NAME

getnameinfo - socket address structure to hostname and service name

SYNOPSIS

#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int
getnameinfo(*const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen, char *serv, size_t servlen, int flags*);

DESCRIPTION

The **getnameinfo**() function is used to convert a sockaddr structure to a pair of host name and service strings. It is a replacement for and provides more flexibility than the gethostbyaddr(3) and getservbyport(3) functions and is the converse of the getaddrinfo(3) function.

If a link-layer address or UNIX-domain address is passed to **getnameinfo**(), its ASCII representation will be stored in *host*. The string pointed to by *serv* will be set to the empty string if non-NULL; *flags* will always be ignored. For a link-layer address, this can be used as a replacement of the legacy link_ntoa(3) function.

The sockaddr structure *sa* should point to either a sockaddr_in, sockaddr_in6, sockaddr_dl, or sockaddr_un structure (for IPv4, IPv6, link-layer, or UNIX-domain respectively) that is *salen* bytes long. If *salen* is shorter than the length corresponding to the specified address family or longer than **sizeof**(*struct sockaddr_storage*), it returns EAI_FAMILY. Note that *sa->sa_len* should be consistent with *salen* though the value of *sa->sa_len* is not directly used in this function.

The host and service names associated with *sa* are stored in *host* and *serv* which have length parameters *hostlen* and *servlen*. The maximum value for *hostlen* is NI_MAXHOST and the maximum value for *servlen* is NI_MAXSERV, as defined by <*netdb.h*>. If a length parameter is zero, no string will be stored. Otherwise, enough space must be provided to store the host name or service string plus a byte for the NUL terminator.

The *flags* argument is formed by OR'ing the following values:

NI_NOFQDN A fully qualified domain name is not required for local hosts. The local part of the fully qualified domain name is returned instead.

NI_NUMERICHOST Return the address in numeric form, as if calling inet_ntop(3), instead of a

host name.

NI_NAMEREQD — A name is required. If the host name cannot be found in DNS and this flag is set, a non-zero error code is returned. If the host name is not found and the flag is not set, the address is returned in numeric form.

NI_NUMERICSERV — The service name is returned as a digit string representing the port number.

NI_NUMERICSCOPE — The scope identifier is returned as a digit string.

NI_DGRAM — Specifies that the service being looked up is a datagram service, and causes getservbyport(3) to be called with a second argument of "udp" instead of its default of "tcp". This is required for the few ports (512-514) that have different services for UDP and TCP.

This implementation allows numeric IPv6 address notation with scope identifier, as documented in chapter 11 of RFC 4007. IPv6 link-local address will appear as a string like "fe80::1%ne0". Refer to getaddrinfo(3) for more information.

RETURN VALUES

getnameinfo() returns zero on success or one of the error codes listed in gai_strerror(3) if an error occurs.

EXAMPLES

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

```
struct sockaddr *sa;/* input */
char hbuf[NI_MAXHOST], sbuf[NI_MAXSERV];

if (getnameinfo(sa, sa->sa_len, hbuf, sizeof(hbuf), sbuf,
   sizeof(sbuf), NI_NUMERICHOST | NI_NUMERICSERV)) {
        errx(1, "could not get numeric hostname");
        /* NOTREACHED */
}
printf("host=%s, serv=%s\n", hbuf, sbuf);
```

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

```
struct sockaddr *sa;/* input */
```

```
char hbuf[NI_MAXHOST];

if (getnameinfo(sa, sa->sa_len, hbuf, sizeof(hbuf), NULL, 0,
   NI_NAMEREQD)) {
        errx(1, "could not resolve hostname");
        /* NOTREACHED */
}
printf("host=%s\n", hbuf);
```

SEE ALSO

gai_strerror(3), getaddrinfo(3), gethostbyaddr(3), getservbyport(3), inet_ntop(3), link_ntoa(3), resolver(3), inet(4), inet6(4), unix(4), hosts(5), resolv.conf(5), services(5), hostname(7)

R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens, *Basic Socket Interface Extensions for IPv6*, RFC 3493, February 2003.

S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill, *IPv6 Scoped Address Architecture*, RFC 4007, March 2005.

Craig Metz, "Protocol Independence Using the Sockets API", *Proceedings of the freenix track: 2000 USENIX annual technical conference*, June 2000.

STANDARDS

The **getnameinfo**() function is defined by the IEEE Std 1003.1-2004 ("POSIX.1") specification and documented in RFC 3493, "Basic Socket Interface Extensions for IPv6".

CAVEATS

getnameinfo() can return both numeric and FQDN forms of the address specified in *sa*. There is no return value that indicates whether the string returned in *host* is a result of binary to numeric-text translation (like inet_ntop(3)), or is the result of a DNS reverse lookup. Because of this, malicious parties could set up a PTR record as follows:

```
1.0.0.127.in-addr.arpa. IN PTR  10.1.1.1
```

and trick the caller of **getnameinfo**() into believing that *sa* is 10.1.1.1 when it is actually 127.0.0.1.

To prevent such attacks, the use of NI_NAMEREQD is recommended when the result of **getnameinfo**() is used for access control purposes:

```
struct sockaddr *sa;
```

```
socklen_t salen;
char addr[NI_MAXHOST];
struct addrinfo hints, *res;
int error;

error = getnameinfo(sa, salen, addr, sizeof(addr),
   NULL, 0, NI_NAMEREQD);
if (error == 0) {
        memset(&hints, 0, sizeof(hints));
        hints.ai_socktype = SOCK_DGRAM;   /*dummy*/
        hints.ai_flags = AI_NUMERICHOST;
        if (getaddrinfo(addr, "0", &hints, &res) == 0) {
                /* malicious PTR record */
                freeaddrinfo(res);
                printf("bogus PTR record\n");
                return -1;
        }
        /* addr is FQDN as a result of PTR lookup */
} else {
        /* addr is numeric string */
        error = getnameinfo(sa, salen, addr, sizeof(addr),
           NULL, 0, NI_NUMERICHOST);
}
```