```
#include "csapp.h"

int open_listenfd(char *port);
```

返回：若成功则为描述符，若出错则为 −1。

code/src/csapp.c

```
1  int open_clientfd(char *hostname, char *port) {
2      int clientfd;
3      struct addrinfo hints, *listp, *p;
4
5      /* Get a list of potential server addresses */
6      memset(&hints, 0, sizeof(struct addrinfo));
7      hints.ai_socktype = SOCK_STREAM;  /* Open a connection */
8      hints.ai_flags = AI_NUMERICSERV;  /* ... using a numeric port arg. */
9      hints.ai_flags |= AI_ADDRCONFIG;  /* Recommended for connections */
10     Getaddrinfo(hostname, port, &hints, &listp);
11
12     /* Walk the list for one that we can successfully connect to */
13     for (p = listp; p; p = p->ai_next) {
14         /* Create a socket descriptor */
15         if ((clientfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol))
16             < 0) continue; /* Socket failed, try the next */
17
18         /* Connect to the server */
19         if (connect(clientfd, p->ai_addr, p->ai_addrlen) != -1)
20             break; /* Success */
21         Close(clientfd); /* Connect failed, try another */
22     }
23
24     /* Clean up */
25     Freeaddrinfo(listp);
26     if (!p) /* All connects failed */
27         return -1;
28     else    /* The last connect succeeded */
29         return clientfd;
30 }
```

code/src/csapp.c

图 11-18　open_clientfd：和服务器建立连接的辅助函数。它是可重入和与协议无关的

open_listenfd 函数打开和返回一个监听描述符，这个描述符准备好在端口 port 上接收连接请求。图 11-19 展示了 open_listenfd 的代码。

open_listenfd 的风格类似于 open_clientfd。调用 getaddrinfo，然后遍历结果列表，直到调用 socket 和 bind 成功。注意，在第 20 行，我们使用 setsockopt 函数(本书中没有讲述)来配置服务器，使得服务器能够被终止、重启和立即开始接收连接请求。一个重启的服务器默认将在大约 30 秒内拒绝客户端的连接请求，这严重地阻碍了调试。

因为我们调用 getaddrinfo 时，使用了 AI_PASSIVE 标志并将 host 参数设置为 NULL，每个套接字地址结构中的地址字段会被设置为通配符地址，这告诉内核这个服务器会接收发送到本主机所有 IP 地址的请求。