

```

new/usr/src/lib/libresolv/Makefile
*****
2080 Thu Jul 9 15:59:30 2015
new/usr/src/lib/libresolv/Makefile
1926 libresolv evades compiler warnings
*****
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 2015 Gary Mills
23 # Copyright 2006 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #

27 LIBRARY= libresolv.a
28 VERS= .1

30 PICS=   pics/res_comp.o pics/res_debug.o pics/res_init.o \
31     pics/res_mkquery.o pics/res_query.o pics/res_send.o \
32     pics/res_gethost.o pics/res_sethost.o

34 pics/%.o: %.c
35     $(COMPILE.c) -o $@ $(
36     $(POST_PROCESS_O)

38 OBJECTS= \
39 res_gethost.o    res_comp.o      res_debug.o      res_init.o      res_mkquery.o \
40 res_query.o      res_send.o      res_sethost.o

42 # include library definitions
43 include ./Makefile.lib

45 # install this library in the root filesystem
46 include ./Makefile.rootfs

48 SRCDIR =
49 .
50 C99MODE=      $(C99_DISABLE)

52 # We really want to say this:
53 #     CPPFLAGS += -DDEBUG -DSYSV -D_REENTRANT -I. -I../common/inc
54 # but some system header files are replaced by local versions
55 # so we must put -I. ahead of the default include directories:
56 CPPFLAGS = -I. -I../common/inc $(CPPFLAGS.master) -DDEBUG -DSYSV -D_REENTRANT
57 LDLIBS += -lsocket -lns1 -lc
58 CFLAGS += $(CCVERBOSE)

58 CERRWARN += -_gcc=-Wno-implicit-function-declaration
60 CERRWARN += -_gcc=-Wno-parentheses

```

```

1
new/usr/src/lib/libresolv/Makefile
*****
60 CERRWARN += -_gcc=-Wno-unused-variable
61 CERRWARN += -_gcc=-Wno-uninitialized
62 CERRWARN += -_gcc=-Wno-implicit-int
63 CERRWARN += -_gcc=-Wno-extra

63 ROOTDYNLIBS=      $(DYNLIB:%=$(ROOTLIBDIR)/%)
65 .KEEP_STATE:
67 LIBS = $(DYNLIB)
69 all: $(LIBS)
71 install: all $(ROOTDYNLIBS)
73 lint: lintcheck
75 # include library targets
76 include ./Makefile.targ

```

```
*****
1309 Thu Jul  9 15:59:31 2015
new/usr/src/lib/libresolv/crossl.h
1926 libresolv evades compiler warnings
*****
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */
12 /*
13  * Copyright 2015 Gary Mills
14 */
16 #ifndef _CROSSL_H
17 #define _CROSSL_H
19 /*
20  * Definitions needed for cross-linkages between source files
21 */
23 #ifdef __cplusplus
24 extern "C" {
25 #endif
27 extern int
28 dn_comp(u_char *, u_char *, int, u_char **, u_char **);
29 extern int
30 dn_expand(u_char *, u_char *, u_char *, u_char *, int);
31 extern int
32 dn_skipname(u_char *, u_char *);
34 extern int
35 res_init(void);
36 extern int
37 res_mkquery(int, char *, int, int, char *, int, struct rrec *, char *, int);
38 extern int
39 res_query(char *, int, int, u_char *, int);
40 extern int
41 res_querydomain(char *, char *, int, int, u_char *, int);
42 extern int
43 res_search(char *, int, int, u_char *, int);
44 extern int
45 res_send(char *, int, char *, int);
47 extern void
48 putlong(u_long, u_char *);
49 extern void
50 putshort(u_short, u_char *);
51 extern void
52 p_query(char *);
53 extern void
54 _res_close();
57 #ifdef __cplusplus
58 }
59 #endif
61 #endif /* _CROSSL_H */
```

new/usr/src/lib/libresolv/res\_debug.c

```
*****
10957 Thu Jul  9 15:59:31 2015
new/usr/src/lib/libresolv/res_debug.c
1926 libresolv evades compiler warnings
*****
```

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 2015 Gary Mills  
24 \* Copyright 2008 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 \*/  
  
40 #pragma ident "%Z%%M% %I% %E% SMI"  
  
41 #include <sys/types.h>  
42 #include <sys/socket.h>  
43 #include <netinet/in.h>  
44 #include <arpa/inet.h>  
45 #include <stdio.h>  
46 #include <string.h>  
47 #include <arpa/nameser.h>  
48 #include <resolv.h>  
49 #include "crossl.h"  
  
47 extern char \*p\_cdname(), \*p\_rr(), \*p\_type(), \*p\_class(), \*p\_time();  
48 extern char \*inet\_ntoa();  
51 void fp\_query(char \*msg, FILE \*file);  
  
53 char \*\_res\_opcodes[] = {  
54 "QUERY",  
55 "IQUERY",  
56 "CQUERYM",  
57 "CQUERYU",  
58 "4",  
59 "5",  
60 "6",  
61 "7",  
62 "8",  
63 "UPDATEA",  
64 "UPDATED",  
65 "UPDATEDA",  
66 "UPDATEM",  
67 "UPDATEMA",  
68 "ZONEINIT",  
69 "ZONEREF",  
70 };  
\_\_\_\_\_

1

new/usr/src/lib/libresolv/res\_debug.c

```
58     "4",  
59     "5",  
60     "6",  
61     "7",  
62     "8",  
63     "UPDATEA",  
64     "UPDATED",  
65     "UPDATEDA",  
66     "UPDATEM",  
67     "UPDATEMA",  
68     "ZONEINIT",  
69     "ZONEREF",  
70 };  
_____
```

192 char \*  
193 p\_cdname(cp, msg, file)  
194 char \*cp, \*msg;  
195 FILE \*file;  
196 {  
197 char name[MAXDNAME];  
198 int n;  
199  
200 if ((n = dn\_expand((u\_char \*)msg, (u\_char \*)(msg + 512), (u\_char \*)cp,  
201 (u\_char \*)name, sizeof (name))) < 0)  
202 if ((n = dn\_expand(msg, msg + 512, cp, name, sizeof (name))) < 0)  
203 return (NULL);  
204 if (name[0] == '.') {  
205 name[0] = '\0';  
206 name[1] = '\0';  
207 }  
208 fputs(name, file);  
209 }  
\_\_\_\_\_

2

new/usr/src/lib/libresolv/res\_gethost.c

```
*****
9953 Thu Jul  9 15:59:31 2015
new/usr/src/lib/libresolv/res_gethost.c
1926 libresolv evades compiler warnings
*****
1 /*
2 * Copyright 2015 Gary Mills
3 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
4 * Use is subject to license terms.
5 */

7 /*
8 * Copyright (c) 1985, 1988 Regents of the University of California.
9 * All rights reserved.
10 */
11 * Redistribution and use in source and binary forms are permitted
12 * provided that this notice is preserved and that due credit is given
13 * to the University of California at Berkeley. The name of the University
14 * may not be used to endorse or promote products derived from this
15 * software without specific prior written permission. This software
16 * is provided 'as is' without express or implied warranty.
17 */
18 */

20 #include <sys/param.h>
21 #include <sys/socket.h>
22 #include <netinet/in.h>
23 #include <ctype.h>
24 #include <netdb.h>
25 #include <stdio.h>
26 #include <errno.h>
27 #include <string.h>
28 #include <arpa/inet.h>
29 #include <arpa/nameser.h>
30 #include <resolv.h>
31 #include <syslog.h>
32 #include "crosssl.h"

34 /*
35 * When the name service switch calls libresolv, it doesn't want fallback
36 * to /etc/hosts, so we provide a method to turn it off.
37 */
38 static int no_hostsFallback = 0;

40 void
41 __res_set_no_hostsFallback(void) {
42     no_hostsFallback = 1;
43 }
unchanged portion omitted

80 int h_errno;

82 static struct hostent *
83 getanswer(answer, anslen, iquery)
84     querybuf *answer;
85     int anslen;
86     int iquery;
87 {
88     register HEADER *hp;
89     register u_char *cp;
90     register int n;
91     u_char *eom;
92     char *bp, **ap;
93     int type, class, buflen, ancount, qdcount;
94     int haveanswer, getClass = C_ANY;
```

1

new/usr/src/lib/libresolv/res\_gethost.c

```
95     char **hap;
96
97     eom = answer->buf + anslen;
98     /*
99      * find first satisfactory answer
100     */
101    hp = &answer->hdr;
102    ancount = ntohs(hp->ancount);
103    qdcount = ntohs(hp->qdcount);
104    bp = hostbuf;
105    buflen = sizeof(hostbuf);
106    cp = answer->buf + sizeof(HEADER);
107    if (qdcount) {
108        if (iquery) {
109            if ((n = dn_expand(answer->buf, eom,
110                               (u_char *)bp, buflen)) < 0) {
111                if ((n = dn_expand((char *)answer->buf, eom,
112                                   cp, bp, buflen)) < 0) {
113                    h_errno = NO_RECOVERY;
114                    return ((struct hostent *) NULL);
115                }
116                cp += n + QFIXEDSZ;
117                host.h_name = bp;
118                n = strlen(bp) + 1;
119                bp += n;
120                buflen -= n;
121            } else
122                cp += dn_skipname(cp, eom) + QFIXEDSZ;
123        } else if (iquery) {
124            if (hp->aa)
125                h_errno = HOST_NOT_FOUND;
126            else
127                h_errno = TRY AGAIN;
128            return ((struct hostent *) NULL);
129        }
130        ap = host_aliases;
131        host.h_aliases = host_aliases;
132        hap = h_addr_ptrs;
133        #if BSD >= 43 || defined(h_addr) /* new-style hostent structure */
134        host.h_addr_list = h_addr_ptrs;
135        #endif
136        haveanswer = 0;
137        while (--ancount >= 0 && cp < eom && haveanswer < MAXADDRS) {
138            if ((n = dn_expand(answer->buf, eom,
139                               cp, (u_char *)bp, buflen)) < 0)
140                if ((n = dn_expand((char *)answer->buf, eom,
141                                   cp, bp, buflen)) < 0)
142                    break;
143                cp += n;
144                type = _getshort(cp);
145                cp += sizeof(u_short);
146                class = _getshort(cp);
147                cp += sizeof(u_short) + sizeof(u_long);
148                n = _getshort(cp);
149                cp += sizeof(u_short);
150                if (type == T_CNAME) {
151                    cp += n;
152                    if (ap >= &host_aliases[MAXALIASES-1])
153                        continue;
154                    *ap++ = bp;
155                    n = strlen(bp) + 1;
156                    bp += n;
157                    buflen -= n;
158                    continue;
159                }
160            }
161        }
162    }
163 }
```

2

```

157         }
158     if (iquery && type == T_PTR) {
159         if ((n = dn_expand(answer->buf, eom,
160                             cp, (u_char *)bp, buflen)) < 0) {
161             if ((n = dn_expand((char *)answer->buf, eom,
162                                 cp, bp, buflen)) < 0) {
163                 cp += n;
164                 continue;
165             }
166             cp += n;
167             host.h_name = bp;
168             return (&host);
169 #ifdef DEBUG
170         if (iquery || type != T_A) {
171             if (_res.options & RES_DEBUG)
172                 printf("unexpected answer type %d, size %d\n",
173                         type, n);
174         }
175         cp += n;
176         continue;
177     }
178     if (haveanswer) {
179         if (n != host.h_length) {
180             cp += n;
181             continue;
182         }
183         if (class != getclass) {
184             cp += n;
185             continue;
186         }
187     } else {
188         host.h_length = n;
189         getclass = class;
190         host.h_addrtype = (class == C_IN) ? AF_INET : AF_UNSPEC;
191         if (!iquery) {
192             host.h_name = bp;
193             bp += strlen(bp) + 1;
194         }
195     }
196     bp += sizeof (align) - ((u_long)bp % sizeof (align));
197     if (bp + n >= &hostbuf[sizeof (hostbuf)]) {
198 #ifdef DEBUG
199         if (_res.options & RES_DEBUG)
200             printf("size (%d) too big\n", n);
201     }
202 #endif
203     break;
204 }
205 #ifdef SYSV
206 memcpy((void *)(*hap++ = bp), (void *)cp, n);
207 #else
208 bcopy(cp, *hap++ = bp, n);
209 #endif
210 bp += n;
211 cp += n;
212 haveanswer++;
213 }
214 if (haveanswer) {
215     *ap = NULL;
216 #if BSD >= 43 || defined(h_addr) /* new-style hostent structure */
217     *hap = NULL;
218 #else
219     host.h_addr = h_addr_ptrs[0];
220 #endif

```

```

221         return (&host);
222     } else {
223         h_errno = TRY AGAIN;
224         return ((struct hostent *) NULL);
225     }
226 }

228 static struct hostent *_gethtbyname();

229 struct hostent *
230 res_gethostbyname(name)
231     char *name;
232 {
233     querybuf buf;
234     register char *cp;
235     int n;
236     struct hostent *hp, *gethostdomain();

237     /*
238      * disallow names consisting only of digits/dots, unless
239      * they end in a dot.
240      */
241     if (isdigit(name[0]))
242         for (cp = name; /*EMPTY*/; ++cp) {
243             if (!*cp) {
244                 if (--cp == '.')
245                     break;
246                 h_errno = HOST_NOT_FOUND;
247                 return ((struct hostent *) NULL);
248             }
249             if (!isdigit(*cp) && *cp != '.')
250                 break;
251         }
252     }

253     if ((n = res_search(name, C_IN, T_A, buf.buf, sizeof (buf))) < 0) {
254 #ifdef DEBUG
255         if (_res.options & RES_DEBUG)
256             printf("res_search failed\n");
257     }
258 #endif
259     if (errno == ECONNREFUSED)
260         return (_gethtbyname(name));
261     else
262         return ((struct hostent *) NULL);
263 }
264 return (getanswer(&buf, n, 0));
265 }

266 static struct hostent *_gethtbyaddr();

267 static struct hostent *
268 _getrhbyaddr(addr, len, type)
269     char *addr;
270     int len, type;
271 {
272     int n;
273     querybuf buf;
274     register struct hostent *hp;
275     char qbuf[MAXDNAME];

276     if (type != AF_INET)
277         return ((struct hostent *) NULL);
278     (void) sprintf(qbuf, "%d.%d.%d.%d.in-addr.arpa",
279                   ((unsigned)addr[3] & 0xff),
280                   ((unsigned)addr[2] & 0xff),
281                   ((unsigned)addr[1] & 0xff),
282                   ((unsigned)addr[0] & 0xff));

```

```
286         n = res_query(qbuf, C_IN, T_PTR, (u_char *)&buf, sizeof (buf));
284         n = res_query(qbuf, C_IN, T_PTR, (char *)&buf, sizeof (buf));
287         if (n < 0) {
288 #ifdef DEBUG
289             if (_res.options & RES_DEBUG)
290                 printf("res_query failed\n");
291 #endif
292             if (errno == ECONNREFUSED)
293                 return (_gethtbyaddr(addr, len, type));
294             return ((struct hostent *) NULL);
295         }
296         hp = getanswer(&buf, n, 1);
297         if (hp == NULL)
298             return ((struct hostent *) NULL);
299         hp->h_addrtype = type;
300         hp->h_length = len;
301         h_addr_ptrs[0] = (char *)&host_addr;
302         h_addr_ptrs[1] = (char *)0;
303         host_addr = *(struct in_addr *)addr;
304     }
305 }
```

unchanged portion omitted

```
new/usr/src/lib/libresolv/res_init.c
```

```
*****
7938 Thu Jul  9 15:59:31 2015
new/usr/src/lib/libresolv/res_init.c
1926 libresolv evades compiler warnings
*****
```

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 2015 Gary Mills  
24 \* Copyright 2008 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 \*/

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

41 #include <sys/types.h>  
42 #include <sys/sockio.h>  
43 #include <sys/socket.h>  
44 #include <netinet/in.h>  
45 #include <stdio.h>  
46 #include <string.h>  
47 #include <stdlib.h>  
48 #include <unistd.h>  
49 #include <stropts.h>  
50 #include <arpa/nameser.h>  
51 #include <resolv.h>

53 #include <netinet/in.h>  
54 #include <net/if.h>  
55 #include <netinet/if\_ether.h>  
56 #include <arpa/inet.h>

58 /\*  
59 \* Undocumented external function in libns1

```
1
```

```
new/usr/src/lib/libresolv/res_init.c
```

60 \*/  
61 extern int  
62 getdomainname(char \*, int);

64 #define MAXIFS 256

66 /\*  
67 \* Resolver state default settings  
68 \*/

70 struct state \_res = {  
71 RES\_TIMEOUT, /\* retransmission time interval \*/  
72 4, /\* number of times to retransmit \*/  
73 RES\_DEFAULT, /\* options flags \*/  
74 1, /\* number of name servers \*/  
75 };

77 /\*  
78 \* Set up default settings. If the configuration file exist, the values  
79 \* there will have precedence. Otherwise, the server address is set to  
80 \* INADDR\_LOOPBACK and the default domain name comes from the gethostname().  
81 \* BUT if the NIS/RPC domain name is set, that is used if all else fails.  
82 \*  
83 \* The configuration file should only be used if you want to redefine your  
84 \* domain or run without a server on your machine.  
85 \*  
86 \* Note the user can always override then domain name with the environment  
87 \* variable LOCALDOMAIN which has absolute priority.  
88 \*  
89 \*  
90 \* Return 0 if completes successfully, -1 on error  
91 \*/

92 int  
93 res\_init(void)  
94 {  
95 register FILE \*fp;  
96 register char \*cp, \*\*pp;  
97 register int n;  
98 char buf[BUFSIZ];

99 #ifdef SYSV  
100 extern char \*strchr();  
101 #else  
102 extern char \*index();  
103 #endif  
104 extern char \*strcpy(), \*strncpy();  
105 extern char \*getenv();  
106 int nserv = 0; /\* number of nameserver records read from file \*/  
107 int haveenv = 0;  
108 int havesearch = 0;

109 \_res.nsaddr.sin\_addr = htonl(INADDR\_ANY); /\* INADDR\_ANY \*/  
110 \_res.nsaddr.sin\_family = AF\_INET;  
111 \_res.nsaddr.sin\_port = htons(NAMESERVER\_PORT);  
112 \_res.nscount = 1;

113 ifdef SIOCGIFNUM  
114 {  
115 int numifs, s, n, int\_up;  
116 struct ifconf ifc;  
117 register struct ifreq \*ifrp;  
118 struct ifreq ifr;  
119 unsigned bufsize;  
120 unsigned int flags;  
121 char \*buf;  
122 extern void \*malloc();

123 if ((s = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

```
2
```

```

118         perror("socket");
119         return (-1);
120     }
121     if (ioctl(s, SIOCGIFNUM, (char *)&numifs) < 0) {
122         numifs = MAXIFS;
123     }
124     bufsize = numifs * sizeof (struct ifreq);
125     buf = (char *)malloc(bufsize);
126     if (buf == NULL) {
127         perror("out of memory");
128         (void) close(s);
129         close(s);
130         return (-1);
131     }
132     ifc.ifc_len = bufsize;
133     ifc.ifc_buf = buf;
134     if (ioctl(s, SIOCGIFCONF, (char *)&ifc) < 0) {
135         perror("ifconfig: SIOCGIFCONF");
136         (void) close(s);
137         close(s);
138         free(buf);
139         return (-1);
140     }
141     int_up = 0;
142     ifrp = ifc.ifc_req;
143     for (n = ifc.ifc_len / sizeof (struct ifreq); n > 0;
144          n--, ifrp++) {
145         (void) memset((void *) &ifr, 0, sizeof (ifr));
146         memset((void *) &ifr, 0, sizeof (ifr));
147         strncpy(ifr.ifr_name, ifrp->ifr_name,
148                 sizeof (ifr.ifr_name));
149         if (ioctl(s, SIOCGIFFLAGS, (char *)&ifr) < 0) {
150             perror("SIOCGIFFLAGS");
151             (void) close(s);
152             close(s);
153             free(buf);
154             return (-1);
155         }
156         flags = ifr.ifr_flags;
157         /* we are looking for a non-loopback interface */
158         if ((flags & IFF_UP) && ((flags & IFF_LOOPBACK) == 0))
159             int_up = 1;
160     }
161     (void) close(s);
162     close(s);
163     free(buf);
164     if (int_up == 0) /* all the non-LOOPBACK interfaces are DOWN */
165         return (-1);
166 }
167 /* for the benefit of hidden NIS domains, we use the same procedure
168 * as sendmail: convert leading + to dot, then drop to first dot
169 */
170 (void) getdomainname(buf, BUFSIZ);
171 getdomainname(buf, BUFSIZ);
172 if (buf[0] == '+')
173     buf[0] = '.';
174 #ifdef SYSV
175     cp = strchr(buf, (int)'.');
176 #else
177     cp = index(buf, '.');
178 
```

```

179     if (cp == NULL)
180         strcpy(_res.defdname, buf);
181     else
182         strcpy(_res.defdname, cp+1);
183     /* Allow user to override the local domain definition */
184     if ((cp = getenv("LOCALDOMAIN")) != NULL) {
185         (void) strcpy(_res.defdname, cp, sizeof (_res.defdname));
186         haveenv++;
187     }
188     if ((fp = fopen(_PATH_RESCONF, "r")) != NULL) {
189         /* read the config file */
190         while (fgets(buf, sizeof (buf), fp) != NULL) {
191             /* read default domain name */
192             if (!strcmp(buf, "domain", sizeof ("domain") - 1)) {
193                 if (haveenv) /* skip if have from environ */
194                     continue;
195                 cp = buf + sizeof ("domain") - 1;
196                 while (*cp == ' ' || *cp == '\t')
197                     cp++;
198                 if ((*cp == '\0') || (*cp == '\n'))
199                     continue;
200                 (void) strncpy(_res.defdname, cp, sizeof (_res.defdname) - 1);
201             #ifdef SYSV
202             if ((cp = strchr(_res.defdname, (int)'\'\n')) != NULL)
203                 if ((cp = index(_res.defdname, '\n')) != NULL)
204                     *cp = '\0';
205             #endif
206             havesearch = 0;
207             continue;
208         }
209     }
210     /* set search list */
211     if (!strcmp(buf, "search", sizeof ("search") - 1)) {
212         if (haveenv) /* skip if have from environ */
213             continue;
214         cp = buf + sizeof ("search") - 1;
215         while (*cp == ' ' || *cp == '\t')
216             cp++;
217         if ((*cp == '\0') || (*cp == '\n'))
218             continue;
219         (void) strncpy(_res.defdname, cp, sizeof (_res.defdname) - 1);
220     }
221     #ifdef SYSV
222     if ((cp = strchr(_res.defdname, (int)'\'\n')) != NULL)
223         if ((cp = index(_res.defdname, '\n')) != NULL)
224             *cp = '\0';
225     #endif
226     /*
227      * Set search list to be blank-separated strings
228      * on rest of line.
229      */
230     cp = _res.defdname;
231     pp = _res.dnsrch;
232     *pp++ = cp;
233     for (n = 0; *cp && pp < _res.dnsrch + MAXDNSRCH; cp++) {
234         if (*cp == ' ' || *cp == '\t') {
235             *cp = 0;
236             n = 1;
237         } else if (n) {
238             *pp++ = cp;
239             n = 0;
240         }
241     }
242     /* null terminate last domain if there are excess */
243 
```

```
244         while (*cp != '\0' && *cp != ' ' && *cp != '\t')
245             cp++;
246         *cp = '\0';
247         *pp++ = 0;
248         haveSearch = 1;
249         continue;
250     }
251     /* read nameservers to query */
252     if (!strcmp(buf, "nameserver", sizeof ("nameserver") - 1) &&
253         (nserv < MAXNS)) {
254         cp = buf + sizeof ("nameserver") - 1;
255         while (*cp == ' ' || *cp == '\t')
256             cp++;
257         if ((*cp == '\0') || (*cp == '\n'))
258             continue;
259         if (_res.nsaddr_list[nserv].sin_addr.s_addr ==
260             inet_addr(cp)) == (unsigned) -1) {
261             _res.nsaddr_list[n].sin_addr.s_addr = INADDR_ANY;
262             continue;
263         }
264         _res.nsaddr_list[nserv].sin_family = AF_INET;
265         _res.nsaddr_list[nserv].sin_port = htons(NAMESERVER_PORT);
266         nserv++;
267         continue;
268     }
269 }
270 if (nserv > 1)
271     _res.nscount = nserv;
272 (void) fclose(fp);
273 }
274 if (_res.defdname[0] == 0) {
275     if (gethostname(buf, sizeof (_res.defdname)) == 0 &&
276 #ifdef SYSV
277         (cp = strchr(buf, (int)'.')))
278 #else
279         (cp = index(buf, '.')))
280 #endif
281     (void) strcpy(_res.defdname, cp + 1);
282 }
283 /* find components of local domain that might be searched */
284 if (haveSearch == 0) {
285     pp = _res.dnsrch;
286     *pp++ = _res.defdname;
287     for (cp = _res.defdname, n = 0; *cp; cp++)
288         if (*cp == '.')
289             n++;
290     cp = _res.defdname;
291     for (; n >= LOCALDOMAINPARTS && pp < _res.dnsrch + MAXDFLSRCH; n--) {
292 #ifdef SYSV
293     cp = strchr(cp, (int)'.');
294 #else
295     cp = index(cp, '.');
296 #endif
297     *pp++ = ++cp;
298     }
299     *pp++ = 0;
300     }
301     _res.options |= RES_INIT;
302     return (0);
303 }
304 }  
unchanged portion omitted
```

```
new/usr/src/lib/libresolv/res_mkquery.c
```

```
*****  
7788 Thu Jul 9 15:59:31 2015  
new/usr/src/lib/libresolv/res_mkquery.c  
1926 libresolv evades compiler warnings  
*****  
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 2015 Gary Mills  
24  * Copyright 2008 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 */  
  
40 #pragma ident "%Z%%M% %I%     %E% SMI"  
  
41 #include <stdio.h>  
42 #include <sys/types.h>  
43 #include <sys/socket.h>  
44 #include <sys/stat.h>  
45 #include <netinet/in.h>  
46 #include <arpa/nameser.h>  
47 #include <resolv.h>  
48 #include <string.h>  
49 #include <stdlib.h>  
50 #include <unistd.h>  
51 #include <errno.h>  
52 #include <netdb.h>  
53 #include "crossl.h"  
  
55 /*  
56  * Kludge to time out quickly if there is no /etc/resolv.conf  
57  * and a TCP connection to the local DNS server fails.  
58 *  
59  * Moved function from res_send.c to res_mkquery.c. This
```

```
1
```

```
new/usr/src/lib/libresolv/res_mkquery.c
```

```
60  * solves a long timeout problem with nslookup.  
61  *  
62  * __areweinnamed is needed because there is a possibility that the  
63  * user might do bad things to resolv.conf and cause in.named to call  
64  * _confcheck and deadlock the server.  
65 */  
  
66 int __areweinnamed()  
67 {  
68     return (0);  
69 }  
  
70 }  
  
71 static int _confcheck()  
72 {  
73     int ns;  
74     struct stat rc_stat;  
75     struct sockaddr_in ns_sin;  
  
76     /* First, we check to see if /etc/resolv.conf exists.  
77      * If it doesn't, then localhost is mostlikely to be  
78      * the nameserver.  
79      */  
80     if (stat(_PATH_RESCONF, &rc_stat) == -1 && errno == ENOENT) {  
81         /* Next, we check to see if _res.nsaddr is set to loopback.  
82          * If it isn't, it has been altered by the application  
83          * explicitly and we then want to bail with success.  
84          */  
85     if (__areweinnamed())  
86         return (0);  
87     if (_res.nsaddr.sin_addr.S_un.S_addr == htonl(INADDR_LOOPBACK))  
88         /* Lastly, we try to connect to the TCP port of the  
89          * nameserver. If this fails, then we know that  
90          * DNS is misconfigured and we can quickly exit.  
91          */  
92     ns = socket(AF_INET, SOCK_STREAM, 0);  
93     IN_SET_LOOPBACK_ADDR(&ns_sin);  
94     ns_sin.sin_port = htons(NAMESEVER_PORT);  
95     if (connect(ns, (struct sockaddr *)&ns_sin,  
96                 sizeof ns_sin) == -1) {  
97         (void) close(ns);  
98         close(ns);  
99         return(-1);  
100    }  
101    else {  
102        (void) close(ns);  
103        close(ns);  
104        return(0);  
105    }  
106    }  
107    }  
108    }  
109    }  
110    }  
111    }  
112    }  
113    }  
114    }  
115    return (0);  
116 }  
  
117 /*  
118  * Form all types of queries.  
119  * Returns the size of the result or -1.  
120  */  
121 /*  
122 int  
123 res_mkquery(op, dname, class, type, data, datalen, newrr, buf, buflen)
```

```
2
```

```

124     int op;           /* opcode of query */
125     char *dname;      /* domain name */
126     int class, type; /* class and type of query */
127     char *data;       /* resource record data */
128     int datalen;     /* length of data */
129     struct rrec *newrr; /* new rr for modify or append */
130     char *buf;        /* buffer to put query */
131     int buflen;       /* size of buffer */
132 {
133     register HEADER *hp;
134     register u_char *cp;
135     register char *cp;
136     register int n;
137     u_char *dnptrs[10], **dpp, **lastdnptr;
138     char *dnptrs[10], **dpp, **lastdnptr;
139
140 #ifdef DEBUG
141     if (_res.options & RES_DEBUG)
142         printf("res_mkquery(%d, %s, %d, %d)\n", op, dname, class, type);
143 #endif /* DEBUG */
144
145     /*
146      * Check to see if we can bailout quickly.
147      * Also rerun res_init if we failed in the past.
148
149     if ((-_res.options & RES_INIT) == 0 && res_init() == -1) {
150         h_errno = NO_RECOVERY;
151         return(-1);
152     }
153
154     if (_confcheck() == -1) {
155         _res.options &= ~RES_INIT;
156         h_errno = NO_RECOVERY;
157         return(-1);
158     }
159
160     /*
161      * Initialize header fields.
162
163     if ((buf == NULL) || (buflen < sizeof (HEADER)))
164 #ifdef SYSV
165         (void) memset(buf, 0, sizeof (HEADER));
166     else
167         bzero(buf, sizeof (HEADER));
168 #endif
169     hp = (HEADER *) buf;
170     hp->id = htons(++_res.id);
171     hp->opcode = op;
172     hp->pr = (_res.options & RES_PRIMARY) != 0;
173     hp->rd = (_res.options & RES_RECURSE) != 0;
174     hp->rcode = NOERROR;
175     cp = (u_char*)(buf + sizeof (HEADER));
176     cp = buf + sizeof (HEADER);
177     buflen -= sizeof (HEADER);
178     dpp = dnptrs;
179     *dpp++ = (u_char*)buf;
180     *dpp++ = buf;
181     *dpp++ = NULL;
182     lastdnptr = dnptrs + sizeof (dnptrs) / sizeof (dnptrs[0]);
183
184     /*
185      * perform opcode specific processing
186
187     switch (op) {

```

```

185     case QUERY:
186         if ((buflen -= QFIXEDSZ) < 0)
187             return (-1);
188         if ((n = dn_comp((u_char *)dname, cp, buflen,
189                         dnptrs, lastdnptr)) < 0)
190             return (-1);
191         cp += n;
192         buflen -= n;
193         putshort(type, cp);
194         cp += sizeof (u_short);
195         putshort(class, cp);
196         cp += sizeof (u_short);
197         hp->qdcount = htons(1);
198         if (op == QUERY || data == NULL)
199             break;
200
201         /*
202          * Make an additional record for completion domain.
203          */
204         buflen -= RRFIXEDSZ;
205         if ((n = dn_comp((u_char *)data, cp, buflen,
206                         dnptrs, lastdnptr)) < 0)
207             return (-1);
208         cp += n;
209         buflen -= n;
210         putshort(T_NULL, cp);
211         cp += sizeof (u_short);
212         putshort(class, cp);
213         cp += sizeof (u_short);
214         putlong(0, cp);
215         cp += sizeof (u_long);
216         putshort(0, cp);
217         cp += sizeof (u_short);
218         hp->arcount = htons(1);
219         break;
220
221     case IQUERY:
222         /*
223          * Initialize answer section
224          */
225         if (buflen < 1 + RRFIXEDSZ + datalen)
226             return (-1);
227         *cp++ = '\0'; /* no domain name */
228         putshort(type, cp);
229         cp += sizeof (u_short);
230         putshort(class, cp);
231         cp += sizeof (u_short);
232         putlong(0, cp);
233         cp += sizeof (u_long);
234         putshort(datalen, cp);
235         cp += sizeof (u_short);
236 #ifdef SYSV
237         (void) memcpy((void *)cp, (void *)data, datalen);
238     else
239         bcopy(data, cp, datalen);
240 #endif
241         cp += datalen;
242     }
243     hp->arcount = htons(1);
244     break;
245
246 #ifdef ALLOW_UPDATES
247     /*

```

```

248         * For UPDATEM/UPDATEMA, do UPDATED/UPDATEDA followed by UPDATEA
249         * (Record to be modified is followed by its replacement in msg.)
250         */
251     case UPDATEM:
252     case UPDATEMA:
253
254     case UPDATED:
255     /*
256         * The res code for UPDATED and UPDATEDA is the same; user
257         * calls them differently: specifies data for UPDATED; server
258         * ignores data if specified for UPDATEDA.
259         */
260     case UPDATEDA:
261         buflen -= RRFIXEDSZ + datalen;
262         if ((n = dn_comp((u_char *)dname, cp, buflen,
263                         dnptrs, lastdnptr)) < 0)
263             if ((n = dn_comp(dname, cp, buflen, dnptrs, lastdnptr)) < 0)
264                 return (-1);
265         cp += n;
266         putshort(type, cp);
267         cp += sizeof (u_short);
268         putshort(class, cp);
269         cp += sizeof (u_short);
270         putlong(0, cp);
271         cp += sizeof (u_long);
272         putshort(datalen, cp);
273         cp += sizeof (u_short);
274         if (datalen) {
275 #ifdef SYSV
276             memcpy((void *)cp, (void *)data, datalen);
277 #else
278             bcopy(data, cp, datalen);
279 #endif
280             cp += datalen;
281         }
282         if ((op == UPDATED) || (op == UPDATEDA)) {
283             hp->ancount = htons(0);
284             break;
285         }
286         /* Else UPDATEM/UPDATEMA, so drop into code for UPDATEA */

287     case UPDATEA: /* Add new resource record */
288         buflen -= RRFIXEDSZ + datalen;
289         if ((n = dn_comp((u_char *)dname, cp, buflen,
290                         dnptrs, lastdnptr)) < 0)
291             if ((n = dn_comp(dname, cp, buflen, dnptrs, lastdnptr)) < 0)
292                 return (-1);
293         cp += n;
294         putshort(newrr->r_type, cp);
295         cp += sizeof (u_short);
296         putshort(newrr->r_class, cp);
297         cp += sizeof (u_short);
298         putlong(0, cp);
299         cp += sizeof (u_long);
300         putshort(newrr->r_size, cp);
301         cp += sizeof (u_short);
302         if (newrr->r_size) {
303 #ifdef SYSV
304             memcpy((void *)cp, newrr->r_data, newrr->r_size);
305 #else
306             bcopy(newrr->r_data, cp, newrr->r_size);
307 #endif
308             cp += newrr->r_size;
309         }
310         hp->ancount = htons(0);
311         break;

```

```

313 #endif /* ALLOW_UPDATES */
314 }
315     return ((char *)cp - buf);
308     return (cp - buf);
316 }

```

*unchanged\_portion\_omitted*

```
*****
8066 Thu Jul 9 15:59:31 2015
new/usr/src/lib/libresolv/res_query.c
1926 libresolv evades compiler warnings
*****
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 2015 Gary Mills
23 * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 */
26 /*
27 * Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 * All Rights Reserved */
29 /*
30 */
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 <sys/param.h>
42 #include <sys/socket.h>
43 #include <netinet/in.h>
44 #include <cctype.h>
45 #include <netdb.h>
46 #include <stdio.h>
47 #include <errno.h>
48 #include <string.h>
49 #include <stdlib.h>
50 #include <arpa/inet.h>
51 #include <arpa/nameser.h>
52 #include <resolv.h>
53 #include "crossl.h"
54
55 #if PACKETSZ > 1024
56 #define MAXPACKET PACKETSZ
57 #else
58 #define MAXPACKET 1024
59#endif
```

```
61 int h_errno;
62 /*
63  * Formulate a normal query, send, and await answer.
64  * Returned answer is placed in supplied buffer "answer".
65  * Perform preliminary check of answer, returning success only
66  * if no error is indicated and the answer count is nonzero.
67  * Return the size of the response on success, -1 on error.
68  * Error number is left in h_errno.
69  * Caller must parse answer and determine whether it answers the question.
70 */
71 /*
72 int
73 res_query(name, class, type, answer, anslen)
74     char *name;           /* domain name */
75     int class, type;     /* class and type of query */
76     u_char *answer;      /* buffer to put answer */
77     int anslen;          /* size of answer buffer */
78 {
79     char buf[MAXPACKET];
80     HEADER *hp;
81     int n;
82
83     if ((-_res.options & RES_INIT) == 0 && res_init() == -1)
84         return (-1);
85 #ifdef DEBUG
86     if (_res.options & RES_DEBUG)
87         printf("res_query(%s, %d, %d)\n", name, class, type);
88 #endif
89     n = res_mkquery(QUERY, name, class, type, (char *)NULL, 0, NULL,
90                     buf, sizeof (buf));
91
92     if (n <= 0) {
93 #ifdef DEBUG
94         if (_res.options & RES_DEBUG)
95             printf("res_query: mkquery failed\n");
96 #endif
97         h_errno = NO_RECOVERY;
98         return (n);
99     }
100    n = res_send(buf, n, (char *)answer, anslen);
101    n = res_send(buf, n, answer, anslen);
102    if (n < 0) {
103 #ifdef DEBUG
104         if (_res.options & RES_DEBUG)
105             printf("res_query: send error\n");
106 #endif
107         h_errno = TRY AGAIN;
108         return (n);
109
110     hp = (HEADER *) answer;
111     if (hp->rcode != NOERROR || ntohs(hp->ancount) == 0) {
112 #ifdef DEBUG
113         if (_res.options & RES_DEBUG)
114             printf("rcode = %d, ancount=%d\n", hp->rcode,
115                   ntohs(hp->ancount));
116 #endif
117         switch (hp->rcode) {
118             case NXDOMAIN:
119                 h_errno = HOST_NOT_FOUND;
120                 break;
121             case SERVFAIL:
122                 h_errno = TRY AGAIN;
123                 break;
124             case NOERROR:
```

```
125             h_errno = NO_DATA;
126             break;
127         case FORMERR:
128         case NOTIMP:
129         case REFUSED:
130         default:
131             h_errno = NO_RECOVERY;
132             break;
133         }
134     return (-1);
135 }
136 if (hp->rcode == NOERROR && ntohs(hp->ancount) > 0)
137     h_errno = 0;
138 return (n);
139 }
```

unchanged portion omitted

```
269 char *
270 hostalias(name)
271 {
272     register char *name;
273     register char *C1, *C2;
274     FILE *fp;
275     char *file;
276     char *file, *getenv(), *strcpy(), *strncpy();
277     char buf[BUFFSIZ];
278     static char abuf[MAXDNAME];
279
280     file = getenv("HOSTALIASES");
281     if (file == NULL || (fp = fopen(file, "r")) == NULL)
282         return (NULL);
283     buf[sizeof (buf) - 1] = '\0';
284     while (fgets(buf, sizeof (buf), fp)) {
285         for (C1 = buf; *C1 && !isspace(*C1); ++C1);
286         if (!*C1)
287             break;
288         *C1 = '\0';
289         if (!strcasecmp(buf, name)) {
290             while (isspace(*++C1));
291             if (!*C1)
292                 break;
293             for (C2 = C1 + 1; *C2 && !isspace(*C2); ++C2);
294             abuf[sizeof (abuf) - 1] = *C2 = '\0';
295             (void) strncpy(abuf, C1, sizeof (abuf) - 1);
296             fclose(fp);
297             return (abuf);
298         }
299     }
300     fclose(fp);
301 }
```

unchanged portion omitted

```
new/usr/src/lib/libresolv/res_send.c
```

```
*****
12291 Thu Jul 9 15:59:31 2015
new/usr/src/lib/libresolv/res_send.c
1926 libresolv evades compiler warnings
*****
```

```
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 2015 Gary Mills
24  * Copyright 2008 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 */

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

41 /*
42  * Send query to name server and wait for reply.
43 */

45 #include <sys/param.h>
46 #include <sys/time.h>
47 #include <sys/socket.h>
48 #include <sys/vio.h>
49 #include <sys/stat.h>
50 #include <netinet/in.h>
51 #include <stdio.h>
52 #include <string.h>
53 #include <unistd.h>
54 #include <errno.h>
55 #include <arpa/nameser.h>
56 #include <arpa/inet.h>
57 #include <resolv.h>
58 #include "crossl.h"
```

```
1
```

```
new/usr/src/lib/libresolv/res_send.c
```

```
60 /*
61  * Undocumented external function in libsocket
62  */
63 extern int
64 _socket(int, int, int);

66 static int s = -1;      /* socket used for communications */
67 #if     BSD >= 43
68 static struct sockaddr no_addr;
69 #endif /* BSD */

72 #ifndef FD_SET
73 #define NFDBITS           32
74 #define FD_SETSIZE         32
75 #define FD_SET(n, p)       ((p)->fds_bits[(n)/NFDBITS] |= (1 << ((n) % NFDBITS)))
76 #define FD_CLR(n, p)       ((p)->fds_bits[(n)/NFDBITS] &= ~(1 << ((n) % NFDBITS)))
77 #define FD_ISSET(n, p)     ((p)->fds_bits[(n)/NFDBITS] & (1 << ((n) % NFDBITS)))
78 #ifdef SYSV
79 #define FD_ZERO(p)         (void) memset((void *)(p), 0, sizeof (*(p)))
80 #define FD_ZERO(p)         memset((void *)(p), 0, sizeof (*(p)))
81 #else
82 #define FD_ZERO(p)         bzero((char *)(p), sizeof (*(p)))
83 #endif

85 /*
86  * 1247019: Kludge to time out quickly if there is no /etc/resolv.conf
87  * and a TCP connection to the local DNS server fails.
88 */

90 static int _confcheck()
91 {
92     int ns;
93     struct stat rc_stat;
94     struct sockaddr_in ns_sin;

97     /* First, we check to see if /etc/resolv.conf exists.
98      * If it doesn't, then localhost is mostlikely to be
99      * the nameserver.
100     */
101    if (stat(_PATH_RESCONF, &rc_stat) == -1 && errno == ENOENT) {

103        /* Next, we check to see if _res.nsaddr is set to loopback.
104         * If it isn't, it has been altered by the application
105         * explicitly and we then want to bail with success.
106         */
107    if (_res.nsaddr.sin_addr.S_un.S_addr == htonl(INADDR_LOOPBACK))

109        /* Lastly, we try to connect to the TCP port of the
110         * nameserver. If this fails, then we know that
111         * DNS is misconfigured and we can quickly exit.
112         */
113    ns = socket(AF_INET, SOCK_STREAM, 0);
114    IN_SET_LOOPBACK_ADDR(&ns_sin);
115    ns_sin.sin_port = htons(NAMESERVER_PORT);
116    if (connect(ns, (struct sockaddr *)&ns_sin,
117                sizeof ns_sin) == -1) {
118        close(ns);
119        return(-1);
120    }
121    else {
122        close(ns);
123        return(0);
124    }
```

```
2
```

```

125             }
126         return(0);
127     }
128     return (0);
129 }
130
131 }

133 int
134 res_send(buf, buflen, answer, anslen)
135     char *buf;
136     int buflen;
137     char *answer;
138     int anslen;
139 {
140     register int n;
141     int try, v_circuit, respplen, ns;
142     int gotsomewhere = 0;
143 #if BSD >= 43
144     int connected = 0;
145 #endif /* BSD */
146     int connreset = 0;
147     u_short id, len;
148     char *cp;
149     fd_set dsmask;
150     struct timeval timeout;
151     HEADER *hp = (HEADER *) buf;
152     HEADER *anhp = (HEADER *) answer;
153     struct iovec iov[2];
154     int terrno = ETIMEDOUT;
155     char junk[512];

157 #ifdef DEBUG
158     if (_res.options & RES_DEBUG) {
159         printf("res_send()\n");
160         p_query(buf);
161     }
162 #endif
163     if (!(_res.options & RES_INIT))
164         if (res_init() == -1) {
165             return (-1);
166         }

168     /* 1247019: Check to see if we can bailout quickly. */
169     if (_confcheck() == -1)
170         return(-1);

172     v_circuit = (_res.options & RES_USEVC) || buflen > PACKETSZ;
173     id = hp->id;
174     /*
175      * Send request, RETRY times, or until successful
176      */
177     for (try = 0; try < _res.retry; try++) {
178         for (ns = 0; ns < _res.nscount; ns++) {
179 #ifdef DEBUG
180             if (_res.options & RES_DEBUG)
181                 printf("Querying server (# %d) address = %s\n",
182                         ns+1, inet_ntoa(_res.nsaddr_list[ns].sin_addr));
183 #endif
184             usevc:
185             if (v_circuit) {
186                 int truncated = 0;
187
188                 /*
189                  * Use virtual circuit;

```

```

190             * at most one attempt per server.
191             */
192             try = _res.retry;
193             if (s < 0) {
194                 s = _socket(AF_INET, SOCK_STREAM, 0);
195                 if (s < 0) {
196                     terrno = errno;
197 #ifdef DEBUG
198                     perror("socket (vc) failed");
199                 }
200             }
201         #endif
202     }
203
204     if (connect(s, (struct sockaddr *)&_res,
205                 sizeof (struct sockaddr)) < 0) {
206         terrno = errno;
207 #ifdef DEBUG
208         perror("connect failed");
209     }
210
211     (void) close(s);
212     s = -1;
213     continue;
214 }
215
216 /* Send length & message
217 */
218 len = htons((u_short)buflen);
219 iov[0].iov_base = (caddr_t)&len;
220 iov[0].iov_len = sizeof (len);
221 iov[1].iov_base = buf;
222 iov[1].iov_len = buflen;
223 if (writev(s, iov, 2) != sizeof (len) +
224     buflen) {
225     terrno = errno;
226
227 #ifdef DEBUG
228         perror("write failed");
229     }
230
231     (void) close(s);
232     s = -1;
233     continue;
234 }
235
236 /* Receive length & response
237 */
238 cp = answer;
239 len = sizeof (short);
240 while (len != 0 && (n = read
241     (s, (char *)cp, (int)len)) > 0) {
242     cp += n;
243     len -= n;
244 }
245
246 #ifdef DEBUG
247     perror("read failed");
248 #endif
249
250 (void) close(s);
251 s = -1;
252
253 /* A long running process might get its TCP

```

```

256
257     * connection reset if the remote server was
258     * restarted. Requery the server instead of
259     * trying a new one. When there is only one
260     * server, this means that a query might work
261     * instead of failing. We only allow one reset
262     * per query to prevent looping.
263     */
264     if (terrno == ECONNRESET &&
265         !connreset) {
266         connreset = 1;
267         ns--;
268     }
269     continue;
270 }
271 cp = answer;
272 if ((resplen = ntohs(*(u_short *)cp)) >
273      anslen) {
274     if (_res.options & RES_DEBUG)
275         fprintf(stderr,
276                 "response truncated\n");
277 #endif
278     len = anslen;
279     truncated = 1;
280 } else
281     len = resplen;
282 while (len != 0 &&
283         (n = read(s, (char *)cp,
284                    (int)len)) > 0) {
285     cp += n;
286     len -= n;
287 }
288 if (n <= 0) {
289     terrno = errno;
290     if (_res.options & RES_DEBUG)
291         perror("read failed");
292     (void) close(s);
293     s = -1;
294     continue;
295 }
296 if (truncated) {
297     /*
298      * Flush rest of answer
299      * so connection stays in synch.
300      */
301     anhp->tc = 1;
302     len = resplen - anslen;
303     /*
304      * set the value of resplen to anslen,
305      * this is done because the caller
306      * assumes resplen contains the size of
307      * message read into the "answer" buffer
308      * passed in.
309      */
310     resplen = anslen;
311
312     while (len != 0) {
313         n = (len > sizeof (junk) ?
314              sizeof (junk) : len);
315         if ((n = read(s, junk, n)) > 0)
316             len -= n;
317         else
318             break;
319     }
320
321 #ifdef DEBUG
322     if (_res.options & RES_DEBUG)
323         fprintf(stderr,
324                 "response truncated\n");
325     len = anslen;
326     if (len < 0) {
327         connreset = 1;
328         ns--;
329     }
330     continue;
331 #endif
332     if (len < 0) {
333         connreset = 1;
334         ns--;
335     }
336     if (len < 0) {
337         connreset = 1;
338         ns--;
339     }
340     if (len < 0) {
341         connreset = 1;
342         ns--;
343     }
344     if (len < 0) {
345         connreset = 1;
346         ns--;
347     }
348     if (len < 0) {
349         connreset = 1;
350         ns--;
351     }
352     if (len < 0) {
353         connreset = 1;
354         ns--;
355     }
356     if (len < 0) {
357         connreset = 1;
358         ns--;
359     }
360     if (len < 0) {
361         connreset = 1;
362         ns--;
363     }
364     if (len < 0) {
365         connreset = 1;
366         ns--;
367     }
368     if (len < 0) {
369         connreset = 1;
370         ns--;
371     }
372     if (len < 0) {
373         connreset = 1;
374         ns--;
375     }
376     if (len < 0) {
377         connreset = 1;
378     }
379     if (len < 0) {
380         connreset = 1;
381     }
382     if (len < 0) {
383         connreset = 1;
384         ns--;
385     }
386     if (len < 0) {
387         connreset = 1;
388         ns--;
389     }
390     if (len < 0) {
391         connreset = 1;
392         ns--;
393     }
394     if (len < 0) {
395         connreset = 1;
396         ns--;
397     }
398     if (len < 0) {
399         connreset = 1;
400         ns--;
401     }
402     if (len < 0) {
403         connreset = 1;
404         ns--;
405     }
406     if (len < 0) {
407         connreset = 1;
408         ns--;
409     }
410     if (len < 0) {
411         connreset = 1;
412         ns--;
413     }
414     if (len < 0) {
415         connreset = 1;
416         ns--;
417     }
418     if (len < 0) {
419         connreset = 1;
420         ns--;
421     }
422     if (len < 0) {
423         connreset = 1;
424         ns--;
425     }
426     if (len < 0) {
427         connreset = 1;
428         ns--;
429     }
430     if (len < 0) {
431         connreset = 1;
432         ns--;
433     }
434     if (len < 0) {
435         connreset = 1;
436         ns--;
437     }
438     if (len < 0) {
439         connreset = 1;
440         ns--;
441     }
442     if (len < 0) {
443         connreset = 1;
444         ns--;
445     }
446     if (len < 0) {
447         connreset = 1;
448         ns--;
449     }
450     if (len < 0) {
451         connreset = 1;
452         ns--;
453     }
454     if (len < 0) {
455         connreset = 1;
456         ns--;
457     }
458     if (len < 0) {
459         connreset = 1;
460         ns--;
461     }
462     if (len < 0) {
463         connreset = 1;
464         ns--;
465     }
466     if (len < 0) {
467         connreset = 1;
468         ns--;
469     }
470     if (len < 0) {
471         connreset = 1;
472         ns--;
473     }
474     if (len < 0) {
475         connreset = 1;
476         ns--;
477     }
478     if (len < 0) {
479         connreset = 1;
480         ns--;
481     }
482     if (len < 0) {
483         connreset = 1;
484         ns--;
485     }
486     if (len < 0) {
487         connreset = 1;
488         ns--;
489     }
490     if (len < 0) {
491         connreset = 1;
492         ns--;
493     }
494     if (len < 0) {
495         connreset = 1;
496         ns--;
497     }
498     if (len < 0) {
499         connreset = 1;
500         ns--;
501     }
502     if (len < 0) {
503         connreset = 1;
504         ns--;
505     }
506     if (len < 0) {
507         connreset = 1;
508         ns--;
509     }
510     if (len < 0) {
511         connreset = 1;
512         ns--;
513     }
514     if (len < 0) {
515         connreset = 1;
516         ns--;
517     }
518     if (len < 0) {
519         connreset = 1;
520         ns--;
521     }
522     if (len < 0) {
523         connreset = 1;
524         ns--;
525     }
526     if (len < 0) {
527         connreset = 1;
528         ns--;
529     }
530     if (len < 0) {
531         connreset = 1;
532         ns--;
533     }
534     if (len < 0) {
535         connreset = 1;
536         ns--;
537     }
538     if (len < 0) {
539         connreset = 1;
540         ns--;
541     }
542     if (len < 0) {
543         connreset = 1;
544         ns--;
545     }
546     if (len < 0) {
547         connreset = 1;
548         ns--;
549     }
550     if (len < 0) {
551         connreset = 1;
552         ns--;
553     }
554     if (len < 0) {
555         connreset = 1;
556         ns--;
557     }
558     if (len < 0) {
559         connreset = 1;
560         ns--;
561     }
562     if (len < 0) {
563         connreset = 1;
564         ns--;
565     }
566     if (len < 0) {
567         connreset = 1;
568         ns--;
569     }
570     if (len < 0) {
571         connreset = 1;
572         ns--;
573     }
574     if (len < 0) {
575         connreset = 1;
576         ns--;
577     }
578     if (len < 0) {
579         connreset = 1;
580         ns--;
581     }
582     if (len < 0) {
583         connreset = 1;
584         ns--;
585     }
586     if (len < 0) {
587         connreset = 1;
588         ns--;
589     }
58
59     if (len < 0) {
590         connreset = 1;
591         ns--;
592     }
593     if (len < 0) {
594         connreset = 1;
595         ns--;
596     }
597     if (len < 0) {
598         connreset = 1;
599         ns--;
600     }
601     if (len < 0) {
602         connreset = 1;
603         ns--;
604     }
605     if (len < 0) {
606         connreset = 1;
607         ns--;
608     }
609     if (len < 0) {
610         connreset = 1;
611         ns--;
612     }
613     if (len < 0) {
614         connreset = 1;
615         ns--;
616     }
617     if (len < 0) {
618         connreset = 1;
619         ns--;
620     }
621     if (len < 0) {
622         connreset = 1;
623         ns--;
624     }
625     if (len < 0) {
626         connreset = 1;
627         ns--;
628     }
629     if (len < 0) {
630         connreset = 1;
631         ns--;
632     }
633     if (len < 0) {
634         connreset = 1;
635         ns--;
636     }
637     if (len < 0) {
638         connreset = 1;
639         ns--;
640     }
641     if (len < 0) {
642         connreset = 1;
643         ns--;
644     }
645     if (len < 0) {
646         connreset = 1;
647         ns--;
648     }
649     if (len < 0) {
650         connreset = 1;
651         ns--;
652     }
653     if (len < 0) {
654         connreset = 1;
655         ns--;
656     }
657     if (len < 0) {
658         connreset = 1;
659         ns--;
660     }
661     if (len < 0) {
662         connreset = 1;
663         ns--;
664     }
665     if (len < 0) {
666         connreset = 1;
667         ns--;
668     }
669     if (len < 0) {
670         connreset = 1;
671         ns--;
672     }
673     if (len < 0) {
674         connreset = 1;
675         ns--;
676     }
677     if (len < 0) {
678         connreset = 1;
679         ns--;
680     }
681     if (len < 0) {
682         connreset = 1;
683         ns--;
684     }
685     if (len < 0) {
686         connreset = 1;
687         ns--;
688     }
689     if (len < 0) {
690         connreset = 1;
691         ns--;
692     }
693     if (len < 0) {
694         connreset = 1;
695         ns--;
696     }
697     if (len < 0) {
698         connreset = 1;
699         ns--;
700     }
701     if (len < 0) {
702         connreset = 1;
703         ns--;
704     }
705     if (len < 0) {
706         connreset = 1;
707         ns--;
708     }
709     if (len < 0) {
710         connreset = 1;
711         ns--;
712     }
713     if (len < 0) {
714         connreset = 1;
715         ns--;
716     }
717     if (len < 0) {
718         connreset = 1;
719         ns--;
720     }
721     if (len < 0) {
722         connreset = 1;
723         ns--;
724     }
725     if (len < 0) {
726         connreset = 1;
727         ns--;
728     }
729     if (len < 0) {
730         connreset = 1;
731         ns--;
732     }
733     if (len < 0) {
734         connreset = 1;
735         ns--;
736     }
737     if (len < 0) {
738         connreset = 1;
739         ns--;
740     }
741     if (len < 0) {
742         connreset = 1;
743         ns--;
744     }
745     if (len < 0) {
746         connreset = 1;
747         ns--;
748     }
749     if (len < 0) {
750         connreset = 1;
751         ns--;
752     }
753     if (len < 0) {
754         connreset = 1;
755         ns--;
756     }
757     if (len < 0) {
758         connreset = 1;
759         ns--;
760     }
761     if (len < 0) {
762         connreset = 1;
763         ns--;
764     }
765     if (len < 0) {
766         connreset = 1;
767         ns--;
768     }
769     if (len < 0) {
770         connreset = 1;
771         ns--;
772     }
773     if (len < 0) {
774         connreset = 1;
775         ns--;
776     }
777     if (len < 0) {
778         connreset = 1;
779         ns--;
780     }
781     if (len < 0) {
782         connreset = 1;
783         ns--;
784     }
785     if (len < 0) {
786         connreset = 1;
787         ns--;
788     }
789     if (len < 0) {
790         connreset = 1;
791         ns--;
792     }
793     if (len < 0) {
794         connreset = 1;
795         ns--;
796     }
797     if (len < 0) {
798         connreset = 1;
799         ns--;
800     }
801     if (len < 0) {
802         connreset = 1;
803         ns--;
804     }
805     if (len < 0) {
806         connreset = 1;
807         ns--;
808     }
809     if (len < 0) {
810         connreset = 1;
811         ns--;
812     }
813     if (len < 0) {
814         connreset = 1;
815         ns--;
816     }
817     if (len < 0) {
818         connreset = 1;
819         ns--;
820     }
821     if (len < 0) {
822         connreset = 1;
823         ns--;
824     }
825     if (len < 0) {
826         connreset = 1;
827         ns--;
828     }
829     if (len < 0) {
830         connreset = 1;
831         ns--;
832     }
833     if (len < 0) {
834         connreset = 1;
835         ns--;
836     }
837     if (len < 0) {
838         connreset = 1;
839         ns--;
840     }
841     if (len < 0) {
842         connreset = 1;
843         ns--;
844     }
845     if (len < 0) {
846         connreset = 1;
847         ns--;
848     }
849     if (len < 0) {
850         connreset = 1;
851         ns--;
852     }
853     if (len < 0) {
854         connreset = 1;
855         ns--;
856     }
857     if (len < 0) {
858         connreset = 1;
859         ns--;
860     }
861     if (len < 0) {
862         connreset = 1;
863         ns--;
864     }
865     if (len < 0) {
866         connreset = 1;
867         ns--;
868     }
869     if (len < 0) {
870         connreset = 1;
871         ns--;
872     }
873     if (len < 0) {
874         connreset = 1;
875         ns--;
876     }
877     if (len < 0) {
878         connreset = 1;
879         ns--;
880     }
881     if (len < 0) {
882         connreset = 1;
883         ns--;
884     }
885     if (len < 0) {
886         connreset = 1;
887         ns--;
888     }
889     if (len < 0) {
890         connreset = 1;
891         ns--;
892     }
893     if (len < 0) {
894         connreset = 1;
895         ns--;
896     }
897     if (len < 0) {
898         connreset = 1;
899         ns--;
900     }
901     if (len < 0) {
902         connreset = 1;
903         ns--;
904     }
905     if (len < 0) {
906         connreset = 1;
907         ns--;
908     }
909     if (len < 0) {
910         connreset = 1;
911         ns--;
912     }
913     if (len < 0) {
914         connreset = 1;
915         ns--;
916     }
917     if (len < 0) {
918         connreset = 1;
919         ns--;
920     }
921     if (len < 0) {
922         connreset = 1;
923         ns--;
924     }
925     if (len < 0) {
926         connreset = 1;
927         ns--;
928     }
929     if (len < 0) {
930         connreset = 1;
931         ns--;
932     }
933     if (len < 0) {
934         connreset = 1;
935         ns--;
936     }
937     if (len < 0) {
938         connreset = 1;
939         ns--;
940     }
941     if (len < 0) {
942         connreset = 1;
943         ns--;
944     }
945     if (len < 0) {
946         connreset = 1;
947         ns--;
948     }
949     if (len < 0) {
950         connreset = 1;
951         ns--;
952     }
953     if (len < 0) {
954         connreset = 1;
955         ns--;
956     }
957     if (len < 0) {
958         connreset = 1;
959         ns--;
960     }
961     if (len < 0) {
962         connreset = 1;
963         ns--;
964     }
965     if (len < 0) {
966         connreset = 1;
967         ns--;
968     }
969     if (len < 0) {
970         connreset = 1;
971         ns--;
972     }
973     if (len < 0) {
974         connreset = 1;
975         ns--;
976     }
977     if (len < 0) {
978         connreset = 1;
979         ns--;
980     }
981     if (len < 0) {
982         connreset = 1;
983         ns--;
984     }
985     if (len < 0) {
986         connreset = 1;
987         ns--;
988     }
989     if (len < 0) {
990         connreset = 1;
991         ns--;
992     }
993     if (len < 0) {
994         connreset = 1;
995         ns--;
996     }
997     if (len < 0) {
998         connreset = 1;
999         ns--;
1000    }
1001
1002    if (len < 0) {
1003        connreset = 1;
1004        ns--;
1005    }
1006    if (len < 0) {
1007        connreset = 1;
1008        ns--;
1009    }
1010    if (len < 0) {
1011        connreset = 1;
1012        ns--;
1013    }
1014    if (len < 0) {
1015        connreset = 1;
1016        ns--;
1017    }
1018    if (len < 0) {
1019        connreset = 1;
1020        ns--;
1021    }
1022    if (len < 0) {
1023        connreset = 1;
1024        ns--;
1025    }
1026    if (len < 0) {
1027        connreset = 1;
1028        ns--;
1029    }
1030    if (len < 0) {
1031        connreset = 1;
1032        ns--;
1033    }
1034    if (len < 0) {
1035        connreset = 1;
1036        ns--;
1037    }
1038    if (len < 0) {
1039        connreset = 1;
1040        ns--;
1041    }
1042    if (len < 0) {
1043        connreset = 1;
1044        ns--;
1045    }
1046    if (len < 0) {
1047        connreset = 1;
1048        ns--;
1049    }
1050    if (len < 0) {
1051        connreset = 1;
1052        ns--;
1053    }
1054    if (len < 0) {
1055        connreset = 1;
1056        ns--;
1057    }
1058    if (len < 0) {
1059        connreset = 1;
1060        ns--;
1061    }
1062    if (len < 0) {
1063        connreset = 1;
1064        ns--;
1065    }
1066    if (len < 0) {
1067        connreset = 1;
1068        ns--;
1069    }
1070    if (len < 0) {
1071        connreset = 1;
1072        ns--;
1073    }
1074    if (len < 0) {
1075        connreset = 1;
1076        ns--;
1077    }
1078    if (len < 0) {
1079        connreset = 1;
1080        ns--;
1081    }
1082    if (len < 0) {
1083        connreset = 1;
1084        ns--;
1085    }
1086    if (len < 0) {
1087        connreset = 1;
1088        ns--;
1089    }
1090    if (len < 0) {
1091        connreset = 1;
1092        ns--;
1093    }
1094    if (len < 0) {
1095        connreset = 1;
1096        ns--;
1097    }
1098    if (len < 0) {
1099        connreset = 1;
1100        ns--;
1101    }
1102    if (len < 0) {
1103        connreset = 1;
1104        ns--;
1105    }
1106    if (len < 0) {
1107        connreset = 1;
1108        ns--;
1109    }
1110    if (len < 0) {
1111        connreset = 1;
1112        ns--;
1113    }
1114    if (len < 0) {
1115        connreset = 1;
1116        ns--;
1117    }
1118    if (len < 0) {
1119        connreset = 1;
1120        ns--;
1121    }
1122    if (len < 0) {
1123        connreset = 1;
1124        ns--;
1125    }
1126    if (len < 0) {
1127        connreset = 1;
1128        ns--;
1129    }
1130    if (len < 0) {
1131        connreset = 1;
1132        ns--;
1133    }
1134    if (len < 0) {
1135        connreset = 1;
1136        ns--;
1137    }
1138    if (len < 0) {
1139        connreset = 1;
1140        ns--;
1141    }
1142    if (len < 0) {
1143        connreset = 1;
1144        ns--;
1145    }
1146    if (len < 0) {
1147        connreset = 1;
1148        ns--;
1149    }
1150    if (len < 0) {
1151        connreset = 1;
1152        ns--;
1153    }
1154    if (len < 0) {
1155        connreset = 1;
1156        ns--;
1157    }
1158    if (len < 0) {
1159        connreset = 1;
1160        ns--;
1161    }
1162    if (len < 0) {
1163        connreset = 1;
1164        ns--;
1165    }
1166    if (len < 0) {
1167        connreset = 1;
1168        ns--;
1169    }
1170    if (len < 0) {
1171        connreset = 1;
1172        ns--;
1173    }
1174    if (len < 0) {
1175        connreset = 1;
1176        ns--;
1177    }
1178    if (len < 0) {
1179        connreset = 1;
1180        ns--;
1181    }
1182    if (len < 0) {
1183        connreset = 1;
1184        ns--;
1185    }
1186    if (len < 0) {
1187        connreset = 1;
1188        ns--;
1189    }
1190    if (len < 0) {
1191        connreset = 1;
1192        ns--;
1193    }
1194    if (len < 0) {
1195        connreset = 1;
1196        ns--;
1197    }
1198    if (len < 0) {
1199        connreset = 1;
1200        ns--;
1201    }
1202    if (len < 0) {
1203        connreset = 1;
1204        ns--;
1205    }
1206    if (len < 0) {
1207        connreset = 1;
1208        ns--;
1209    }
1210    if (len < 0) {
1211        connreset = 1;
1212        ns--;
1213    }
1214    if (len < 0) {
1215        connreset = 1;
1216        ns--;
1217    }
1218    if (len < 0) {
1219        connreset = 1;
1220        ns--;
1221    }
1222    if (len < 0) {
1223        connreset = 1;
1224        ns--;
1225    }
1226    if (len < 0) {
1227        connreset = 1;
1228        ns--;
1229    }
1230    if (len < 0) {
1231        connreset = 1;
1232        ns--;
1233    }
1234    if (len < 0) {
1235        connreset = 1;
1236        ns--;
1237    }
1238    if (len < 0) {
1239        connreset = 1;
1240        ns--;
1241    }
1242    if (len < 0) {
1243        connreset = 1;
1244        ns--;
1245    }
1246    if (len < 0) {
1247        connreset = 1;
1248        ns--;
1249    }
1250    if (len < 0) {
1251        connreset = 1;
1252        ns--;
1253    }
1254    if (len < 0) {
1255        connreset = 1;
1256        ns--;
1257    }
1258    if (len < 0) {
1259        connreset = 1;
1260        ns--;
1261    }
1262    if (len < 0) {
1263        connreset = 1;
1264        ns--;
1265    }
1266    if (len < 0) {
1267        connreset = 1;
1268        ns--;
1269    }
1270    if (len < 0) {
1271        connreset = 1;
1272        ns--;
1273    }
1274    if (len < 0) {
1275        connreset = 1;
1276        ns--;
1277    }
1278    if (len < 0) {
1279        connreset = 1;
1280        ns--;
1281    }
1282    if (len < 0) {
1283        connreset = 1;
1284        ns--;
1285    }
1286    if (len < 0) {
1287        connreset = 1;
1288        ns--;
1289    }
1290    if (len < 0) {
1291        connreset = 1;
1292        ns--;
1293    }
1294    if (len < 0) {
1295        connreset = 1;
1296        ns--;
1297    }
1298    if (len < 0) {
1299        connreset = 1;
1300        ns--;
1301    }
1302    if (len < 0) {
1303        connreset = 1;
1304        ns--;
1305    }
1306    if (len < 0) {
1307        connreset = 1;
1308        ns--;
1309    }
1310    if (len < 0) {
1311        connreset = 1;
1312        ns--;
1313    }
1314    if (len < 0) {
1315        connreset = 1;
1316        ns--;
1317    }
1318    if (len < 0) {
1319        connreset = 1;
1320        ns--;
1321    }
1322    if (len < 0) {
1323        connreset = 1;
1324        ns--;
1325    }
1326    if (len < 0) {
1327        connreset = 1;
1328        ns--;
1329    }
1330    if (len < 0) {
1331        connreset = 1;
1332        ns--;
1333    }
1334    if (len < 0) {
1335        connreset = 1;
1336        ns--;
1337    }
1338    if (len < 0) {
1339        connreset = 1;
1340        ns--;
1341    }
1342    if (len < 0) {
1343        connreset = 1;
1344        ns--;
1345    }
1346    if (len < 0) {
1347        connreset = 1;
1348        ns--;
1349    }
1350    if (len < 0) {
1351        connreset = 1;
1352        ns--;
1353    }
1354    if (len < 0) {
1355        connreset = 1;
1356        ns--;
1357    }
1358    if (len < 0) {
1359        connreset = 1;
1360        ns--;
1361    }
1362    if (len < 0) {
1363        connreset = 1;
1364        ns--;
1365    }
1366    if (len < 0) {
1367        connreset = 1;
1368        ns--;
1369    }
1370    if (len < 0) {
1371        connreset = 1;
1372        ns--;
1373    }
1374    if (len < 0) {
1375        connreset = 1;
1376        ns--;
1377    }
1378    if (len < 0) {
1379        connreset = 1;
1380        ns--;
1381    }
1382    if (len < 0) {
1383        connreset = 1;
1384        ns--;
1385    }
1386    if (len < 0) {
1387        connreset = 1;
1388        ns--;
1389    }
1390    if (len < 0) {
1391        connreset = 1;
1392        ns--;
1393    }
1394    if (len < 0) {
1395        connreset = 1;
1396        ns--;
1397    }
1398    if (len < 0) {
1399        connreset = 1;
1400        ns--;
1401    }
1402    if (len < 0) {
1403        connreset = 1;
1404        ns--;
1405    }
1406    if (len < 0) {
1407        connreset = 1;
1408        ns--;
1409    }
1410    if (len < 0) {
1411        connreset = 1;
1412        ns--;
1413    }
1414    if (len < 0) {
1415        connreset = 1;
1416        ns--;
1417    }
1418    if (len < 0) {
1419        connreset = 1;
1420        ns--;
1421    }
1422    if (len < 0) {
1423        connreset = 1;
1424        ns--;
1425    }
1426    if (len < 0) {
1427        connreset = 1;
1428        ns--;
1429    }
1430    if (len < 0) {
1431        connreset = 1;
1432        ns--;
1433    }
1434    if (len < 0) {
1435        connreset = 1;
1436        ns--;
1437    }
1438    if (len < 0) {
1439        connreset = 1;
1440        ns--;
1441    }
1442    if (len < 0) {
1443        connreset = 1;
1444        ns--;
1445    }
1446    if (len < 0) {
1447        connreset = 1;
1448        ns--;
1449    }
1450    if (len < 0) {
1451        connreset = 1;
1452        ns--;
1453    }
1454    if (len < 0) {
1455        connreset = 1;
1456        ns--;
1457    }
1458    if (len < 0) {
1459        connreset = 1;
1460        ns--;
1461    }
1462    if (len < 0) {
1463        connreset = 1;
1464        ns--;
1465    }
1466    if (len < 0) {
1467        connreset = 1;
1468        ns--;
1469    }
1470    if (len < 0) {
1471        connreset = 1;
1472        ns--;
1473    }
1474    if (len < 0) {
1475        connreset = 1;
1476        ns--;
1477    }
1478    if (len < 0) {
1479        connreset = 1;
1480        ns--;
1481    }
1482    if (
```

```

388
389
390
391
392
393
394 #endif /* BSD */
395
396
397
398 #ifdef DEBUG
399
400
401 #endif
402
403 #if      BSD >= 43
404
405
406 #endif
407
408
409
410
411
412
413
414
415
416
417 wait:
418
419
420
421
422
423 #ifdef DEBUG
424
425
426 #endif
427
428
429
430
431
432
433 #ifdef DEBUG
434
435
436 #endif
437 #if BSD >= 43
438
439 #endif
440
441
442
443
444 #ifdef DEBUG
445
446
447 #endif
448
449
450
451
452
453
    */
    if (connected) {
        (void) connect(s, &no_addr,
                      sizeof (no_addr));
        connected = 0;
    }
    if (sendto(s, buf, buflen, 0,
               (struct sockaddr *) &_res.nsaddr
               sizeof (struct sockaddr)) != buflen) {
        if (_res.options & RES_DEBUG)
            perror("sendto");
        continue;
    }
}
/*
 * Wait for reply
 */
timeout.tv_sec = (_res.retrans << try);
if (try > 0)
    timeout.tv_sec /= _res.nscount;
if (timeout.tv_sec <= 0)
    timeout.tv_sec = 1;
timeout.tv_usec = 0;
FD_ZERO(&dsmask);
FD_SET(s, &dsmask);
n = select(s+1, &dsmask, (fd_set *)NULL,
           (fd_set *)NULL, &timeout);
if (n < 0) {
    if (_res.options & RES_DEBUG)
        perror("select");
    continue;
}
if (n == 0) {
    /*
     * timeout
     */
    if (_res.options & RES_DEBUG)
        printf("timeout\n");
}
got somewhere = 1;
continue;
if ((resplen = recv(s, answer, anslen, 0))
    <= 0) {
    if (_res.options & RES_DEBUG)
        perror("recvfrom");
    continue;
}
got somewhere = 1;
if (id != anhp->id) {
    /*
     * response from old query, ignore it

```

```

454
455 #ifdef DEBUG
456
457
458
459
460 #endif
461
462
463
464
465
466
467
468 #ifdef DEBUG
469
470
471 #endif
472
473
474
475
476
477
478 #ifdef DEBUG
479
480
481
482
483 #endif
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513 }

*/
if (_res.options & RES_DEBUG) {
    printf("old answer:\n");
    p_query(answer);
}
goto wait;
}
if (!(_res.options & RES_IGNTC) && anhp->tc) {
    /*
     * get rest of answer;
     * use TCP with same server.
     */
    if (_res.options & RES_DEBUG)
        printf("truncated answer\n");
    (void) close(s);
    s = -1;
    v_circuit = 1;
    goto usevc;
}
if (_res.options & RES_DEBUG) {
    printf("got answer:\n");
    p_query(answer);
}
/*
 * If using virtual circuits, we assume that the first server
 * is preferred * over the rest (i.e. it is on the local
 * machine) and only keep that one open.
 * If we have temporarily opened a virtual circuit,
 * or if we haven't been asked to keep a socket open,
 * close the socket.
 */
if ((v_circuit &&
     ((_res.options & RES_USEVC) == 0 || ns != 0) ||
     (_res.options & RES_STAYOPEN) == 0) {
    (void) close(s);
    s = -1;
}
return (resplen);
}
if (s >= 0) {
    (void) close(s);
    s = -1;
}
if (v_circuit == 0)
    if (got somewhere == 0)
        errno = ECONNREFUSED; /* no nameservers found */
    else
        errno = ETIMEDOUT; /* no answer obtained */
else
    errno = terrno;
return (-1);
}



---


unchanged portion omitted

```

```
*****
1727 Thu Jul 9 15:59:32 2015
new/usr/src/lib/libresolv/res_sethost.c
1926 libresolv evades compiler warnings
*****
```

```
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 2015 Gary Mills
24  * Copyright 2008 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 */
```

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

```
41 #include <sys/types.h>
42 #include <arpa/nameser.h>
43 #include <netinet/in.h>
44 #include <resolv.h>
45 #include "crossl.h"
```

```
47 void
48 res_sethostent(stayopen)
49 int stayopen;
50 {
51     if (stayopen)
52         _res.options |= RES_STAYOPEN | RES_USEVC;
53 }
```

---

```
unchanged portion omitted
```