bind 函数告诉内核将 addr 中的服务器套接字地址和套接字描述符 sockfd 联系起来。参数 addrlen 就是 sizeof(sockaddr_in)。对于 socket 和 connect，最好的方法是用 getaddrinfo 来为 bind 提供参数(见 11.4.8 节)。

11.4.5 listen 函数

客户端是发起连接请求的主动实体。服务器是等待来自客户端的连接请求的被动实体。默认情况下，内核会认为 socket 函数创建的描述符对应于*主动套接字*(active socket)，它存在于一个连接的客户端。服务器调用 listen 函数告诉内核，描述符是被服务器而不是客户端使用的。

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

int listen(int sockfd, int backlog);
                                        返回：若成功则为 0，若出错则为 -1。
```

listen 函数将 sockfd 从一个主动套接字转化为一个*监听套接字*(listening socket)，该套接字可以接受来自客户端的连接请求。backlog 参数暗示了内核在开始拒绝连接请求之前，队列中要排队的未完成的连接请求的数量。backlog 参数的确切含义要求对 TCP/IP 协议的理解，这超出了我们讨论的范围。通常我们会把它设置为一个较大的值，比如 1024。

11.4.6 accept 函数

服务器通过调用 accept 函数来等待来自客户端的连接请求。

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

int accept(int listenfd, struct sockaddr *addr, int *addrlen);
                                        返回：若成功则为非负连接描述符，若出错则为 -1。
```

accept 函数等待来自客户端的连接请求到达侦听描述符 listenfd，然后在 addr 中填写客户端的套接字地址，并返回一个*已连接描述符*(connected descriptor)，这个描述符可被用来利用 Unix I/O 函数与客户端通信。

监听描述符和已连接描述符之间的区别使很多人感到迷惑。监听描述符是作为客户端连接请求的一个端点。它通常被创建一次，并存在于服务器的整个生命周期。已连接描述符是客户端和服务器之间已经建立起来了的连接的一个端点。服务器每次接受连接请求时都会创建一次，它只存在于服务器为一个客户端服务的过程中。

图 11-14 描绘了监听描述符和已连接描述符的角色。在第一步中，服务器调用 accept，等待连接请求到达监听描述符，具体地我们设定为描述符 3。回忆一下，描述符 0～2 是预留给了标准文件的。

在第二步中，客户端调用 connect 函数，发送一个连接请求到 listenfd。第三步，accept 函数打开了一个新的已连接描述符 connfd(我们假设是描述符 4)，在 clientfd 和 connfd 之间建立连接，并且随后返回 connfd 给应用程序。客户端也从 connect 返回，在这一点以后，客户端和服务器就可以分别通过读和写 clientfd 和 connfd 来回传送数据了。