Networking, Part 2: Using getaddrinfo

Thomas Liu edited this page on Jan 25 · 1 revision

How do I use `getaddrinfo` to convert the hostname into an IP address?

The function `getaddrinfo` can convert a human readable domain name (e.g. `www.illinois.edu`) into an IPv4 and IPv6 address. In fact it will return a linked-list of addrinfo structs:

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

It's very easy to use. For example, suppose you wanted to find out the numeric IPv4 address of a webserver at www.bbc.com. We do this in two stages. First use getaddrinfo to build a linked-list of possible connections. Secondly use `getnameinfo` to convert the binary address into a readable form.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

struct addrinfo hints, *infoptr; // So no need to use memset global variables

int main() {
  hints.ai_family = AF_INET; // AF_INET means IPv4 only addresses

  int result = getaddrinfo("www.bbc.com", NULL, &hints, &infoptr);
  if (result) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(result));
    exit(1);
  }

  struct addrinfo *p;
  char host[256],service[256];

  for(p = infoptr; p != NULL; p = p->ai_next) {

    getnameinfo(p->ai_addr, p->ai_addrlen, host, sizeof(host), service, sizeof(se
```

```
    puts(host);
  }

  freeaddrinfo(infoptr);
  return 0;
}
```

Typical output:

```
212.58.244.70
212.58.244.71
```

How is www.cs.illinois.edu converted into an IP address?

Magic! No seriously, a system called "DNS" (Domain Name Service) is used. If a machine does not hold the answer locally then it sends a UDP packet to a local DNS server. This server in turn may query other upstream DNS servers.

Is DNS secure?

DNS by itself is fast but not secure. DNS requests are not encrypted and susceptible to 'man-in-the-middle' attacks. For example, a coffee shop internet connection could easily subvert your DNS requests and send back different IP addresses for a particular domain

How do I connect to a TCP server (e.g. web server?)

TODO There are three basic system calls you need to connect to a remote machine:

```
getaddrinfo -- Determine the remote addresses of a remote host
socket  -- Create a socket
connect  -- Connect to the remote host using the socket and address information
```

The `getaddrinfo` call if successful, creates a linked-list of `addrinfo` structs and sets the given pointer to point to the first one.

The socket call creates an outgoing socket and returns a descriptor (sometimes called a 'file descriptor') that can be used with `read` and `write` etc.In this sense it is the network analog of `open` that opens a file stream - except that we haven't connected the socket to anything yet!

Finally the connect call attempts the connection to the remote machine. We pass the original socket descriptor and also the socket address information which is stored inside the addrinfo structure. There are different kinds of socket address structures (e.g. IPv4 vs IPv6) which can require more memory. So in addition to passing the pointer, the size of the structure is also passed:

```
// Pull out the socket address info from the addrinfo struct:
```

```
connect(sockfd, p->ai_addr, p->ai_addrlen)
```

How do I free the memory allocated for the linked-list of addrinfo structs?

As part of the clean up code call `freeaddrinfo` on the top-most `addrinfo` struct:

```
void freeaddrinfo(struct addrinfo *ai);
```

If getaddrinfo fails can I use `strerror` to print out the error?

No. Error handling with `getaddrinfo` is a little different:

- The return value *is* the error code (i.e. don't use `errno`)
- Use `gai_strerror` to get the equivalent short English error text:

```
int result = getaddrinfo(...);
if(result) {
   char *mesg = gai_strerror(result);
   ...
}
```

Can I request only IPv4 or IPv6 connection? TCP only?

Yes! Use the addrinfo structure that is passed into `getaddrinfo` to define the kind of connection you'd like.

For example, to specify stream-based protocols over IPv6:

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

hints.ai_family = AF_INET6; // Only want IPv6 (use AF_INET for IPv4)
hints.ai_socktype = SOCK_STREAM; // Only want stream-based connection
```

What about code examples that use `gethostbyname`?

The old function `gethostbyname` is deprecated; it's the old way convert a host name into an IP address. The port address still needs to be manually set using htons function. It's much easier to write code to support IPv4 AND IPv6 using the newer `getaddrinfo`

Is it that easy!?

Yes and no. It's easy to create a simple TCP client - however network communications offers many different levels of abstraction and several attributes and options that can be set at each level of abstraction (for example we haven't talked about `setsockopt` which can manipulate options for the socket). For more information see this guide.