gchain;
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);
434          loopcnt--;
435          if(argsav)
436              argfor = (struct dolnod *)freeargs(argsa
437          }
438          break;
440      case TWH:
441      case TUN:
442      {
443          int i = 0;
445          loopcnt++;
446          while (execbrk == 0 && (execute(whptr(t)->whtr
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;
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;
471      case TSW:
472      {
473          unsigned char *r = mactrim(swptr(t)->swarg);
474          struct regnod *regp;
476          regp = swptr(t)->swlst;
477          while (regp)
478          {
479              struct argnod *rex = regp->regptr;
481              while (rex)
482              {
483                  unsigned char *s;
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          break;
498      }
499      }
500      exitset();
501      }
502      sigchk();
503      tdystak(sav);
504      flags |= eflag;
505      return(exitval);
506  }

```

unchanged_portion_omitted

new/usr/src/cmd/svr4pkg/pkgremove/special.c

1

```
*****
18493 Sun Jul 16 20:04:40 2017
new/usr/src/cmd/svr4pkg/pkgremove/special.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2017 Gary Mills
24  * Copyright 2003 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

29 /*
30  * special.c
31  *
32  * This module contains code required to remove special contents from
33  * the contents file when a pkgrm is done on a system upgraded to use
34  * the new database.
35 */

37 #include <stdio.h>
38 #include <stdlib.h>
39 #include <assert.h>
40 #include <errno.h>
41 #include <unistd.h>
42 #include <string.h>
43 #include <time.h>
44 #include <limits.h>
45 #include <fnmatch.h>
46 #include <sys/types.h>
47 #include <sys/stat.h>
48 #include <pkgstrct.h>
49 #include "pkglib.h"
50 #include <libintl.h>

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

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

new/usr/src/cmd/svr4pkg/pkgremove/special.c

2

```
62 /*
63  * strcmpare
64  *
65  * This function is used by qsort to sort an array of special contents
66  * rule strings. This array must be sorted to facilitate efficient
67  * rule processing. See qsort(3c) regarding qsort compare functions.
68 */
69 static int
70 strcmpare(const void *pv1, const void *pv2)
71 {
72     char **ppc1 = (char **) pv1;
73     char **ppc2 = (char **) pv2;
74     int i = strcmp(*ppc1, *ppc2);
75     if (i < 0)
76         return (-1);
77     if (i > 0)
78         return (1);
79     return (0);
80 }
    unchanged_portion_omitted

472 /*
473  * -----
474  * Externally visible function.
475 */

477 /*
478  * special_contents_remove
479  *
480  * Given a set of entries to remove and an alternate root, this function
481  * will do everything required to ensure that the entries are removed
482  * from the contents file if they are listed in the special_contents
483  * file. The contents file will get changed only in the case that the
484  * entire operation has succeeded.
485  *
486  * ient        The number of entries.
487  * ppcfent     The entries to remove.
488  * pcroot      The alternate install root. Could be NULL. In this
489  *              case, assume root is '/'
490  *
491  * Result: 0 on success, nonzero on failure. If an error occurs, an
492  * error string will get output to standard error alerting the user.
493  * Side effects: The contents file may change as a result of this call,
494  * such that lines in the in the file will be changed or removed.
495  * If the call fails, a t.contents file may be left behind. This
496  * temporary file should be removed subsequently.
497 */
498 int
499 special_contents_remove(int ient, struct cfmt **ppcfent, const char *pcroot)
500 {
501     int result = 0; /* Assume we will succeed. Return result. */
502     char **ppcSC = NULL; /* The special contents rules, sorted. */
503     int i; /* Index into contents & special contents */
504     int j; /* Indexes into contents & special contents */
505     FILE *fpi = NULL, /* Input of contents file */
506         *fpo = NULL; /* Output to temp contents file */
507     char cpath[PATH_MAX], /* Contents file path */
508         tpath[PATH_MAX]; /* Temp contents file path */
509     const char *pccontents = "var/sadm/install/contents";
510     const char *pctcontents = "var/sadm/install/t.contents";
511     char line[LINESZ]; /* Reads in and writes out contents lines. */
512     time_t t; /* Used to create a timestamp comment. */
513     int max; /* Max number of special contents entries. */
514     int *piIndex; /* An index to ppcfents to remove from cfile */

515     cpath[0] = tpath[0] = '\0';
```



```

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

521     if ((get_special_contents(pcrout, &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) pcrout = "/";
532     if (pcroot[strlen(pcrout) - 1] == '/') {
533         if (snprintf(cpath, PATH_MAX, "%s%s", pcrout, pccontents)
534             >= PATH_MAX ||
535             snprintf(tcpath, PATH_MAX, "%s%s", pcrout, 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", pcrout, pccontents)
543             >= PATH_MAX ||
544             snprintf(tcpath, PATH_MAX, "%s/%s", pcrout, 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;
568     }

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

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

```

```

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 match 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; ) {
604         for (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(pccpath, ppcfent[i]))
650 >= 0) {
651         i++;
652         if (k == 0) {
653             free(pccpath);
654             (void) memset(line, 0, LINESZ);
655             break;
656         }
657     } else {
658         /* path < rule, end special case */
659         break;
660     }
661 }
662
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 }
670
671 if (fprintf(fpo, "%s", line) < 0) {
672     /* Failing to write output would be catastrophic. */
673     proterr(gettext(SPECIAL_ACCESS));
674     result = 1;
675     break;
676 }
677 (void) memset(line, 0, LINESZ);
678 }
679
680 t = time(NULL);
681 (void) fprintf(fpo, "# Last modified by pkgremove\n");
682 (void) fprintf(fpo, "# %s", ctime(&t));
683
684 remove_done:
685     free_special_contents(&ppcSC, max);
686
687     if (fpi != NULL)
688         (void) fclose(fpi);
689
690     if (fpo != NULL)
691         (void) fclose(fpo);
692
693     if (result == 0) {
694         if (tcpath[0] != '\0' && cpath[0] != '\0' &&
695             rename(tcpath, cpath) != 0) {
696             proterr(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     }
709
710     return (result);
711 }

```

unchanged_portion_omitted

new/usr/src/cmd/troff/nroff.d/n6.c

1

```
*****
7107 Sun Jul 16 20:04:40 2017
new/usr/src/cmd/troff/nroff.d/n6.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 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
29 /*      All Rights Reserved */

31 /*
32  * University Copyright- Copyright (c) 1982, 1986, 1988
33  * The Regents of the University of California
34  * All Rights Reserved
35  *
36  * University Acknowledgment- Portions of this document are derived from
37  * software developed by the University of California, Berkeley, and its
38  * contributors.
39  */

41 #include "tdef.h"
42 #include "tw.h"
43 #include "ext.h"
44 #include <ctype.h>

46 /*
47  * n6.c -- width functions, sizes and fonts
48  */

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

54 extern int      nchtab;

56 int
57 width(j)
58 tchar j;
59 {
60     int      i, k;
```

new/usr/src/cmd/troff/nroff.d/n6.c

2

```
62     if (j & (ZBIT|MOT)) {
63         if (iszbit(j))
64             return(0);
65         if (isvmot(j))
66             return(0);
67         k = absmot(j);
68         if (isnmot(j))
69             k = -k;
70         return(k);
71     }
72     i = cbits(j);
73     if (i < ' ') {
74         if (i == '\b')
75             return(-widthp);
76         if (i == PRESC)
77             i = eschar;
78         else if (iscontrol(i))
79             return(0);
80     }
81     if (i==ohc)
82         return(0);
83 #ifdef EUC
84 #ifdef NROFF
85     if (multi_locale) {
86         if ((j & MBMASK) || (j & CSMASK)) {
87             switch(j & MBMASK) {
88                 case BYTE_CHR:
89                 case LASTOFMB:
90                     k = t.Char * csi_width[cs(j)];
91                     break;
92                 default:
93                     k = 0;
94                     break;
95             }
96             widthp = k;
97             return(k);
98         }
99     }
100     i &= 0x1ff;
101 #endif /* NROFF */
102 #endif /* EUC */
103     i = trtab[i];
104     if (i < 32)
105         return(0);
106     k = t.width[i] * t.Char;
107     widthp = k;
108     return(k);
109 }
```

unchanged_portion_omitted

```
203 tchar setht()          /* set character height from \H'...' */
204 {
205     int      n;
206     tchar c;

207     getch();
208     (void) inumb(&apts);
209     n = inumb(&apts);
210     getch();
211     return(0);
212 }
```

unchanged_portion_omitted

new/usr/src/cmd/volcheck/volcheck.c

1

```
*****
2310 Sun Jul 16 20:04:40 2017
new/usr/src/cmd/volcheck/volcheck.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 (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22  * Copyright 2017 Gary Mills
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */

27 #include <stdio.h>
28 #include <stdlib.h>
29 #include <unistd.h>
30 #include <fcntl.h>
31 #include <string.h>
32 #include <strings.h>
33 #include <signal.h>
34 #include <errno.h>
35 #include <libintl.h>
36 #include <sys/types.h>

38 #include "vold.h"
39 #include "rmm_common.h"

41 char *progname = "volcheck";

43 static void
44 usage()
45 {
46     fprintf(stderr,
47         gettext("usage: %s [-t #secs -i #secs] [-v] [path | nickname]\n"),
48         progname);
49     fprintf(stderr,
50         gettext("If path is not supplied all media is checked\n"));
51 }

53 int
54 main(int argc, char **argv)
55 {
56     const char    *opts = "itv";
57     int           c;
58     boolean_t     opt_i = B_FALSE;
59     boolean_t     opt_t = B_FALSE;
60     boolean_t     opt_v = B_FALSE;
61     LibHalContext *hal_ctx;
```

new/usr/src/cmd/volcheck/volcheck.c

2

```
59     DBusError      error;
60     rmm_error_t     rmm_error;
61     int             ret = 0;

63     vold_init(argc, argv);

65     while ((c = getopt(argc, argv, opts)) != EOF) {
66         switch (c) {
67             case 'i':
68                 opt_i = B_TRUE;
69                 break;
70             case 't':
71                 opt_t = B_TRUE;
72                 break;
73             case 'v':
74                 opt_v = B_TRUE;
75                 break;
76             default:
77                 usage();
78                 return (1);
79         }
80     }

81     if ((hal_ctx = rmm_hal_init(0, 0, 0, 0, &error, &rmm_error)) == NULL) {
82         (void) fprintf(stderr,
83             gettext("HAL initialization failed: %s\n"),
84             rmm_strerror(&error, rmm_error));
85         rmm_dbus_error_free(&error);
86         return (1);
87     }

88     if (optind == argc) {
89         /* no name provided, check all */
90         ret = rmm_rescan(hal_ctx, NULL, B_FALSE);
91     } else {
92         for (; optind < argc; optind++) {
93             if (rmm_rescan(hal_ctx, argv[optind], B_FALSE) != 0) {
94                 ret = 1;
95             }
96         }
97     }

98     rmm_hal_fini(hal_ctx);

100     return (ret);
101 }

unchanged_portion_omitted
```

new/usr/src/cmd/ypcmd/ypserv.c

1

```
*****
13606 Sun Jul 16 20:04:40 2017
new/usr/src/cmd/ypcmd/ypserv.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 2004 Sun Microsystems, Inc. All rights reserved.
25 * Use is subject to license terms.
26 */

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

31 /*
32 * Portions of this source code were derived from Berkeley 4.3 BSD
33 * under license from the Regents of the University of California.
34 */

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

36 /*
37 * This contains the mainline code for the YP server.  Data
38 * structures which are process-global are also in this module.
39 */

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

46 #include "ypsym.h"
47 #include <sys/types.h>
48 #include <sys/wait.h>
49 #include <fcntl.h>
50 #include <rpc/rpc.h>
51 #include <netconfig.h>
52 #include <netdir.h>
53 #include <sys/select.h>
54 #include <stdlib.h>
55 #include <unistd.h>
56 #include <stdio.h>
57 #include <stdarg.h>
58 #include <signal.h>
59 #include "shim.h"
```

new/usr/src/cmd/ypcmd/ypserv.c

2

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

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

66 /*
67  * client_setup_failure will be TRUE, if setup of the
68  * connection to rpc.nisd_resolv failed
69 */
70 bool client_setup_failure = FALSE;

72 /* N2L options */
73 bool init_dit = FALSE;
74 bool init_containers = FALSE;
75 bool init_maps = FALSE;
76 char **ldapCLA = NULL;

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

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

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

92 static void ypexit(void);
93 static void ypinit(int argc, char **argv);
94 static void ypdspatch(struct svc_req *rqstp, SVCXPRT *transp);
95 static void ypolddispatch(struct svc_req *rqstp, SVCXPRT *transp);
96 static void ypget_command_line_args(int argc, char **argv);
97 extern void setup_resolv(bool *fwding, int *child,
98                          CLIENT **client, char *tp_type, long prognum);
99 static void cleanup_resolv(int);

101 /*
102  * This is the main line code for the yp server.
103  */
104 int
105 main(int argc, char **argv)
106 {
107     if (geteuid() != 0) {
108         fprintf(stderr, "must be root to run %s\n", argv[0]);
109         exit(1);
110     }

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

115     /* If requested set up the N2L maps. May take a while */
116     if (init_dit)
117         if (FAILURE == dump_maps_to_dit(init_containers)) {
118             fprintf(stderr, "Fatal error dumping maps to DIT."
119                     " See syslog and LDAP server logs for details.\n");
120             exit(1);
121         }

123     if (init_maps)
124         if (FAILURE == dump_dit_to_maps()) {
125             fprintf(stderr, "Fatal error dumping DIT to maps."
126                     " See syslog and LDAP server logs for details.\n");
127             exit(1);
128         }
129 }
```

```

126         " See syslog and LDAP server logs for details.\n");
127         exit(1);
128     }
129
130     /*
131     * If we were asked to init the maps now exit. User will then use
132     * yptest 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 yptest.\n");
137         exit(0);
138     }
139
140     svc_run();
141
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]))
166
167 int      service_classes[MAXSERVICES];
168
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;
184
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();
190
191     ypget_command_line_args(argc, argv);
192
193     if (silent) {
194         pid = (int)fork();
195
196         if (pid == -1) {
197             logprintf("ypserv: ypinit fork failure.\n");
198             ypexit();
199         }
200
201         if (pid != 0) {
202             exit(0);
203         }
204     }

```

```

206     if (!init_lock_system(FALSE)) {
207         ypexit();
208     }
209
210     get_secure_nets(argv[0]);
211
212     if (silent) {
213         closelog();
214         closefrom(3);
215     }
216
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     }
228
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         }
238     }
239     (void) open("/dev/tty", 2);
240     t = open("/dev/tty", 2);
241
242     setpgpr();
243
244 #ifdef SYSVCONFIG
245     sigset(SIGHUP, (void (*)())sysvconfig);
246 #else
247     sigset(SIGHUP, SIG_IGN);
248 #endif
249
250     /*
251     * Setting disposition to SIG_IGN will not create zombies when child
252     * processes terminate.
253     */
254     sigset(SIGCHLD, SIG_IGN);
255
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);
264
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].xpirt = 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].xpirt, YPPROG, YPVERS, ypdispatch,
314             nconf)) {
315             logprintf("%s %s\n", service[i].netid, register_failed);
316             svc_destroy(service[i].xpirt);
317             service[i].xpirt = 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].xpirt, 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].xpirt);
330             service[i].xpirt = 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