## NAME

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>**
**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***);**

**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***);**

Link with *-lresolv*.

## DESCRIPTION

These functions make queries to and interpret the responses from Internet domain name servers.

The **res_init**() function reads the configuration files (see resolv.conf(5)) to get the default domain name, search order 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_init**() is normally executed by the first call to one of the other functions.

The **res_query**() function queries 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_search**() function makes a query and waits for the response like **res_query**(), but in addition implements the default and search rules controlled by **RES_DEFNAMES** and **RES_DNSRCH** (see description of _*res* options below).

The **res_querydomain**() function makes a query using **res_query**() on the concatenation of *name* and *domain*.

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

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

The **res_send**() function sends a preformatted query given in *msg* of length *msglen* and returns the answer in *answer* which is of length *anslen*. It will call **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 *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 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 global configuration and state information contained in the structure *_res*, which is defined in *<resolv.h>*. The only field that is normally manipulated by the user is *_res.options*. This field can contain the bitwise OR of the following options:

**RES_INIT**
> True if **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**
> Accept authoritative answers only. **res_send**() continues until it finds an authoritative answer or returns an error. [Not currently implemented].

**RES_USEVC**
> Use TCP connections for queries rather than UDP datagrams.

**RES_PRIMARY**
> Query primary domain name server only.

**RES_IGNTC**
> Ignore truncation errors. Don't retry with TCP. [Not currently implemented].

**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].

This list is not complete. You can find some other flags described in resolv.conf(5).

## RETURN VALUE

The **res_init**() function returns 0 on success, or -1 if an error occurs.

The **res_query**(), **res_search**(), **res_querydomain**(), **res_mkquery**() 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

**CONFORMING TO**

      4.3BSD.

**SEE ALSO**

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

**COLOPHON**

      This page is part of release 3.74 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 http://www.kernel.org/doc/man-pages/.