NAME

ares_init, ares_init_options − Initialize a resolver channel

SYNOPSIS

#include <ares.h>

int ares_init(ares_channel ****channelptr*)**
int ares_init_options(ares_channel ****channelptr*,**
    struct ares_options ****options*, int** *optmask***)**

cc file.c -lcares

DESCRIPTION

The **ares_init** function initializes a communications channel for name service lookups. If it returns successfully, **ares_init** will set the variable pointed to by *channelptr* to a handle used to identify the name service channel. The caller should invoke **ares_destroy**(3) on the handle when the channel is no longer needed.

The **ares_init_options** function also initializes a name service channel, with additional options useful for applications requiring more control over name service configuration. The *optmask* parameter specifies which fields in the structure pointed to by *options* are set, as follows:

ARES_OPT_FLAGS
: **int** *flags***;**
Flags controlling the behavior of the resolver. See below for a description of possible flag values.

ARES_OPT_TIMEOUT
: **int** *timeout***;**
The number of seconds each name server is given to respond to a query on the first try. (After the first try, the timeout algorithm becomes more complicated, but scales linearly with the value of *timeout*.) The default is five seconds. This option is being deprecated by *ARES_OPT_TIMEOUTMS* starting in c-ares 1.5.2.

ARES_OPT_TIMEOUTMS
: **int** *timeout***;**
The number of milliseconds each name server is given to respond to a query on the first try. (After the first try, the timeout algorithm becomes more complicated, but scales linearly with the value of *timeout*.) The default is five seconds. Note that this option is specified with the same struct field as the former *ARES_OPT_TIMEOUT*, it is but the option bits that tell c-ares how to interpret the number. This option was added in c-ares 1.5.2.

ARES_OPT_TRIES
: **int** *tries***;**
The number of tries the resolver will try contacting each name server before giving up. The default is four tries.

ARES_OPT_NDOTS
: **int** *ndots***;**
The number of dots which must be present in a domain name for it to be queried for "as is" prior to querying for it with the default domain extensions appended. The default value is 1 unless set otherwise by resolv.conf or the RES_OPTIONS environment variable.

ARES_OPT_PORT **unsigned short** *port***;**
The port to use for queries (both TCP and UDP), in network byte order. The default value is 53 (in network byte order), the standard name service port.

ARES_OPT_SERVERS
: **struct in_addr ****servers***;**
int *nservers***;**

The list of IPv4 servers to contact, instead of the servers specified in resolv.conf or the local named. In order to allow specification of either IPv4 or IPv6 name servers, function **ares_set_servers(3)** must be used instead.

ARES_OPT_DOMAINS
char *****domains*;**
int *ndomains***;**
The domains to search, instead of the domains specified in resolv.conf or the domain derived from the kernel hostname variable.

ARES_OPT_LOOKUPS
char ****lookups*;**
The lookups to perform for host queries. *lookups* should be set to a string of the characters "b" or "f", where "b" indicates a DNS lookup and "f" indicates a lookup in the hosts file.

ARES_OPT_SOCK_STATE_CB
void (**sock_state_cb***)(void *data, int s, int read, int write);**
void ****sock_state_cb_data*;**
A callback function to be invoked when a socket changes state. *s* will be passed the socket whose state has changed; *read* will be set to true if the socket should listen for read events, and *write* will be set to true if the socket should listen for write events. The value of *sock_state_cb_data* will be passed as the *data* argument.

The *flags* field should be the bitwise or of some subset of the following values:

ARES_FLAG_USEVC Always use TCP queries (the "virtual circuit") instead of UDP queries. Normally, TCP is only used if a UDP query yields a truncated result.

ARES_FLAG_PRIMARY
Only query the first server in the list of servers to query.

ARES_FLAG_IGNTC If a truncated response to a UDP query is received, do not fall back to TCP; simply continue on with the truncated response.

ARES_FLAG_NORECURSE
Do not set the "recursion desired" bit on outgoing queries, so that the name server being contacted will not try to fetch the answer from other servers if it doesn't know the answer locally. Be aware that ares will not do the recursion for you. Recursion must be handled by the application calling ares if *ARES_FLAG_NORECURSE* is set.

ARES_FLAG_STAYOPEN
Do not close communications sockets when the number of active queries drops to zero.

ARES_FLAG_NOSEARCH
Do not use the default search domains; only query hostnames as-is or as aliases.

ARES_FLAG_NOALIASES
Do not honor the HOSTALIASES environment variable, which normally specifies a file of hostname translations.

ARES_FLAG_NOCHECKRESP
Do not discard responses with the SERVFAIL, NOTIMP, or REFUSED response code or responses whose questions don't match the questions in the request. Primarily useful for writing clients which might be used to test or debug name servers.

RETURN VALUES

ares_init or *ares_init_options* can return any of the following values:

ARES_SUCCESS
Initialization succeeded.

ARES_EFILE A configuration file could not be read.

ARES_ENOMEM
The process's available memory was exhausted.

ARES_ENOTINITIALIZED
c-ares library initialization not yet performed.

SEE ALSO

ares_destroy(3), ares_dup(3), ares_library_init(3), ares_set_servers(3)

AUTHOR

Greg Hudson, MIT Information Systems
Copyright 1998 by the Massachusetts Institute of Technology.
Copyright (C) 2004-2010 by Daniel Stenberg.