Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'freeaddrinfo.3p' command

$ man freeaddrinfo.3p

FREEADDRINFO(3P) POSIX Programmer's Manual FREEADDRINFO(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

freeaddrinfo, getaddrinfo ? get address information

SYNOPSIS

```
#include <sys/socket.h>
#include <netdb.h>
void freeaddrinfo(struct addrinfo *ai);
int getaddrinfo(const char *restrict nodename,
    const char *restrict servname,
    const struct addrinfo *restrict hints,
    struct addrinfo **restrict res);
```

DESCRIPTION

The freeaddrinfo() function shall free one or more addrinfo structures returned by getaddrinfo(), along with any additional storage associated with those structures. If the ai_next field of the structure is not null, the entire list of structures shall be freed. The freeaddrinfo() function shall support the freeing of arbitrary sublists of an addrinfo list originally returned by getaddrinfo().

The getaddrinfo() function shall translate the name of a service loca?
tion (for example, a host name) and/or a service name and shall return
a set of socket addresses and associated information to be used in cre?
ating a socket with which to address the specified service.

Note: In many cases it is implemented by the Domain Name System, as documented in RFC 1034, RFC 1035, and RFC 1886.

The freeaddrinfo() and getaddrinfo() functions shall be thread-safe.

The nodename and servname arguments are either null pointers or point?
ers to null-terminated strings. One or both of these two arguments
shall be supplied by the application as a non-null pointer.

The format of a valid name depends on the address family or families.
If a specific family is not given and the name could be interpreted as
valid within multiple supported families, the implementation shall at?
tempt to resolve the name in all supported families and, in absence of
errors, one or more results shall be returned.

If the nodename argument is not null, it can be a descriptive name or
can be an address string. If the specified address family is AF_INET,
AF_INET6, or AF_UNSPEC, valid descriptive names include host names. If
the specified address family is AF_INET or AF_UNSPEC, address strings
using Internet standard dot notation as specified in inet_addr() are
valid.

If the specified address family is AF_INET6 or AF_UNSPEC, standard IPv6
text forms described in inet_ntop() are valid.

If nodename is not null, the requested service location is named by
nodename; otherwise, the requested service location is local to the
caller.

If servname is null, the call shall return network-level addresses for
the specified nodename. If servname is not null, it is a null-termi?
nated character string identifying the requested service. This can be
either a descriptive name or a numeric representation suitable for use
with the address family or families. If the specified address family
is AF_INET, AF_INET6, or AF_UNSPEC, the service can be specified as a
string specifying a decimal port number.

If the hints argument is not null, it refers to a structure containing input values that directs the operation by providing options and by limiting the returned information to a specific socket type, address family, and/or protocol, as described below. The application shall ensure that each of the ai_addrlen, ai_addr, ai_canonname, and ai_next members, as well as each of the non-standard additional members, if any, of this hints structure is initialized. If any of these members has a value other than the value that would result from default initialization, the behavior is implementation-defined. A value of AF_UNSPEC for ai_family means that the caller shall accept any address family. A value of zero for ai_socktype means that the caller shall accept any socket type. A value of zero for ai_protocol means that the caller shall accept any protocol. If hints is a null pointer, the behavior shall be as if it referred to a structure containing the value zero for the ai_flags, ai_socktype, and ai_protocol fields, and AF_UNSPEC for the ai_family field.

The ai_flags field to which the hints parameter points shall be set to zero or be the bitwise-inclusive OR of one or more of the values AI_PASSIVE, AI_CANONNAME, AI_NUMERICHOST, AI_NUMERICSERV, AI_V4MAPPED, AI_ALL, and AI_ADDRCONFIG.

If the AI_PASSIVE flag is specified, the returned address information shall be suitable for use in binding a socket for accepting incoming connections for the specified service. In this case, if the nodename argument is null, then the IP address portion of the socket address structure shall be set to INADDR_ANY for an IPv4 address or IN6ADDR_ANY_INIT for an IPv6 address. If the AI_PASSIVE flag is not specified, the returned address information shall be suitable for a call to connect() (for a connection-mode protocol) or for a call to connect(), sendto(), or sendmsg() (for a connectionless protocol). In this case, if the nodename argument is null, then the IP address portion of the socket address structure shall be set to the loopback address. The AI_PASSIVE flag shall be ignored if the nodename argument is not null.

If the AI_CANONNAME flag is specified and the nodename argument is not null, the function shall attempt to determine the canonical name corre? sponding to nodename (for example, if nodename is an alias or shorthand notation for a complete name).

Note: Since different implementations use different conceptual mod? els, the terms ``canonical name'' and ``alias'' cannot be precisely defined for the general case. However, Domain Name System implementations are expected to interpret them as they are used in RFC 1034.

A numeric host address string is not a ``name'', and thus does not have a ``canonical name'' form; no address to host name translation is performed. See below for handling of the case where a canonical name cannot be obtained.

If the AI_NUMERICHOST flag is specified, then a non-null nodename string supplied shall be a numeric host address string. Otherwise, an [EAI_NONAME] error is returned. This flag shall prevent any type of name resolution service (for example, the DNS) from being invoked.

If the AI_NUMERICSERV flag is specified, then a non-null servname string supplied shall be a numeric port string. Otherwise, an [EAI_NO? NAME] error shall be returned. This flag shall prevent any type of name resolution service (for example, NIS+) from being invoked.

By default, with an ai_family of AF_INET6, getaddrinfo() shall return only IPv6 addresses. If the AI_V4MAPPED flag is specified along with an ai_family of AF_INET6, then getaddrinfo() shall return IPv4-mapped IPv6 addresses on finding no matching IPv6 addresses. The AI_V4MAPPED flag shall be ignored unless ai_family equals AF_INET6. If the AI_ALL flag is used with the AI_V4MAPPED flag, then getaddrinfo() shall return all matching IPv6 and IPv4 addresses. The AI_ALL flag without the AI_V4MAPPED flag shall be ignored.

If the AI_ADDRCONFIG flag is specified, IPv4 addresses shall be re? turned only if an IPv4 address is configured on the local system, and IPv6 addresses shall be returned only if an IPv6 address is configured on the local system.

The ai_socktype field to which argument hints points specifies the socket type for the service, as defined in socket(). If a specific socket type is not given (for example, a value of zero) and the service name could be interpreted as valid with multiple supported socket types, the implementation shall attempt to resolve the service name for all supported socket types and, in the absence of errors, all possible results shall be returned. A non-zero socket type value shall limit the returned information to values with the specified socket type.

If the ai_family field to which hints points has the value AF_UNSPEC, addresses shall be returned for use with any address family that can be used with the specified nodename and/or servname. Otherwise, addresses shall be returned for use only with the specified address family. If ai_family is not AF_UNSPEC and ai_protocol is not zero, then addresses shall be returned for use only with the specified address family and protocol; the value of ai_protocol shall be interpreted as in a call to the socket() function with the corresponding values of ai_family and ai_protocol.

RETURN VALUE

A zero return value for getaddrinfo() indicates successful completion; a non-zero return value indicates failure. The possible values for the failures are listed in the ERRORS section.

Upon successful return of getaddrinfo(), the location to which res points shall refer to a linked list of addrinfo structures, each of which shall specify a socket address and information for use in creat? ing a socket with which to use that socket address. The list shall in? clude at least one addrinfo structure. The ai_next field of each struc? ture contains a pointer to the next structure on the list, or a null pointer if it is the last structure on the list. Each structure on the list shall include values for use with a call to the socket() function, and a socket address for use with the connect() function or, if the AI_PASSIVE flag was specified, for use with the bind() function. The fields ai_family, ai_socktype, and ai_protocol shall be usable as the arguments to the socket() function to create a socket suitable for use

with the returned address. The fields ai_addr and ai_addrlen are usable as the arguments to the connect() or bind() functions with such a socket, according to the AI_PASSIVE flag.

If nodename is not null, and if requested by the AI_CANONNAME flag, the ai_canonname field of the first returned addrinfo structure shall point to a null-terminated string containing the canonical name corresponding to the input nodename; if the canonical name is not available, then ai_canonname shall refer to the nodename argument or a string with the same contents. The contents of the ai_flags field of the returned structures are undefined.

All fields in socket address structures returned by getaddrinfo() that are not filled in through an explicit argument (for example, sin6_flow? info) shall be set to zero.

Note: This makes it easier to compare socket address structures.

ERRORS

The getaddrinfo() function shall fail and return the corresponding er? ror value if:

[EAI_AGAIN] The name could not be resolved at this time. Future at? tempts may succeed.

[EAI_BADFLAGS]

The flags parameter had an invalid value.

[EAI_FAIL] A non-recoverable error occurred when attempting to resolve the name.

[EAI_FAMILY]

The address family was not recognized.

[EAI_MEMORY]

There was a memory allocation failure when trying to allo? cate storage for the return value.

[EAI_NONAME]

The name does not resolve for the supplied parameters. Neither nodename nor servname were supplied. At least one of these shall be supplied.

[EAI_SERVICE]

The service passed was not recognized for the specified socket type.

[EAI_SOCKTYPE]

The intended socket type was not recognized.

[EAI_SYSTEM]

A system error occurred; the error code can be found in er? rno.

The following sections are informative.

EXAMPLES

The following (incomplete) program demonstrates the use of getad? drinfo() to obtain the socket address structure(s) for the service named in the program's command-line argument. The program then loops through each of the address structures attempting to create and bind a socket to the address, until it performs a successful bind().

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netdb.h>
int
main(int argc, char *argv[])
{
    struct addrinfo *result, *rp;
    int sfd, s;
    if (argc != 2) {
        fprintf(stderr, "Usage: %s port\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    struct addrinfo hints = {0};
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_DGRAM;
    hints.ai_flags = AI_PASSIVE;
```

```
        hints.ai_protocol = 0;
        s = getaddrinfo(NULL, argv[1], &hints, &result);
        if (s != 0) {
            fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
            exit(EXIT_FAILURE);
        }
        /* getaddrinfo() returns a list of address structures.
           Try each address until a successful bind().
           If socket(2) (or bind(2)) fails, close the socket
           and try the next address. */
        for (rp = result; rp != NULL; rp = rp->ai_next) {
            sfd = socket(rp->ai_family, rp->ai_socktype,
                  rp->ai_protocol);
            if (sfd == -1)
                continue;
            if (bind(sfd, rp->ai_addr, rp->ai_addrlen) == 0)
                break;           /* Success */
            close(sfd);
        }
        if (rp == NULL) {        /* No address succeeded */
            fprintf(stderr, "Could not bind\n");
            exit(EXIT_FAILURE);
        }
        freeaddrinfo(result);    /* No longer needed */
               /* ... use socket bound to sfd ... */
    }
```

APPLICATION USAGE

If the caller handles only TCP and not UDP, for example, then the ai_protocol member of the hints structure should be set to IPPROTO_TCP when getaddrinfo() is called.

If the caller handles only IPv4 and not IPv6, then the ai_family member of the hints structure should be set to AF_INET when getaddrinfo() is called.

Although it is common practice to initialize the hints structure using:

```
struct addrinfo hints;
memset(&hints, 0, sizeof hints);
```

this method is not portable according to this standard, because the structure can contain pointer or floating-point members that are not required to have an all-bits-zero representation after default initial?ization. Portable methods make use of default initialization; for exam?ple:

```
struct addrinfo hints = { 0 };
```

or:

```
static struct addrinfo hints_init;
struct addrinfo hints = hints_init;
```

A future version of this standard may require that a pointer object with an all-bits-zero representation is a null pointer, and that ad?drinfo does not have any floating-point members if a floating-point ob?ject with an all-bits-zero representation does not have the value 0.0.

The term ``canonical name'' is misleading; it is taken from the Domain Name System (RFC 2181). It should be noted that the canonical name is a result of alias processing, and not necessarily a unique attribute of a host, address, or set of addresses. See RFC 2181 for more discussion of this in the Domain Name System context.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

connect(), endservent(), gai_strerror(), getnameinfo(), socket()

The Base Definitions volume of POSIX.1?2017, <netdb.h>, <sys_socket.h>

COPYRIGHT

Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker? nel.org/doc/man-pages/reporting_bugs.html .