


首先，初始化 hints 结构，使 getaddrinfo 返回我们想要的地址。在这里，我们想查找 32 位的 IP 地址(第 16 行)，用作连接的端点(第 17 行)。因为只想 getaddrinfo 转换域名，所以用 service 参数为 NULL 来调用它。

调用 getaddrinfo 之后，会遍历 addrinfo 结构，用 getnameinfo 将每个套接字地址转换成点分十进制地址字符串。遍历完列表之后，我们调用 freeaddrinfo 小心地释放这个列表(虽然对于这个简单的程序来说，并不是严格需要这样做的)。

运行 HOSTINFO 时，我们看到 twitter.com 映射到了四个 IP 地址，和 11.3.2 节用 NSLOOKUP 的结果一样。

```
linux> ./hostinfo twitter.com
199.16.156.102
199.16.156.230
199.16.156.6
199.16.156.70
```

 **练习题 11.4** 函数 getaddrinfo 和 getnameinfo 分别包含了 inet\_pton 和 inet\_ntop 的功能，提供了更高级别的、独立于任何特殊地址格式的抽象。想看看这到底有多方便，编写 HOSTINFO(图 11-17)的一个版本，用 inet\_pton 而不是 getnameinfo 将每个套接字地址转换成点分十进制地址字符串。

#### 11.4.8 套接字接口的辅助函数

初学时，getnameinfo 函数和套接字接口看上去有些可怕。用高级的辅助函数包装一下会方便很多，称为 open\_clientfd 和 open\_listenfd，客户端和服务端互相通信时可以使用这些函数。

##### 1. open\_clientfd 函数

客户端调用 open\_clientfd 建立与服务器的连接。

```
#include "csapp.h"

int open_clientfd(char *hostname, char *port);
```

返回：若成功则为描述符，若出错则为 -1。

open\_clientfd 函数建立与服务器的连接，该服务器运行在主机 hostname 上，并在端口号 port 上监听连接请求。它返回一个打开的套接字描述符，该描述符准备好了，可以用 Unix I/O 函数做输入和输出。图 11-18 给出了 open\_clientfd 的代码。

我们调用 getaddrinfo，它返回 addrinfo 结构的列表，每个结构指向一个套接字地址结构，可用于建立与服务器的连接，该服务器运行在 hostname 上并监听 port 端口。然后遍历该列表，依次尝试列表中的每个条目，直到调用 socket 和 connect 成功。如果 connect 失败，在尝试下一个条目之前，要小心地关闭套接字描述符。如果 connect 成功，我们会释放列表内存，并把套接字描述符返回给客户端，客户端可以立即开始用 Unix I/O 与服务器通信了。

注意，所有的代码都与任何版本的 IP 无关。socket 和 connect 的参数都是用 getaddrinfo 自动产生的，这使得我们的代码干净可移植。

##### 2. open\_listenfd 函数

调用 open\_listenfd 函数，服务器创建一个监听描述符，准备好接收连接请求。