NAME

res_query, **res_search**, **res_mkquery**, **res_send**, **res_init**, **dn_comp**, **dn_expand**, **dn_skipname**, **ns_get16**, **ns_get32**, **ns_put16**, **ns_put32** - resolver routines

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>

int
res_query(const char *dname, int class, int type, u_char *answer, int anslen);

int
res_search(const char *dname, int class, int type, u_char *answer, int anslen);

int
res_mkquery(int op, const char *dname, int class, int type, const u_char *data, int datalen,
    const u_char *newrr_in, u_char *buf, int buflen);

int
res_send(const u_char *msg, int msglen, u_char *answer, int anslen);

int
res_init(void);

int
dn_comp(const char *exp_dn, u_char *comp_dn, int length, u_char **dnptrs, u_char **lastdnptr);

int
dn_expand(const u_char *msg, const u_char *eomorig, const u_char *comp_dn, char *exp_dn,
    int length);

int
dn_skipname(const u_char *comp_dn, const u_char *eom);

u_int
```

ns_get16(*const u_char *src*);

u_long
ns_get32(*const u_char *src*);

void
ns_put16(*u_int src, u_char *dst*);

void
ns_put32(*u_long src, u_char *dst*);

DESCRIPTION

These routines are used for making, sending and interpreting query and reply messages with Internet domain name servers.

Global configuration and state information that is used by the resolver routines is kept in the structure *_res*. Most of the values have reasonable defaults and can be ignored. Options stored in *_res.options* are defined in <*resolv.h*> and are as follows. Options are stored as a simple bit mask containing the bitwise ‘‘or’’ of the options enabled.

RES_INIT — True if the initial name server address and default domain name are initialized (i.e., **res_init**() has been called).

RES_DEBUG — Print debugging messages.

RES_AAONLY — Accept authoritative answers only. With this option, **res_send**() should continue until it finds an authoritative answer or finds an error. Currently this is not implemented.

RES_USEVC — Use TCP connections for queries instead of UDP datagrams.

RES_STAYOPEN — Used with RES_USEVC to keep the TCP connection open between queries. This is useful only in programs that regularly do many queries. UDP should be the normal mode used.

RES_IGNTC — Unused currently (ignore truncation errors, i.e., do not retry with TCP).

RES_RECURSE — Set the recursion-desired bit in queries. This is the default. (**res_send**() does not do iterative queries and expects the name server to handle recursion.)

RES_DEFNAMES
: If set, **res_search**() will append the default domain name to single-component names (those that do not contain a dot). This option is enabled by default.

RES_DNSRCH
: If this option is set, **res_search**() will search for host names in the current domain and in parent domains; see hostname(7). This is used by the standard host lookup routine gethostbyname(3). This option is enabled by default.

RES_NOALIASES
: This option turns off the user level aliasing feature controlled by the "HOSTALIASES" environment variable. Network daemons should set this option.

RES_USE_INET6
: Enables support for IPv6-only applications. This causes IPv4 addresses to be returned as an IPv4 mapped address. For example, 10.1.1.1 will be returned as ::ffff:10.1.1.1. The option is meaningful with certain kernel configuration only.

RES_USE_EDNS0
: Enables support for OPT pseudo-RR for EDNS0 extension. With the option, resolver code will attach OPT pseudo-RR into DNS queries, to inform of our receive buffer size. The option will allow DNS servers to take advantage of non-default receive buffer size, and to send larger replies. DNS query packets with EDNS0 extension is not compatible with non-EDNS0 DNS servers.

The **res_init**() routine reads the configuration file (if any; see resolver(5)) to get the default domain name, search list and the Internet address of the local name server(s). If no server is configured, the host running the resolver is tried. The current domain name is defined by the hostname if not specified in the configuration file; it can be overridden by the environment variable LOCALDOMAIN. This environment variable may contain several blank-separated tokens if you wish to override the *search list* on a per-process basis. This is similar to the **search** command in the configuration file. Another environment variable "RES_OPTIONS" can be set to override certain internal resolver options which are otherwise set by changing fields in the *_res* structure or are inherited from the configuration file's **options** command. The syntax of the "RES_OPTIONS" environment variable is explained in resolver(5). Initialization normally occurs on the first call to one of the following routines.

The **res_query**() function provides an interface to the server query mechanism. It constructs a query, sends it to the local server, awaits a response, and makes preliminary checks on the reply. The query requests information of the specified *type* and *class* for the specified fully-qualified domain name *dname*. The reply message is left in the *answer* buffer with length *anslen* supplied by the caller.

The **res_search**() routine makes a query and awaits a response like **res_query**(), but in addition, it

implements the default and search rules controlled by the RES_DEFNAMES and RES_DNSRCH options. It returns the first successful reply.

The remaining routines are lower-level routines used by **res_query**(). The **res_mkquery**() function constructs a standard query message and places it in *buf*. It returns the size of the query, or -1 if the query is larger than *buflen*. The query type *op* is usually QUERY, but can be any of the query types defined in <*arpa/nameser.h*>. The domain name for the query is given by *dname*. The *newrr_in* argument is currently unused but is intended for making update messages.

The **res_send**() routine sends a pre-formatted query and returns an answer. It will call **res_init**() if RES_INIT is not set, send the query to the local name server, and handle timeouts and retries. The length of the reply message is returned, or -1 if there were errors.

The **dn_comp**() function compresses the domain name *exp_dn* and stores it in *comp_dn*. The size of the compressed name is returned or -1 if there were errors. The size of the array pointed to by *comp_dn* is given by *length*. The compression uses an array of pointers *dnptrs* to previously-compressed names in the current message. The first pointer points to the beginning of the message and the list ends with NULL. The limit to the array is specified by *lastdnptr*. A side effect of **dn_comp**() is to update the list of pointers for labels inserted into the message as the name is compressed. If *dnptr* is NULL, names are not compressed. If *lastdnptr* is NULL, the list of labels is not updated.

The **dn_expand**() entry expands the compressed domain name *comp_dn* to a full domain name The compressed name is contained in a query or reply message; *msg* is a pointer to the beginning of the message. The uncompressed name is placed in the buffer indicated by *exp_dn* which is of size *length*. The size of compressed name is returned or -1 if there was an error.

The **dn_skipname**() function skips over a compressed domain name, which starts at a location pointed to by *comp_dn*. The compressed name is contained in a query or reply message; *eom* is a pointer to the end of the message. The size of compressed name is returned or -1 if there was an error.

The **ns_get16**() function gets a 16-bit quantity from a buffer pointed to by *src*.

The **ns_get32**() function gets a 32-bit quantity from a buffer pointed to by *src*.

The **ns_put16**() function puts a 16-bit quantity *src* to a buffer pointed to by *dst*.

The **ns_put32**() function puts a 32-bit quantity *src* to a buffer pointed to by *dst*.

IMPLEMENTATION NOTES

This implementation of the resolver is thread-safe, but it will not function properly if the programmer

attempts to declare his or her own *_res* structure in an attempt to replace the per-thread version referred to by that macro.

The following compile-time option can be specified to change the default behavior of resolver routines when necessary.

RES_ENFORCE_RFC1034 If this symbol is defined during compile-time, **res_search**() will enforce RFC 1034 check, namely, disallow using of underscore character within host names. This is used by the standard host lookup routines like gethostbyname(3). For compatibility reasons this option is not enabled by default.

RETURN VALUES

The **res_init**() function will return 0 on success, or -1 in a threaded program if per-thread storage could not be allocated.

The **res_mkquery**(), **res_search**(), and **res_query**() functions return the size of the response on success, or -1 if an error occurs. The integer *h_errno* may be checked to determine the reason for error. See gethostbyname(3) for more information.

FILES

/etc/resolv.conf The configuration file, see resolver(5).

SEE ALSO

gethostbyname(3), resolver(5), hostname(7)

RFC1032, RFC1033, RFC1034, RFC1035, RFC974

HISTORY

The **res_query** function appeared in 4.3BSD.