



Linux Ubuntu 26.04 Manual Pages on command 'getnameinfo.3'

\$ man getnameinfo.3

getnameinfo(3) Library Functions Manual getnameinfo(3)

NAME

getnameinfo - address-to-name translation in protocol-independent manner

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <sys/socket.h>

#include <netdb.h>

int getnameinfo(socklen_t hostlen, socklen_t servlen;

               const struct sockaddr *restrict addr, socklen_t addrlen,

               char host[_Nullable restrict hostlen],

               socklen_t hostlen,

               char serv[_Nullable restrict servlen],

               socklen_t servlen,

               int flags);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

getnameinfo():

Since glibc 2.22:

 _POSIX_C_SOURCE >= 200112L

glibc 2.21 and earlier:

 _POSIX_C_SOURCE

DESCRIPTION

The getnameinfo() function is the inverse of getaddrinfo(3): it converts a socket address to

a corresponding host and service, in a protocol-independent manner. It combines the functionality of `gethostbyaddr(3)` and `getservbyport(3)`, but unlike those functions, `getnameinfo()` is reentrant and allows programs to eliminate IPv4-versus-IPv6 dependencies.

The `addr` argument is a pointer to a generic socket address structure (of type `sockaddr_in` or `sockaddr_in6`) of size `addrlen` that holds the input IP address and port number. The arguments `host` and `serv` are pointers to caller-allocated buffers (of size `hostlen` and `servlen` respectively) into which `getnameinfo()` places null-terminated strings containing the host and service names respectively.

The caller can specify that no hostname (or no service name) is required by providing a `NULL` `host` (or `serv`) argument or a zero `hostlen` (or `servlen`) argument. However, at least one of hostname or service name must be requested.

The `flags` argument modifies the behavior of `getnameinfo()` as follows:

`NI_NAMEREQD`

If set, then an error is returned if the hostname cannot be determined.

`NI_DGRAM`

If set, then the service is datagram (UDP) based rather than stream (TCP) based.

This is required for the few ports (512-514) that have different services for UDP and TCP.

`NI_NOFQDN`

If set, return only the hostname part of the fully qualified domain name for local hosts.

`NI_NUMERICHOST`

If set, then the numeric form of the hostname is returned. (When not set, this will still happen in case the node's name cannot be determined.)

`NI_NUMERICSERV`

If set, then the numeric form of the service address is returned. (When not set, this will still happen in case the service's name cannot be determined.)

Extensions to `getnameinfo()` for Internationalized Domain Names

Starting with glibc 2.3.4, `getnameinfo()` has been extended to selectively allow hostnames to be transparently converted to and from the Internationalized Domain Name (IDN) format (see RFC 3490, Internationalizing Domain Names in Applications (IDNA)). Three new flags are defined:

`NI_IDN` If this flag is used, then the name found in the lookup process is converted from IDN

format to the locale's encoding if necessary. ASCII-only names are not affected by the conversion, which makes this flag usable in existing programs and environments.

NI_IDN_ALLOW_UNASSIGNED

NI_IDN_USE_STD3_ASCII_RULES

Setting these flags will enable the IDNA_ALLOW_UNASSIGNED (allow unassigned Unicode code points) and IDNA_USE_STD3_ASCII_RULES (check output to make sure it is a STD3 conforming hostname) flags respectively to be used in the IDNA handling.

RETURN VALUE

On success, 0 is returned, and node and service names, if requested, are filled with null-terminated strings, possibly truncated to fit the specified buffer lengths. On error, one of the following nonzero error codes is returned:

EAI_AGAIN

The name could not be resolved at this time. Try again later.

EAI_BADFLAGS

The flags argument has an invalid value.

EAI_FAIL

A nonrecoverable error occurred.

EAI_FAMILY

The address family was not recognized, or the address length was invalid for the specified family.

EAI_MEMORY

Out of memory.

EAI_NONAME

The name does not resolve for the supplied arguments. NI_NAMEREQD is set and the host's name cannot be located, or neither hostname nor service name were requested.

EAI_OVERFLOW

The buffer pointed to by host or serv was too small.

EAI_SYSTEM

A system error occurred. The error code can be found in errno.

The gai_strerror(3) function translates these error codes to a human readable string, suitable for error reporting.

FILES

/etc/hosts

```
/etc/resolv.conf
```

For an explanation of the terms used in this section, see `attributes(7)`.

[illegible]

Interface	Attribute	Value	
?	?	?	?

[illegible]

? getnameinfo() ? Thread safety ? MT-Safe env locale ?

[illegible]

POSIX.1-2008. RFC 2553.

glibc 2.1. POSIX.1-2001.

Before glibc 2.2, the `hostlen` and `servlen` arguments were typed as `size_t`.

In order to assist the programmer in choosing reasonable sizes for the supplied buffers,

<netdb.h> defines the constants

```
#define NI_MAXHOST 1025
```

```
#define NI_MAXSERV    32
```

Since glibc 2.8, these definitions are exposed only if suitable feature test macros are de?

defined, namely: `_GNU_SOURCE`, `_DEFAULT_SOURCE` (since glibc 2.19), or (in glibc versions up to and including 2.19) `_BSD_SOURCE` or `_SVID_SOURCE`.

The former is the constant MAXDNAME in recent versions of BIND's <arpa/nameser.h> header file. The latter is a guess based on the services listed in the current Assigned Numbers RFC.

The following code tries to get the numeric hostname and service name, for a given socket address. Note that there is no hardcoded reference to a particular address family.

```
struct sockaddr *addr;    /* input */
```

```
socklen_t addrlen;    /* input */
```

```
char hbuf[NI_MAXHOST], sbuf[NI_MAXSERV];
```

```
if (getnameinfo(addr, addrlen, hbuf, sizeof(hbuf), sbuf,
```

```
sizeof(sbuf), NI_NUMERICHOST | NI_NUMERICSERV) == 0)
```

```
printf("host=%s, serv=%s\n", hbuf, sbuf);
```

The following version checks if the socket address has a reverse address mapping.

```
struct sockaddr *addr;    /* input */  
  
socklen_t addrlen;       /* input */  
  
char hbuf[NI_MAXHOST];  
  
if (getnameinfo(addr, addrlen, hbuf, sizeof(hbuf),  
                NULL, 0, NI_NAMEREQD))  
    printf("could not resolve hostname");  
  
else  
    printf("host=%s\n", hbuf);
```

An example program using `getnameinfo()` can be found in `getaddrinfo(3)`.

SEE ALSO

`accept(2)`, `getpeername(2)`, `getsockname(2)`, `recvfrom(2)`, `socket(2)`, `getaddrinfo(3)`, `gethost?`
`byaddr(3)`, `getservbyname(3)`, `getservbyport(3)`, `inet_ntop(3)`, `hosts(5)`, `services(5)`, `host?`
`name(7)`, `named(8)`

R. Gilligan, S. Thomson, J. Bound and W. Stevens, Basic Socket Interface Extensions for IPv6, RFC 2553, March 1999.

Tatsuya Jinmei and Atsushi Onoe, An Extension of Format for IPv6 Scoped Addresses, internet draft, work in progress <ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipngwg-scopedaddr-for?mat-02.txt>.

Craig Metz, Protocol Independence Using the Sockets API, Proceedings of the freenix track: 2000 USENIX annual technical conference, June 2000 <http://www.usenix.org/publications/library/proceedings/usenix2000/freenix/metzprotocol.html>.