

NAME

Socket6 – IPv6 related part of the C socket.h defines and structure manipulators

SYNOPSIS

```
use Socket;
use Socket6;

@res = getaddrinfo('hishost.com', 'daytime', AF_UNSPEC, SOCK_STREAM);
$family = -1;
while (scalar(@res) >= 5) {
    ($family, $socktype, $proto, $saddr, $scanonname, @res) = @res;

    ($host, $port) = getnameinfo($saddr, NI_NUMERICHOST | NI_NUMERICSERV);
    print STDERR "Trying to connect to $host port $port...\n";

    socket(Socket_Handle, $family, $socktype, $proto) || next;
    connect(Socket_Handle, $saddr) && last;

    close(Socket_Handle);
    $family = -1;
}

if ($family != -1) {
    print STDERR "connected to $host port $port\n";
} else {
    die "connect attempt failed\n";
}
```

DESCRIPTION

This module provides glue routines to the various IPv6 functions.

If you use the Socket6 module, be sure to specify “use Socket” as well as “use Socket6”.

Functions supplied are:

inet_pton FAMILY, TEXT_ADDRESS

This function takes an IP address in presentation (or string) format and converts it into numeric (or binary) format. The type of IP address conversion (IPv4 versus IPv6) is controlled by the FAMILY argument.

inet_ntop FAMILY, BINARY_ADDRESS

This function takes an IP address in numeric (or binary) format and converts it into presentation (or string) format. The type of IP address conversion (IPv4 versus IPv6) is controlled by the FAMILY argument.

pack_sockaddr_in6 PORT, ADDR

This function takes two arguments: a port number, and a 16-octet IPv6 address structure (as returned by `inet_pton()`). It returns the `sockaddr_in6` structure with these arguments packed into their correct fields, as well as the `AF_INET6` family. The other fields are not set and their values should not be relied upon.

pack_sockaddr_in6_all PORT, FLOWINFO, ADDR, SCOPEID

This function takes four arguments: a port number, a 16-octet IPv6 address structure (as returned by `inet_pton`), any special flow information, and any specific scope information. It returns a complete `sockaddr_in6` structure with these arguments packed into their correct fields, as well as the `AF_INET6` family.

`unpack_sockaddr_in6 NAME`

This function takes a `sockaddr_in6` structure (as returned by `pack_sockaddr_in6()`) and returns a list of two elements: the port number and the 16-octet IP address. This function will croak if it determines it has not been passed an IPv6 structure.

`unpack_sockaddr_in6_all NAME`

This function takes a `sockaddr_in6` structure (as returned by `pack_sockaddr_in6()`) and returns a list of four elements: the port number, the flow information, the 16-octet IP address, and the scope information. This function will croak if it determines it has not been passed an IPv6 structure.

`gethostbyname2 HOSTNAME, FAMILY`

`getaddrinfo NODENAME, SERVICENAME, [FAMILY, SOCKTYPE, PROTOCOL, FLAGS]`

This function converts node names to addresses and service names to port numbers.

If the `NODENAME` argument is not a false value, then a nodename to address lookup is performed; otherwise a service name to port number lookup is performed. At least one of `NODENAME` and `SERVICENAME` must have a true value.

If the lookup is successful, a list consisting of multiples of five elements is returned.

Each group of five elements consists of the address family, socket type, protocol, 16-octet IP address, and the canonical name (undef if the node name passed is already the canonical name).

The arguments `FAMILY`, `SOCKTYPE`, `PROTOCOL`, and `FLAGS` are all optional.

This function will croak if it determines it has not been passed an IPv6 structure.

If the lookup is unsuccessful, the function returns a single scalar. This will contain the string version of that error in string context, and the numeric value in numeric context.

`getnameinfo NAME, [FLAGS]`

This function takes a socket address structure. If successful, it returns two strings containing the node name and service name.

The optional `FLAGS` argument controls what kind of lookup is performed.

If the lookup is unsuccessful, the function returns a single scalar. This will contain the string version of that error in string context, and the numeric value in numeric context.

getipnodebyname HOST, [FAMILY, FLAGS]

This function takes either a node name or an IP address string and performs a lookup on that name (or conversion of the string). It returns a list of five elements: the canonical host name, the address family, the length in octets of the IP addresses returned, a reference to a list of IP address structures, and a reference to a list of aliases for the host name.

The arguments FAMILY and FLAGS are optional.

Note: This function does not handle IPv6 scope identifiers, and should be used with care.

And, this function was deprecated in RFC3493.

The getnameinfo function should be used instead.

getipnodebyaddr FAMILY, ADDRESS

This function takes an IP address family and an IP address structure and performs a reverse lookup on that address.

It returns a list of five elements: the canonical host name, the address family, the length in octets of the IP addresses returned, a reference to a list of IP address structures, and a reference to a list of aliases for the host name.

Note: This function does not handle IPv6 scope identifiers, and should be used with care.

And, this function was deprecated in RFC3493.

The getaddrinfo function should be used instead.

gai_strerror ERROR_NUMBER

This function returns a string corresponding to the error number passed in as an argument.

in6addr_any

This function returns the 16-octet wildcard address.

in6addr_loopback

This function returns the 16-octet loopback address.