11.4.2　socket 函数

客户端和服务器使用 socket 函数来创建一个套接字描述符(socket descriptor)。

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

int socket(int domain, int type, int protocol);
                                    返回：若成功则为非负描述符，若出错则为-1。
```

如果想要使套接字成为连接的一个端点，就用如下硬编码的参数来调用 socket 函数：

```
clientfd = Socket(AF_INET, SOCK_STREAM, 0);
```

其中，AF_INET 表明我们正在使用 32 位 IP 地址，而 SOCK_STREAM 表示这个套接字是连接的一个端点。不过最好的方法是用 getaddrinfo 函数(11.4.7 节)来自动生成这些参数，这样代码就与协议无关了。我们会在 11.4.8 节中向你展示如何配合 socket 函数来使用 getaddrinfo。

socket 返回的 clientfd 描述符仅是部分打开的，还不能用于读写。如何完成打开套接字的工作，取决于我们是客户端还是服务器。下一节描述当我们是客户端时如何完成打开套接字的工作。

11.4.3　connect 函数

客户端通过调用 connect 函数来建立和服务器的连接。

```
#include <sys/socket.h>

int connect(int clientfd, const struct sockaddr *addr,
            socklen_t addrlen);
                                    返回：若成功则为 0，若出错则为-1。
```

connect 函数试图与套接字地址为 addr 的服务器建立一个因特网连接，其中 addrlen 是 sizeof(sockaddr_in)。connect 函数会阻塞，一直到连接成功建立或是发生错误。如果成功，clientfd 描述符现在就准备好可以读写了，并且得到的连接是由套接字对

```
(x:y, addr.sin_addr:addr.sin_port)
```

刻画的，其中 x 表示客户端的 IP 地址，而 y 表示临时端口，它唯一地确定了客户端主机上的客户端进程。对于 socket，最好的方法是用 getaddrinfo 来为 connect 提供参数(见 11.4.8 节)。

11.4.4　bind 函数

剩下的套接字函数——bind、listen 和 accept，服务器用它们来和客户端建立连接。

```
#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *addr,
         socklen_t addrlen);
                                    返回：若成功则为 0，若出错则为-1。
```