NAME

res_ninit, res_nquery, res_nsearch, res_nquerydomain, res_nmkquery, res_nsend, res_init, res_query, res_search, res_querydomain, res_mkquery, res_send, dn_comp, dn_expand - resolver routines

SYNOPSIS

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

struct __res_state;
typedef struct __res_state *res_state;

int res_ninit(res_state statep);

int res_nquery(res_state statep,
 const char *dname, int class, int type,
 unsigned char *answer, int anslen);

int res_nsearch(res_state statep,
 const char *dname, int class, int type,
 unsigned char *answer, int anslen);

int res_nquerydomain(res_state statep,
 const char *name, const char *domain,
 int class, int type, unsigned char *answer,
 int anslen);

int res_nmkquery(res_state statep,
 int op, const char *dname, int class,
 int type, const unsigned char *data, int datalen,
 const unsigned char *newrr,
 unsigned char *buf, int buflen);

int res_nsend(res_state statep,
 const unsigned char *msg, int msglen,
 unsigned char *answer, int anslen);

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

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

Deprecated

```
extern struct __res_state _res;

int res_init(void);

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

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

int res_querydomain(const char *name, const char *domain,
 int class, int type, unsigned char *answer,
 int anslen);

int res_mkquery(int op, const char *dname, int class,
 int type, const unsigned char *data, int datalen,
```

const unsigned char ****newrr*,**
unsigned char ****buf*, int** *buflen***);**

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

Link with *-lresolv*.

DESCRIPTION

Note: This page is incomplete (various resolver functions provided by glibc are not described) and likely out of date.

The functions described below make queries to and interpret the responses from Internet domain name servers.

The API consists of a set of more modern, reentrant functions and an older set of nonreentrant functions that have been superseded. The traditional resolver interfaces such as **res_init**() and **res_query**() use some static (global) state stored in the *_res* structure, rendering these functions non-thread-safe. BIND 8.2 introduced a set of new interfaces **res_ninit**(), **res_nquery**(), and so on, which take a *res_state* as their first argument, so you can use a per-thread resolver state.

The **res_ninit**() and **res_init**() functions read the configuration files (see resolv.conf(5)) to get the default domain name and name server address(es). If no server is given, the local host is tried. If no domain is given, that associated with the local host is used. It can be overridden with the environment variable **LOCALDOMAIN**. **res_ninit**() or **res_init**() is normally executed by the first call to one of the other functions.

The **res_nquery**() and **res_query**() functions query the name server for the fully qualified domain name *name* of specified *type* and *class*. The reply is left in the buffer *answer* of length *anslen* supplied by the caller.

The **res_nsearch**() and **res_search**() functions make a query and waits for the response like **res_nquery**() and **res_query**(), but in addition they implement the default and search rules controlled by **RES_DEFNAMES** and **RES_DNSRCH** (see description of *_res* options below).

The **res_nquerydomain**() and **res_querydomain**() functions make a query using **res_nquery**()/**res_query**() on the concatenation of *name* and *domain*.

The following functions are lower-level routines used by **res_query**()/**res_query**().

The **res_nmkquery**() and **res_mkquery**() functions construct a query message in *buf* of length *buflen* for the domain name *dname*. The query type*op* is usually **Q UERY**, but can be any of the types defined in *<arpa/nameser.h>*. *newrr* is currently unused.

The **res_nsend**() and **res_send**() function send a preformatted query given in *msg* of length *msglen* and returns the answer in *answer* which is of length *anslen*. They will call **res_ninit**()/**res_init**() if it has not already been called.

The **dn_comp**() function compresses the domain name *exp_dn* and stores it in the buffer *comp_dn* of length *length*. The compression uses an array of pointers*dnptr s* 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 of the array is specified by *lastdnptr*. If*dnptr* is NULL, domain names are not compressed. If *lastdnptr* is NULL, the list of labels is not updated.

The **dn_expand**() function expands the compressed domain name *comp_dn* to a full domain name, which is placed in the buffer *exp_dn* of size *length*. The compressed name is contained in a query or reply message, and *msg* points to the beginning of the message.

The resolver routines use configuration and state information contained in a *__res_state* structure (either passed as the *statep* argument, or in the global variable *_res*, in the case of the older nonreentrant functions). The only field of this structure that is normally manipulated by the user is the*options* field. This field can contain the bitwise "OR" of the following options:

RES_INIT
True if **res_ninit**() or **res_init**() has been called.

RES_DEBUG
Print debugging messages. This option is available only if glibc was built with debugging enabled, which is not the default.

RES_AAONLY (unimplemented; deprecated in glibc 2.25)
Accept authoritative answers only. **res_send**() continues until it finds an authoritative answer or returns an error. This option was present but unimplemented in glibc until version 2.24; since glibc 2.25, it is deprecated, and its usage produces a warning.

RES_USEVC
Use TCP connections for queries rather than UDP datagrams.

RES_PRIMARY (unimplemented; deprecated in glibc 2.25)
Query primary domain name server only. This option was present but unimplemented in glibc until version 2.24; since glibc 2.25, it is deprecated, and its usage produces a warning.

RES_IGNTC
Ignore truncation errors. Don't retry with TCP.

RES_RECURSE
Set the recursion desired bit in queries. Recursion is carried out by the domain name server, not by **res_send**(). [Enabled by default].

RES_DEFNAMES
If set, **res_search**() will append the default domain name to single component names—that is, those that do not contain a dot. [Enabled by default].

RES_STAYOPEN
Used with **RES_USEVC** to keep the TCP connection open between queries.

RES_DNSRCH
If set, **res_search**() will search for hostnames in the current domain and in parent domains. This option is used by gethostbyname(3). [Enabled by default].

RES_INSECURE1
Accept a response from a wrong server. This can be used to detect potential security hazards, but you need to compile glibc with debugging enabled and use **RES_DEBUG** option (for debug purpose only).

RES_INSECURE2
Accept a response which contains a wrong query. This can be used to detect potential security hazards, but you need to compile glibc with debugging enabled and use **RES_DEBUG** option (for debug purpose only).

RES_NOALIASES
Disable usage of **HOSTALIASES** environment variable.

RES_USE_INET6
Try an AAAA query before an A query inside the gethostbyname(3) function, and map IPv4 responses in IPv6 "tunneled form" if no AAAA records are found but an A record set exists. Since glibc 2.25, this option is deprecated, and its usage produces a warning; applications should use getaddrinfo(3), rather than gethostbyname(3).

RES_ROTATE
Causes round-robin selection of name servers from among those listed. This has the effect of spreading the query load among all listed servers, rather than having all clients try the first listed server first every time.

RES_NOCHECKNAME (unimplemented; deprecated in glibc 2.25)
Disable the modern BIND checking of incoming hostnames and mail names for invalid characters such as underscore (_), non-ASCII, or control characters. This option was present in glibc until version 2.24; since glibc 2.25, it is deprecated, and its usage produces a warning.

RES_KEEPTSIG (unimplemented; deprecated in glibc 2.25)
Do not strip TSIG records. This option was present but unimplemented in glibc until version 2.24; since glibc 2.25, it is deprecated, and its usage produces a warning.

RES_BLAST (unimplemented; deprecated in glibc 2.25)
Send each query simultaneously and recursively to all servers. This option was present but unimplemented in glibc until version 2.24; since glibc 2.25, it is deprecated, and its usage produces a warning.

RES_USEBSTRING (glibc 2.3.4 to 2.24)
Make reverse IPv6 lookups using the bit-label format described in RFC 2673; if this option is not set (which is the default), then nibble format is used. This option was removed in glibc 2.25, since it relied on a backward-incompatible DNS extension that was never deployed on the Internet.

RES_NOIP6DOTINT (glibc 2.24 and earlier)
Use *ip6.arpa* zone in IPv6 reverse lookup instead of *ip6.int*, which is deprecated since glibc 2.3.4. This option is present in glibc up to and including version 2.24, where it is enabled by default. In glibc 2.25, this option was removed.

RES_USE_EDNS0 (since glibc 2.6)
Enables support for the DNS extensions (EDNS0) described in RFC 2671.

RES_SNGLKUP (since glibc 2.10)
By default, glibc performs IPv4 and IPv6 lookups in parallel since version 2.9. Some appliance DNS servers cannot handle these queries properly and make the requests time out. This option disables the behavior and makes glibc perform the IPv6 and IPv4 requests sequentially (at the cost of some slowdown of the resolving process).

RES_SNGLKUPREOP
When **RES_SNGLKUP** option is enabled, opens a new socket for the each request.

RES_USE_DNSSEC
Use DNSSEC with OK bit in OPT record. This option implies **RES_USE_EDNS0**.

RES_NOTLDQUERY
Do not look up unqualified name as a top-level domain (TLD).

RES_DEFAULT
Default option which implies: **RES_RECURSE**, **RES_DEFNAMES**, **RES_DNSRCH** and **RES_NOIP6DOTINT**.

RETURN VALUE

The **res_ninit**() and **res_init**() functions return 0 on success, or -1 if an error occurs.

The **res_nquery**(), **res_query**(), **res_nsearch**(), **res_search**(), **res_nquerydomain**(), **res_querydomain**(), **res_nmkquery**(), **res_mkquery**(), **res_nsend**(), and **res_send**() functions return the length of the response, or -1 if an error occurs.

The **dn_comp**() and **dn_expand**() functions return the length of the compressed name, or -1 if an error occurs.

FILES

/etc/resolv.conf resolver configuration file
/etc/host.conf resolver configuration file

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
res_ninit(), **res_nquery**(), **res_nsearch**(), **res_nquerydomain**(), **res_nsend**()	Thread safety	MT-Safe locale
res_nmkquery(), **dn_comp**(), **dn_expand**()	Thread safety	MT-Safe

CONFORMING TO

4.3BSD.

SEE ALSO

gethostbyname(3), resolv.conf(5), resolver(5), hostname(7), **named(8)**

The GNU C library source file *resolv/README*.

COLOPHON

This page is part of release 4.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.