DNS Query Code in C with Linux sockets

By Silver Moon | May 18, 2020 12 Comments

DNS Query Code in C

A dns query lets you to lookup the various dns records for a given hostname or domain name on the internet. The simplest use case is to lookup the ip address of a domain name.

When you open a url in your browser, it first performs a dns query to find out the ip address of the request url domain. Its the dns servers that respond with the correct answer which the browser then uses to connect.

So in this article we shall write a simple function to perform a dns query using sockets in C. This code works on Linux platform and can be compiled and run using gcc command.

We wrote a similar function for the windows platform using the winsock socket api that can be found here:

https://www.binarytides.com/blog/dns-query-code-in-c-with-winsock/

The Linux version has some differences. On Linux the dns server ips are stored in a file called **/etc/resolv.conf**. So the get_dns_servers function will open this file and pickup the dns server ip addresses.

A typical /etc/resolv.conf file can look like this :

```
# Generated by NetworkManager nameserver 208.67.222.222 nameserver 208.67.220.220
```

So the lines starting with `nameserver` can be picked up and broken into parts using strtok.

Rest of the code is very much similar to the winsock version. A udp socket is used to send a UDP packet and the response is analysed.

Code :

Code

```
// DNS Query Program on Linux

//Header Files
#include<stdio.h>   //printf
#include<string.h>  //strlen
#include<stdlib.h>  //malloc
#include<sys/socket.h>  //you know what this is for
#include<arpa/inet.h>   //inet_addr , inet_ntoa , ntohs etc
#include<netinet/in.h>
#include<unistd.h>  //getpid

//List of DNS Servers registered on the system
char dns_servers[10][100];
```

SEARCH

SEARCH

RELATED POSTS

How to Code a Server and Client in C with Sockets on Linux - Code Examples
C

DNS Query Code in C with winsock
Winsock

How to Code Raw Sockets in C on Linux
C

How to Get IP Whois Data in C with Sockets on Linux - Code Example
C

How to Code a Tcp Syn Port Scanner in C with Linux Sockets
C

How to code Packet Sniffer in C with Sockets on Linux

```c
int dns_server_count = 0;
//Types of DNS resource records :)

#define T_A 1 //Ipv4 address
#define T_NS 2 //Nameserver
#define T_CNAME 5 // canonical name
#define T_SOA 6 /* start of authority zone */
#define T_PTR 12 /* domain name pointer */
#define T_MX 15 //Mail server

//Function Prototypes
void ngethostbyname (unsigned char* , int);
void ChangetoDnsNameFormat (unsigned char*,unsigned char*);
unsigned char* ReadName (unsigned char*,unsigned char*,int*);
void get_dns_servers();

//DNS header structure
struct DNS_HEADER
{
    unsigned short id; // identification number

    unsigned char rd :1; // recursion desired
    unsigned char tc :1; // truncated message
    unsigned char aa :1; // authoritive answer
    unsigned char opcode :4; // purpose of message
    unsigned char qr :1; // query/response flag

    unsigned char rcode :4; // response code
    unsigned char cd :1; // checking disabled
    unsigned char ad :1; // authenticated data
    unsigned char z :1; // its z! reserved
    unsigned char ra :1; // recursion available

    unsigned short q_count; // number of question entries
    unsigned short ans_count; // number of answer entries
    unsigned short auth_count; // number of authority entries
    unsigned short add_count; // number of resource entries
};

//Constant sized fields of query structure
struct QUESTION
{
    unsigned short qtype;
    unsigned short qclass;
};

//Constant sized fields of the resource record structure
#pragma pack(push, 1)
struct R_DATA
{
    unsigned short type;
    unsigned short _class;
    unsigned int ttl;
    unsigned short data_len;
};
#pragma pack(pop)

//Pointers to resource record contents
struct RES_RECORD
{
    unsigned char *name;
    struct R_DATA *resource;
    unsigned char *rdata;
};

//Structure of a Query
typedef struct
{
    unsigned char *name;
    struct QUESTION *ques;
} QUERY;

int main( int argc , char *argv[])
{
    unsigned char hostname[100];

    //Get the DNS servers from the resolv.conf file
    get_dns_servers();

    //Get the hostname from the terminal
    printf("Enter Hostname to Lookup : ");
    scanf("%s" , hostname);

    //Now get the ip of this hostname , A record
    ngethostbyname(hostname , T_A);

    return 0;
}

/*
```

C

FOLLOW US

```
 * Perform a DNS query by sending a packet
 * */
void ngethostbyname(unsigned char *host , int query_type)
{
    unsigned char buf[65536],*qname,*reader;
    int i , j , stop , s;

    struct sockaddr_in a;

    struct RES_RECORD answers[20],auth[20],addit[20]; //the replies from the DNS server
    struct sockaddr_in dest;

    struct DNS_HEADER *dns = NULL;
    struct QUESTION *qinfo = NULL;

    printf("Resolving %s" , host);

    s = socket(AF_INET , SOCK_DGRAM , IPPROTO_UDP); //UDP packet for DNS queries

    dest.sin_family = AF_INET;
    dest.sin_port = htons(53);
    dest.sin_addr.s_addr = inet_addr(dns_servers[0]); //dns servers

    //Set the DNS structure to standard queries
    dns = (struct DNS_HEADER *)&buf;

    dns->id = (unsigned short) htons(getpid());
    dns->qr = 0; //This is a query
    dns->opcode = 0; //This is a standard query
    dns->aa = 0; //Not Authoritative
    dns->tc = 0; //This message is not truncated
    dns->rd = 1; //Recursion Desired
    dns->ra = 0; //Recursion not available! hey we dont have it (lol)
    dns->z = 0;
    dns->ad = 0;
    dns->cd = 0;
    dns->rcode = 0;
    dns->q_count = htons(1); //we have only 1 question
    dns->ans_count = 0;
    dns->auth_count = 0;
    dns->add_count = 0;

    //point to the query portion
    qname =(unsigned char*)&buf[sizeof(struct DNS_HEADER)];

    ChangetoDnsNameFormat(qname , host);
    qinfo =(struct QUESTION*)&buf[sizeof(struct DNS_HEADER) + (strlen((const char*)qname) + 1)]; //fill

    qinfo->qtype = htons( query_type ); //type of the query , A , MX , CNAME , NS etc
    qinfo->qclass = htons(1); //its internet (lol)

    printf("\nSending Packet...");
    if( sendto(s,(char*)buf,sizeof(struct DNS_HEADER) + (strlen((const char*)qname)+1) + sizeof(struct Q
    {
        perror("sendto failed");
    }
    printf("Done");

    //Receive the answer
    i = sizeof dest;
    printf("\nReceiving answer...");
    if(recvfrom (s,(char*)buf , 65536 , 0 , (struct sockaddr*)&dest , (socklen_t*)&i ) < 0)
    {
        perror("recvfrom failed");
    }
    printf("Done");

    dns = (struct DNS_HEADER*) buf;

    //move ahead of the dns header and the query field
    reader = &buf[sizeof(struct DNS_HEADER) + (strlen((const char*)qname)+1) + sizeof(struct QUESTION)];

    printf("\nThe response contains : ");
    printf("\n %d Questions.",ntohs(dns->q_count));
    printf("\n %d Answers.",ntohs(dns->ans_count));
    printf("\n %d Authoritative Servers.",ntohs(dns->auth_count));
    printf("\n %d Additional records.\n\n",ntohs(dns->add_count));

    //Start reading answers
    stop=0;

    for(i=0;i<ntohs(dns->ans_count);i++)
    {
        answers[i].name=ReadName(reader,buf,&stop);
        reader = reader + stop;

        answers[i].resource = (struct R_DATA*)(reader);
        reader = reader + sizeof(struct R_DATA);

        if(ntohs(answers[i].resource->type) == 1) //if its an ipv4 address
        {
            answers[i].rdata = (unsigned char*)malloc(ntohs(answers[i].resource->data_len));

            for(j=0 ; j<ntohs(answers[i].resource->data_len) ; j++)
            {
                answers[i].rdata[j]=reader[j];
            }

            answers[i].rdata[ntohs(answers[i].resource->data_len)] = '\0';

            reader = reader + ntohs(answers[i].resource->data_len);
        }
        else
        {
            answers[i].rdata = ReadName(reader,buf,&stop);
            reader = reader + stop;
        }
    }

    //read authorities
    for(i=0;i<ntohs(dns->auth_count);i++)
    {
        auth[i].name=ReadName(reader,buf,&stop);
        reader+=stop;

        auth[i].resource=(struct R_DATA*)(reader);
        reader+=sizeof(struct R_DATA);

        auth[i].rdata=ReadName(reader,buf,&stop);
        reader+=stop;
    }

    //read additional
    for(i=0;i<ntohs(dns->add_count);i++)
    {
        addit[i].name=ReadName(reader,buf,&stop);
        reader+=stop;

        addit[i].resource=(struct R_DATA*)(reader);
        reader+=sizeof(struct R_DATA);

        if(ntohs(addit[i].resource->type)==1)
        {
            addit[i].rdata = (unsigned char*)malloc(ntohs(addit[i].resource->data_len));
            for(j=0;j<ntohs(addit[i].resource->data_len);j++)
            addit[i].rdata[j]=reader[j];

            addit[i].rdata[ntohs(addit[i].resource->data_len)]='\0';
            reader+=ntohs(addit[i].resource->data_len);
        }
        else
        {
            addit[i].rdata=ReadName(reader,buf,&stop);
            reader+=stop;
        }
    }

    //print answers
    printf("\nAnswer Records : %d \n" , ntohs(dns->ans_count) );
    for(i=0 ; i < ntohs(dns->ans_count) ; i++)
    {
        printf("Name : %s ",answers[i].name);

        if( ntohs(answers[i].resource->type) == T_A) //IPv4 address
        {
            long *p;
            p=(long*)answers[i].rdata;
            a.sin_addr.s_addr=(*p); //working without ntohl
            printf("has IPv4 address : %s",inet_ntoa(a.sin_addr));
        }

        if(ntohs(answers[i].resource->type)==5)
        {
            //Canonical name for an alias
            printf("has alias name : %s",answers[i].rdata);
        }

        printf("\n");
    }

    //print authorities
    printf("\nAuthoritive Records : %d \n" , ntohs(dns->auth_count) );
    for( i=0 ; i < ntohs(dns->auth_count) ; i++)
    {

        printf("Name : %s ",auth[i].name);
        if(ntohs(auth[i].resource->type)==2)
        {
            printf("has nameserver : %s",auth[i].rdata);
        }
        printf("\n");
    }

    //print additional resource records
    printf("\nAdditional Records : %d \n" , ntohs(dns->add_count) );
    for(i=0; i < ntohs(dns->add_count) ; i++)
    {
        printf("Name : %s ",addit[i].name);
        if(ntohs(addit[i].resource->type)==1)
        {
            long *p;
            p=(long*)addit[i].rdata;
            a.sin_addr.s_addr=(*p);
            printf("has IPv4 address : %s",inet_ntoa(a.sin_addr));
        }
        printf("\n");
    }
    return;
}

/*
 *
 * */
u_char* ReadName(unsigned char* reader,unsigned char* buffer,int* count)
{
    unsigned char *name;
    unsigned int p=0,jumped=0,offset;
    int i , j;

    *count = 1;
    name = (unsigned char*)malloc(256);

    name[0]='\0';

    //read the names in 3www6google3com format
    while(*reader!=0)
    {
        if(*reader>=192)
        {
            offset = (*reader)*256 + *(reader+1) - 49152; //49152 = 11000000 00000000 ;)
            reader = buffer + offset - 1;
            jumped = 1; //we have jumped to another location so counting wont go up!
        }
        else
        {
            name[p++]=*reader;
        }

        reader = reader+1;

        if(jumped==0)
        {
            *count = *count + 1; //if we havent jumped to another location then we can count up
        }
    }

    name[p]='\0'; //string complete
    if(jumped==1)
    {
        *count = *count + 1; //number of steps we actually moved forward in the packet
    }

    //now convert 3www6google3com0 to www.google.com
    for(i=0;i<(int)strlen((const char*)name);i++)
    {
        p=name[i];
        for(j=0;j<(int)p;j++)
        {
            name[i]=name[i+1];
            i=i+1;
        }
        name[i]='.';
    }
    name[i-1]='\0'; //remove the last dot
    return name;
}

/*
 * Get the DNS servers from /etc/resolv.conf file on Linux
 * */
void get_dns_servers()
{
    FILE *fp;
    char line[200] , *p;
    if((fp = fopen("/etc/resolv.conf" , "r")) == NULL)
    {
        printf("Failed opening /etc/resolv.conf file \n");
    }

    while(fgets(line , 200 , fp))
    {
        if(line[0] == '#')
        {
            continue;
        }
        if(strncmp(line , "nameserver" , 10) == 0)
        {
            p = strtok(line , " ");
            p = strtok(NULL , " ");

            //p now is the dns ip :)
            //????
        }
    }

    strcpy(dns_servers[0] , "208.67.222.222");
    strcpy(dns_servers[1] , "208.67.220.220");
}

/*
 * This will convert www.google.com to 3www6google3com
 * got it :)
 * */
void ChangetoDnsNameFormat(unsigned char* dns,unsigned char* host)
{
    int lock = 0 , i;
    strcat((char*)host,".");

    for(i = 0 ; i < strlen((char*)host) ; i++)
    {
        if(host[i]=='.')
        {
            *dns++ = i-lock;
            for(;lock<i;lock++)
            {
                *dns++=host[lock];
            }
            lock++; //or lock=i+1;
        }
    }
    *dns++='\0';
}
```

Compile

```
$ gcc dns.c
```

Output

```
$ ./a.out
Enter Hostname to Lookup : www.google.com
Resolving www.google.com
Sending Packet...Done
Receiving answer...Done
The response contains :
 1 Questions.
 6 Answers.
 4 Authoritative Servers.
 4 Additional records.
Answer Records : 6
Name : www.google.com has alias name : www.l.google.com
Name : www.l.google.com has IPv4 address : 74.125.235.16
Name : www.l.google.com has IPv4 address : 74.125.235.17
Name : www.l.google.com has IPv4 address : 74.125.235.18
Name : www.l.google.com has IPv4 address : 74.125.235.19
Name : www.l.google.com has IPv4 address : 74.125.235.20
Authoritive Records : 4
Name : google.com has nameserver : ns2.google.com
Name : google.com has nameserver : ns1.google.com
Name : google.com has nameserver : ns3.google.com
Name : google.com has nameserver : ns4.google.com
```

```
Additional Records : 4
Name : ns1.google.com has IPv4 address : 216.239.32.10
Name : ns2.google.com has IPv4 address : 216.239.34.10
Name : ns3.google.com has IPv4 address : 216.239.36.10
Name : ns4.google.com has IPv4 address : 216.239.38.10
```

The same code can be used to fetch different kinds of dns records like Nameserver , MX , CNAME , SOA etc.

CATEGORY: C
TAGS: C , DNS , SOCKET PROGRAMMING

About Silver Moon

A Tech Enthusiast, Blogger, Linux Fan and a Software Developer. Writes about Computer hardware, Linux and Open Source software and coding in Python, Php and Javascript. He can be reached at binarytides@gmail.com.

View all posts by Silver Moon →

PREVIOUS

PHP benefits of storing session data in database

NEXT

Programming box2d in javascript – tutorial on basics

12 Comments

DNS Query Code in C with Linux sockets

bamless
June 10, 2018 at 10:10 pm

I know that this post is really old but i thought i should comment anyway, to help people who will stumble on this while browsing the web. This code shouldn't be used in any production code, because it violates c strict aliasing rules (more info: https://stackoverflow.com/questions/98650/what-is-the-strict-aliasing-rule?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa). When casting the pointer `buf` of unsigned chars to the struct DNS_HEADER (or in any other place that a cast like this happens) we're invoking undefined behaviour. A better solution would be to use a union to alias the char buffer to a struct DNS_HEADER (this is supported by the c99 and c11 standards), or to use a simple memcpy to copy the contents of the struct into the buffer.

Reply

Iharob Al Asimi
August 17, 2016 at 3:03 am

I really appreciate the effort. And it seems that you do understand how to perform a DNS query. But you write terrible c code. I suggest fixing it because it's really bad.

Reply

AD
November 12, 2014 at 10:10 pm

isnt aa part in dns header should be after opcode ? In your code you are using standard query which is 0 and aa which is again set to 0, that's why it is working fine i guess. reference : https://www.ietf.org/rfc/rfc1035.txt

Anyways thanks for snippet it is useful :)

Reply

Nick
November 10, 2014 at 5:05 pm

Is there some way I can get the TTL value from the answers portion?? I tried the following but it returns 0 (ntohs(answers[i].resource->ttl) majority of the time.

Reply

Nick
May 21, 2013 at 4:04 pm

Very nice article! Can you provide more information about what the ReadName functions does? In my opinion it's the most important function from the entire program and it's very poor explainded. Thank you!

Reply

Razvan
May 5, 2012 at 3:03 pm

Please ignore my previous comment,i saw that you also posted a linux version.I want to ask you , how do you fetch dns records for , lets say CNAME or PTR ?

Reply

Razvan
May 9, 2012 at 1:01 pm

Couldn't edit the post and i replied to it.Seems all is working but the PTR. I seem to not receive the information i need when i try to fetch the domain for an ip address , a little help with this ?

Reply

---

cristian_gog
May 14, 2012 at 2:02 pm

yeah .. is not implemented, & there are some problems interpreting the message when MX is used (because of the whitespaces i think)

Reply

---

ching

February 24, 2012 at 4:04 pm

i do not understand this part…

```
if(*reader>=192)
{
offset = (*reader)*256 + *(reader+1) – 49152; //49152 = 11000000 00000000 ;)
reader = buffer + offset – 1;
jumped = 1; //we have jumped to another location so counting wont go up!
}
else
{
name[p++]=*reader;
}

reader = reader+1;

if(jumped==0)
{
*count = *count + 1; //if we havent jumped to another location then we can count up
}
}

name[p]=“; //string complete
if(jumped==1)
{
*count = *count + 1; //number of steps we actually moved forward in the packet
}

```

rest of program is understood…
please help

Reply

Tylkas
February 9, 2012 at 7:07 pm

But I don't understand this moment:

```
~$ ./client
Enter Hostname to Lookup : asdqwe
Resolving asdqwe
…
Answer Records : 1
Name : asdqwe has IPv4 address : 67.215.65.132
```

Why 67.215.65.132 ?

Reply

Rajaona
April 12, 2012 at 10:10 pm

Thanks soo much …
Easily Understandable Code….
can you also provide the code for gethostbyaddr() function (in same pattern without using netdb.h library)

Reply

Tylkas

February 9, 2012 at 7:07 pm

Nice code, thank you =]

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

POST COMMENT