NAME

resolv.conf − resolver configuration file

SYNOPSIS

/etc/resolv.conf

DESCRIPTION

The *resolver* is a set of routines in the C library that provide access to the Internet Domain Name System (DNS). The resolver configuration file contains information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information. The configuration file is considered a trusted source of DNS information; see the **trust-ad** option below for details.

If this file does not exist, only the name server on the local machine will be queried, and the search list contains the local domain name determined from the hostname.

The different configuration options are:

nameserver Name server IP address

Internet address of a name server that the resolver should query, either an IPv4 address (in dot notation), or an IPv6 address in colon (and possibly dot) notation as per RFC 2373. Up to **MAXNS** (currently 3, see *<resolv.h>*) name servers may be listed, one per keyword. If there are multiple servers, the resolver library queries them in the order listed. If no **nameserver** entries are present, the default is to use the name server on the local machine. (The algorithm used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made.)

search Search list for host-name lookup.

By default, the search list contains one entry, the local domain name. It is determined from the local hostname returned by **gethostname**(2); the local domain name is taken to be everything after the first '.'. Finally, if the hostname does not contain a '.', the root domain is assumed as the local domain name.

This may be changed by listing the desired domain search path following the *search* keyword with spaces or tabs separating the names. Resolver queries having fewer than *ndots* dots (default is 1) in them will be attempted using each component of the search path in turn until a match is found. For environments with multiple subdomains please read **options ndots:***n* below to avoid man-in-the-middle attacks and unnecessary traffic for the root-dns-servers. Note that this process may be slow and will generate a lot of network traffic if the servers for the listed domains are not local, and that queries will time out if no server is available for one of the domains.

If there are multiple **search** directives, only the search list from the last instance is used.

In glibc 2.25 and earlier, the search list is limited to six domains with a total of 256 characters. Since glibc 2.26, the search list is unlimited.

The **domain** directive is an obsolete name for the **search** directive that handles one search list entry only.

sortlist This option allows addresses returned by **gethostbyname**(3) to be sorted. A sortlist is specified by IP-address-netmask pairs. The netmask is optional and defaults to the natural netmask of the net. The IP address and optional network pairs are separated by slashes. Up to 10 pairs may be specified. Here is an example:

```
sortlist 130.155.160.0/255.255.240.0 130.155.0.0
```

options

Options allows certain internal resolver variables to be modified. The syntax is

options *option* ...

where *option* is one of the following:

debug Sets **RES_DEBUG** in *_res.options* (effective only if glibc was built with debug support; see **resolver**(3)).

ndots:*n*
Sets a threshold for the number of dots which must appear in a name given to **res_query**(3) (see **resolver**(3)) before an *initial absolute query* will be made. The default for *n* is 1, meaning that if there are any dots in a name, the name will be tried first as an absolute name before any *search list* elements are appended to it. The value for this option is silently capped to 15.

timeout:*n*
Sets the amount of time the resolver will wait for a response from a remote name server before retrying the query via a different name server. This may**not** be the total time taken by any resolver API call and there is no guarantee that a single resolver API call maps to a single timeout. Measured in seconds, the default is **RES_TIMEOUT** (currently 5, see <*resolv.h*>). The value for this option is silently capped to 30.

attempts:*n*
Sets the number of times the resolver will send a query to its name servers before giving up and returning an error to the calling application. The default is **RES_DFLRETRY** (currently 2, see <*resolv.h*>). The value for this option is silently capped to 5.

rotate Sets **RES_ROTATE** in *_res.options*, which 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.

no−check−names
Sets **RES_NOCHECKNAME** in *_res.options*, which disables the modern BIND checking of incoming hostnames and mail names for invalid characters such as underscore (_), non-ASCII, or control characters.

inet6 Sets **RES_USE_INET6** in *_res.options*. This has the effect of trying an AAAA query before an A query inside the **gethostbyname**(3) function, and of mapping 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; applications should use **getaddrinfo**(3), rather than **gethostbyname**(3).

ip6−bytestring (since glibc 2.3.4 to glibc 2.24)
Sets **RES_USEBSTRING** in *_res.options*. This causes reverse IPv6 lookups to be made 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.

ip6−dotint/**no−ip6−dotint** (glibc 2.3.4 to glibc 2.24)
Clear/set **RES_NOIP6DOTINT** in *_res.options*. When this option is clear (**ip6−dotint**), reverse IPv6 lookups are made in the (deprecated) *ip6.int* zone; when this option is set (**no−ip6−dotint**), reverse IPv6 lookups are made in the *ip6.arpa* zone by default. These options are available up to glibc 2.24, where **no−ip6−dotint** is the default. Since **ip6−dotint** support long ago ceased to be available on the Internet, these options were removed in glibc 2.25.

edns0 (since glibc 2.6)
Sets **RES_USE_EDNS0** in *_res.options*. This enables support for the DNS extensions described in RFC 2671.

single−request (since glibc 2.10)
Sets **RES_SNGLKUP** in *_res.options*. By default, glibc performs IPv4 and IPv6 lookups in parallel since glibc 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).

single−request−reopen (since glibc 2.9)
Sets **RES_SNGLKUPREOP** in *_res.options*. The resolver uses the same socket for the A and AAAA requests. Some hardware mistakenly sends back only one reply. When that happens the client system will sit and wait for the second reply. Turning this option on changes this behavior so that if two requests from the same port are not handled correctly it will close the socket and open a new one before sending the second request.

no−tld−query (since glibc 2.14)
Sets **RES_NOTLDQUERY** in *_res.options*. This option causes**r es_nsearch**() to not attempt to resolve an unqualified name as if it were a top level domain (TLD). This option can cause problems if the site has ‘‘localhost’’ as a TLD rather than having localhost on one or more elements of the search list. This option has no effect if neither RES_DEFNAMES or RES_DNSRCH is set.

use−vc (since glibc 2.14)
Sets **RES_USEVC** in *_res.options*. This option forces the use of TCP for DNS resolutions.

no−reload (since glibc 2.26)
Sets **RES_NORELOAD** in *_res.options*. This option disables automatic reloading of a changed configuration file.

trust−ad (since glibc 2.31)
Sets **RES_TRUSTAD** in *_res.options*. This option controls the AD bit behavior of the stub resolver. If a validating resolver sets the AD bit in a response, it indicates that the data in the response was verified according to the DNSSEC protocol. In order to rely on the AD bit, the local system has to trust both the DNSSEC-validating resolver and the network path to it, which is why an explicit opt-in is required. If the **trust−ad** option is active, the stub resolver sets the AD bit in outgoing DNS queries (to enable AD bit support), and preserves the AD bit in responses. Without this option, the AD bit is not set in queries, and it is always removed from responses before they are returned to the application. This means that applications can trust the AD bit in responses if the**trust−ad** option has been set correctly.

In glibc 2.30 and earlier, the AD is not set automatically in queries, and is passed through unchanged to applications in responses.

The *search* keyword of a system’s *resolv.conf* file can be overridden on a per-process basis by setting the environment variable **LOCALDOMAIN** to a space-separated list of search domains.

The *options* keyword of a system’s *resolv.conf* file can be amended on a per-process basis by setting the environment variable **RES_OPTIONS** to a space-separated list of resolver options as explained above under **options**.

The keyword and value must appear on a single line, and the keyword (e.g., **nameserver**) must start the line. The value follows the keyword, separated by white space.

Lines that contain a semicolon (;) or hash character (#) in the first column are treated as comments.

FILES

/etc/resolv.conf , *<resolv.h>*

SEE ALSO

gethostbyname(3), **resolver**(3), **host.conf**(5), **hosts**(5), **nsswitch.conf**(5), **hostname**(7), **named**(8)

Name Server Operations Guide for BIND