

DNS Operations Working Group  
PREN: 4  
Category: Informational

Peter M. Dambier  
April 28, 2005

gethostbyname() function Syntax Rules for  
processing an IPv4 Address or Host Name

Status of this Memo

These notes provide information to the Internet community. Public-Root Experimental Notes (PREN) do not specify an Internet standard of any kind. Distribution of this PREN is unlimited.

Copyright Notice

Copyright © The Public Root (2005). All Rights Reserved.

Abstract

The `gethostbyname()` function retrieves information about hosts that is stored in databases accessed sequentially or randomly. In many cases these databases are implemented by the Domain Name System (DNS) as documented in RFC 1034 and RFC 1035.

This note details the syntax rules used by `gethostbyname()` to differentiate between a host name and an IPv4 number. This note DOES NOT provide any detail on the syntax rules used by `gethostbyname` to support IPv6 numbers as documented in RFC 1886.

This note reviews the `gethostbyname()` function found in source code `gethostent.c` (APPENDIX A) and `gethnmadr.c` (APPENDIX B) copyright © 1985, 1988 by the Regents of the University of California.

General Information

Queries carried out by `gethostbyname()` use a combination of any or all of the name server named (DNS), a broken out line from `/etc/hosts`, and the Network Information Service (NIS or YP), depending upon the contents of the order line in `/etc/host.conf`. The default action is to query the named (DNS), followed by `/etc/hosts`.

The `gethostbyname()` function shall return an entry (structure of type `hostent`) containing addresses of address family `AF_INET` for the host name. The name argument of `gethostbyname()` shall be a node name or an IPv4 numeric address string in standard dot notation.

If name is an IPv4 numeric address no lookup is performed and `gethostbyname()` simply copies name into the `h_name` field and its struct `in_addr` equivalent into the `h_addr_list[0]` field of the returned `hostent` structure.

If name is not an IPv4 numeric address string and is an alias for a valid host name, then `gethostbyname()` shall return information about

the host name to which the alias refers, and name shall be included in the list of aliases returned.

If name doesn't end in a dot and the environment variable HOSTALIASES is set, the alias file pointed to by HOSTALIASES will first be searched for name. The current domain and its parents are searched unless name ends in a dot.

#### Syntax Rules

The gethostbyname() function uses both gethostent.c (APPENDIX A) and gethnmadr.c (APPENDIX B). The gethostent.c is used to read the /etc/hosts file and gethnmadr.c is used for DNS lookups.

The order in which gethostent.c or gethnmadr.c depends on the contents of the order line in /etc/host.conf. The default action is to query the DNS, followed by /etc/hosts.

The source code gethnmadr.c uses a very elegant approach to test if a string (STRING) is a domain name "NAME" or an IPv4 number "NUMBER". gethnmadr.c disallows a {STRING} consisting only of digits/dots, unless the {STRING} ends in a dot.

An all numeric {STRING} without a dot at the end will be treated as a NUMBER and gethnmadr.c will fake a hostent as if it had actually done a lookup. In any other case {STRING} is considered a NAME.

The logical steps used to test a {STRING} are:

For every character {CHAR} in string {STRING}

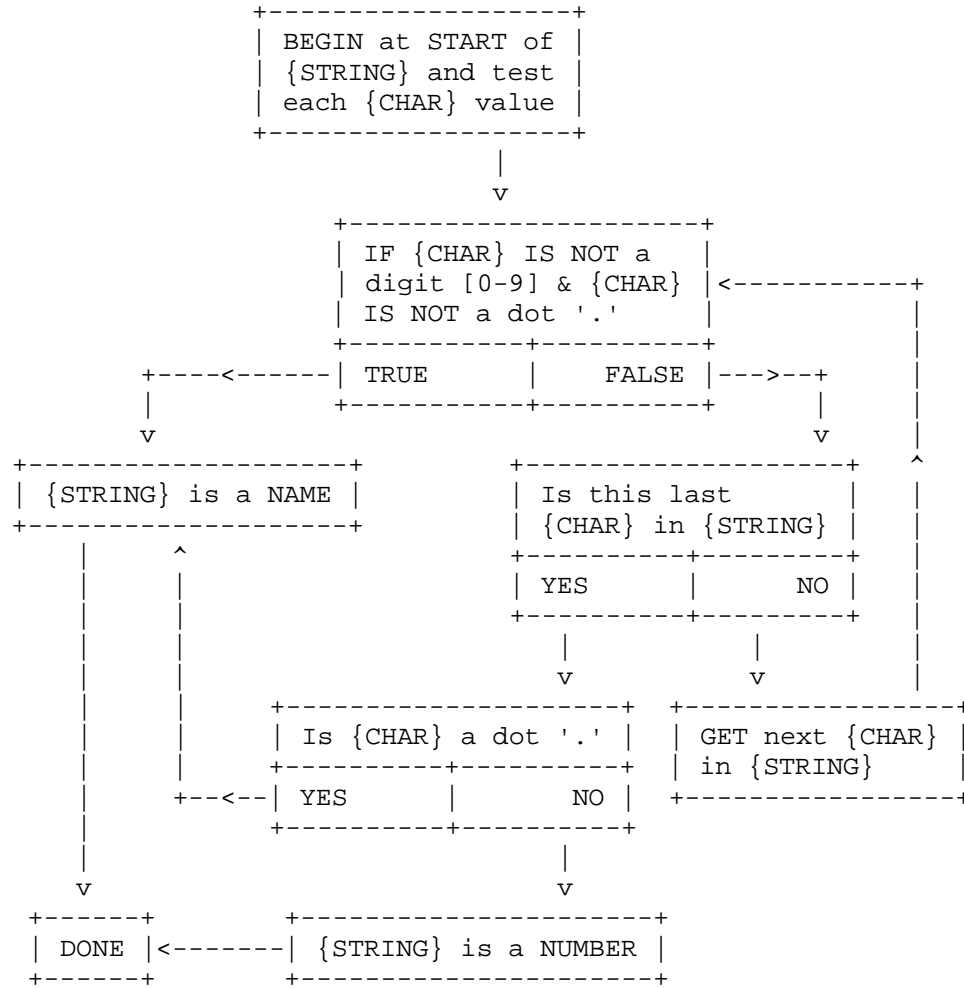
```
IF {CHAR} IS NOT the LAST character in {STRING},
  AND {CHAR} IS NOT a digit [0-9],
  AND {CHAR} IS NOT a dot '.',
THEN we are done {STRING} is a NAME.
```

```
IF {CHAR} is the LAST character in {STRING}
  AND {CHAR} IS a dot '.'
THEN {STRING} is a NAME.
```

```
ELSE {STRING} is a NUMBER.
```

These steps have been documented in figure 1 as a flow chart to provide a visual reference of the syntax rules.

Figure 1. Flow Chart - gethostbyname()



References

[RFC 1034] P. Mockapetris, "Domain Names - Concepts and Facilities," RFC-1034, USC/Information Sciences Institute, November 1987.

[RFC 1035] P. Mockapetris, "Domain names - Implementation and Specification," RFC-1035, USC/Information Sciences Institute, November 1987.

[RFC 1886] S. Thomson & C. Huitema, "DNS Extensions to support IP version 6," RFC-1886, Bellcore & INRIA, December 1995.

[Tannenbaum] Andrew S. Tannenbaum & Albert S. Woodhull, "Operating Systems Design and Implementation, Second Edition," Prentice Hall, 1987.

Editor's Address

Peter M. Dambier  
IASON  
Gräffstraße 14  
Heppenheim D-64646  
Germany

Phone: +49.6252.599091  
Email: peter@peter-dambier.de

```
/*
 * Copyright (c) 1985, 1988 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement: ``This product includes software
 * developed by the University of California, Berkeley and its contributors''
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)gethostnamadr.c 6.41 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

/* Prefix the functions defined here with underscores to distinguish them
 * from the newer replacements in the resolver library.
 */
#define sethostent _sethostent
#define endhostent _endhostent
#define gethostent _gethostent
#define gethostbyname _gethostbyname
#define gethostbyaddr _gethostbyaddr

#ifdef _MINIX
#include <sys/types.h>
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/resolv.h>
#include <net/gen/socket.h>
#else
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <ctype.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif /* !_MINIX */

#define MAXALIASES 35
#define MAXADDRS 35

#ifdef _MINIX
typedef u32_t u_long;
typedef u16_t u_short;
typedef u8_t u_char;
#endif

#define bcmp memcmp
#endif /* _MINIX */
```

```

static struct hostent host;
static char *host_aliases[MAXALIASES];
static char hostbuf[BUFSIZ+1];
static FILE *hostf = NULL;
static u_long hostaddr[(MAXADDRS+sizeof(u_long)-1)/sizeof(u_long)];
static char *host_addrs[2];
static int stayopen = 0;

#ifdef !_MINIX
char *strpbrk();
#endif /* !_MINIX */

void
sethostent(f)
    int f;
{
    if (hostf == NULL)
        hostf = fopen(_PATH_HOSTS, "r" );
    else
        rewind(hostf);
    stayopen |= f;
}

void
endhostent()
{
    if (hostf && !stayopen) {
        (void) fclose(hostf);
        hostf = NULL;
    }
}

struct hostent *
gethostent()
{
    char *p;
    register char *cp, **q;

    if (hostf == NULL && (hostf = fopen(_PATH_HOSTS, "r" )) == NULL)
        return (NULL);
again:
    if ((p = fgets(hostbuf, BUFSIZ, hostf)) == NULL)
        return (NULL);
    if (*p == '#')
        goto again;
    cp = strpbrk(p, "#\n");
    if (cp == NULL)
        goto again;
    *cp = '\0';
    cp = strpbrk(p, " \t");
    if (cp == NULL)
        goto again;
    *cp++ = '\0';
    /* THIS STUFF IS INTERNET SPECIFIC */
#ifdef BSD >= 43 || defined(h_addr) /* new-style hostent structure */
    host.h_addr_list = host_addrs;
#endif
    host.h_addr = (char *) hostaddr;
    *((u_long *)host.h_addr) = inet_addr(p);
    host.h_length = sizeof (u_long);
    host.h_addrtype = AF_INET;
    while (*cp == ' ' || *cp == '\t')
        cp++;
    host.h_name = cp;
    q = host.h_aliases = host_aliases;
    cp = strpbrk(cp, " \t");
    if (cp != NULL)
        *cp++ = '\0';
}

```

```

    while (cp && *cp) {
        if (*cp == ' ' || *cp == '\t') {
            cp++;
            continue;
        }
        if (q < &host_aliases[MAXALIASES - 1])
            *q++ = cp;
        cp = strpbrk(cp, " \t");
        if (cp != NULL)
            *cp++ = '\0';
    }
    *q = NULL;
    return (&host);
}

struct hostent *
gethostbyname(name)
    char *name;
{
    register struct hostent *p;
    register char **cp;

    sethostent(0);
    while (p = gethostent()) {
        if (strcasecmp(p->h_name, name) == 0)
            break;
        for (cp = p->h_aliases; *cp != 0; cp++)
            if (strcasecmp(*cp, name) == 0)
                goto found;
    }
found:
    endhostent();
    return (p);
}

struct hostent *
gethostbyaddr(addr, len, type)
    const char *addr;
    int len, type;
{
    register struct hostent *p;

    sethostent(0);
    while (p = gethostent())
        if (p->h_addrtype == type && !bcmp(p->h_addr, addr, len))
            break;
    endhostent();
    return (p);
}

```

Source: Andrew S. Tannenbaum & Albert S. Woodhull, "Operating Systems Design and Implementation, Second Edition," Prentice Hall, 1987.

```
/*
 * Copyright (c) 1985, 1988 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement: ``This product includes software
 * developed by the University of California, Berkeley and its contributors''
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)gethostnamadr.c 6.41 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#ifdef _MINIX
#include <sys/types.h>
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/resolv.h>
#include <net/gen/socket.h>
#else
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <ctype.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif /* AMOEABA */

#define MAXALIASES 35
#define MAXADDRS 35

static char *h_addr_ptrs[MAXADDRS + 1];

#ifdef _MINIX
struct in_addr
{
    ipaddr_t s_addr;
};
typedef u32_t u_long;
typedef u16_t u_short;
typedef u8_t u_char;
union querybuf;

extern int dn_skipname_ARGS(( const u_char *comp_dn, const u_char *eom ));
#define getshort _getshort
static struct hostent *getanswer_ARGS(( union querybuf *answer, int anslen,
    int iquery ));
```

```

#define bcmp memcmp
#define bcopy(s, d, l) memcpy(d, s, l)
#endif /* _MINIX */

static struct hostent host;
static char *host_aliases[MAXALIASES];
static char hostbuf[BUFSIZ+1];
static struct in_addr host_addr;

#ifdef _MINIX
char *strpbrk();
#endif /* !_MINIX */

#if PACKETSZ > 1024
#define MAXPACKET    PACKETSZ
#else
#define MAXPACKET    1024
#endif

typedef union querybuf
{
    dns_hdr_t hdr;
    u_char buf[MAXPACKET];
} querybuf_t;

typedef union align {
    long al;
    char ac;
} align_t;

static struct hostent *
getanswer(answer, anslen, iquery)
    querybuf_t *answer;
    int anslen;
    int iquery;
{
    register dns_hdr_t *hp;
    register u_char *cp;
    register int n;
    u_char *eom;
    char *bp, **ap;
    int type, class, buflen, ancount, qdcount;
    int haveanswer, getclass = C_ANY;
    char **hap;

    eom = answer->buf + anslen;
    /*
     * find first satisfactory answer
     */
    hp = &answer->hdr;
    ancount = ntohs(hp->dh_ancount);
    qdcount = ntohs(hp->dh_qdcount);
    bp = hostbuf;
    buflen = sizeof(hostbuf);
    cp = answer->buf + sizeof(dns_hdr_t);
    if (qdcount) {
        if (iquery) {
            if ((n = dn_expand((u_char *)answer->buf, eom,
                cp, (u_char *)bp, buflen)) < 0) {
                h_errno = NO_RECOVERY;
                return ((struct hostent *) NULL);
            }
            cp += n + QFIXEDSZ;
            host.h_name = bp;
            n = strlen(bp) + 1;
            bp += n;
            buflen -= n;
        } else
            cp += dn_skipname(cp, eom) + QFIXEDSZ;
    }
}

```

```

        while (--qdcnt > 0)
            cp += dn_skipname(cp, eom) + QFIXEDSZ;
    } else if (iquery) {
        if (hp->dh_flag1 & DHF_AA)
            h_errno = HOST_NOT_FOUND;
        else
            h_errno = TRY_AGAIN;
        return ((struct hostent *) NULL);
    }
    ap = host_aliases;
    *ap = NULL;
    host.h_aliases = host_aliases;
    hap = h_addr_ptrs;
    *hap = NULL;
#if BSD >= 43 || defined(h_addr) /* new-style hostent structure */
    host.h_addr_list = h_addr_ptrs;
#endif
    haveanswer = 0;
    while (--ancount >= 0 && cp < eom) {
        if ((n = dn_expand((u_char *)answer->buf, eom, cp, (u_char *)bp,
            buflen)) < 0)
            break;
        cp += n;
        type = getshort(cp);
        cp += sizeof(u_short);
        class = getshort(cp);
        cp += sizeof(u_short) + sizeof(u_long);
        n = getshort(cp);
        cp += sizeof(u_short);
        if (type == T_CNAME) {
            cp += n;
            if (ap >= &host_aliases[MAXALIASES-1])
                continue;
            *ap++ = bp;
            n = strlen(bp) + 1;
            bp += n;
            buflen -= n;
            continue;
        }
        if (iquery && type == T_PTR) {
            if ((n = dn_expand((u8_t *)answer->buf, eom,
                cp, (u8_t *)bp, buflen)) < 0) {
                cp += n;
                continue;
            }
            cp += n;
            host.h_name = bp;
            return(&host);
        }
        if (iquery || type != T_A) {
#ifdef DEBUG
            if (_res.options & RES_DEBUG)
                printf("unexpected answer type %d, size %d\n",
                    type, n);
#endif
            cp += n;
            continue;
        }
        if (haveanswer) {
            if (n != host.h_length) {
                cp += n;
                continue;
            }
            if (class != getclass) {
                cp += n;
                continue;
            }
        } else {
            host.h_length = n;

```

```

        getclass = class;
        host.h_addrtype = (class == C_IN) ? AF_INET : AF_UNSPEC;
        if (!iquery) {
            host.h_name = bp;
            bp += strlen(bp) + 1;
        }
    }

    bp += (size_t)(sizeof(align_t) -
                  ((u_long)bp % sizeof(align_t)));

#ifdef DEBUG
    if (bp + n >= &hostbuf[sizeof(hostbuf)]) {
        if (_res.options & RES_DEBUG)
            printf("size (%d) too big\n", n);
    }
#endif

    break;
}
bcopy(cp, *hap++ = bp, n);
bp += n;
cp += n;
haveanswer++;
}
if (haveanswer) {
    *ap = NULL;
#if BSD >= 43 || defined(h_addr) /* new-style hostent structure */
    *hap = NULL;
#else
    host.h_addr = h_addr_ptrs[0];
#endif
    return (&host);
} else {
    h_errno = TRY_AGAIN;
    return ((struct hostent *) NULL);
}
}

struct hostent *
gethostbyname(name)
    char *name;
{
    querybuf_t buf;
    register char *cp;
    int n;

    /*
     * disallow names consisting only of digits/dots, unless
     * they end in a dot.
     */
    if (isdigit(name[0]))
        for (cp = name; ++cp) {
            if (!*cp) {
                if (*--cp == '.')
                    break;
            }
            /*
             * All-numeric, no dot at the end.
             * Fake up a hostent as if we'd actually
             * done a lookup. What if someone types
             * 255.255.255.255? The test below will
             * succeed spuriously... ???
             */
            if ((host_addr.s_addr = inet_addr(name)) == -1) {
                h_errno = HOST_NOT_FOUND;
                return((struct hostent *) NULL);
            }
            host.h_name = name;
            host.h_aliases = host_aliases;
            host_aliases[0] = NULL;
            host.h_addrtype = AF_INET;

```

```

        host.h_length = sizeof(u_long);
        h_addr_ptrs[0] = (char *)&host_addr;
        h_addr_ptrs[1] = (char *)0;
#if BSD >= 43 || defined(h_addr)    /* new-style hostent structure */
        host.h_addr_list = h_addr_ptrs;
#else
        host.h_addr = h_addr_ptrs[0];
#endif
    }
    return (&host);
}
if (!isdigit(*cp) && *cp != '.')
    break;
}

    if ((n = res_search(name, C_IN, T_A, buf.buf, sizeof(buf))) < 0) {
#ifdef DEBUG
        if (_res.options & RES_DEBUG)
            printf("res_search failed\n");
#endif
        return ((struct hostent *) NULL);
    }
    return (getanswer(&buf, n, 0));
}

struct hostent *
gethostbyaddr(addr, len, type)
    const char *addr;
    int len, type;
{
    int n;
    querybuf_t buf;
    register struct hostent *hp;
    char qbuf[MAXDNAME];

    if (type != AF_INET)
        return ((struct hostent *) NULL);
    (void)sprintf(qbuf, "%u.%u.%u.%u.in-addr.arpa",
        ((unsigned)addr[3] & 0xff),
        ((unsigned)addr[2] & 0xff),
        ((unsigned)addr[1] & 0xff),
        ((unsigned)addr[0] & 0xff));
    n = res_query(qbuf, C_IN, T_PTR, (u8_t *)&buf, sizeof(buf));
    if (n < 0) {
#ifdef DEBUG
        if (_res.options & RES_DEBUG)
            printf("res_query failed\n");
#endif
        return ((struct hostent *) NULL);
    }
    hp = getanswer(&buf, n, 1);
    if (hp == NULL)
        return ((struct hostent *) NULL);
    hp->h_addrtype = type;
    hp->h_length = len;
    h_addr_ptrs[0] = (char *)&host_addr;
    h_addr_ptrs[1] = (char *)0;
    host_addr = *(struct in_addr *)addr;
#if BSD < 43 && !defined(h_addr)    /* new-style hostent structure */
    hp->h_addr = h_addr_ptrs[0];
#endif
    return(hp);
}

```

Source: Andrew S. Tannenbaum & Albert S. Woodhull, "Operating Systems Design and Implementation, Second Edition," Prentice Hall, 1987.