```
20      static pool pool;
21
22      if (argc != 2) {
23          fprintf(stderr, "usage: %s <port>\n", argv[0]);
24          exit(0);
25      }
26      listenfd = Open_listenfd(argv[1]);
27      init_pool(listenfd, &pool);
28
29      while (1) {
30          /* Wait for listening/connected descriptor(s) to become ready */
31          pool.ready_set = pool.read_set;
32          pool.nready = Select(pool.maxfd+1, &pool.ready_set, NULL, NULL, NULL);
33
34          /* If listening descriptor ready, add new client to pool */
35          if (FD_ISSET(listenfd, &pool.ready_set)) {
36              clientlen = sizeof(struct sockaddr_storage);
37              connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
38              add_client(connfd, &pool);
39          }
40
41          /* Echo a text line from each ready connected descriptor */
42          check_clients(&pool);
43      }
44  }
```

code/conc/echoservers.c

图 12-8　(续)

init_pool 函数(图 12-9)初始化客户端池。clientfd 数组表示已连接描述符的集合，其中整数−1 表示一个可用的槽位。初始时，已连接描述符集合是空的(第 5～7 行)，而且监听描述符是 select 读集合中唯一的描述符(第 10～12 行)。

code/conc/echoservers.c

```
1   void init_pool(int listenfd, pool *p)
2   {
3       /* Initially, there are no connected descriptors */
4       int i;
5       p->maxi = -1;
6       for (i=0; i< FD_SETSIZE; i++)
7           p->clientfd[i] = -1;
8
9       /* Initially, listenfd is only member of select read set */
10      p->maxfd = listenfd;
11      FD_ZERO(&p->read_set);
12      FD_SET(listenfd, &p->read_set);
13  }
```

code/conc/echoservers.c

图 12-9　init_pool 初始化活动客户端池

add_client 函数(图 12-10)添加一个新的客户端到活动客户端池中。在 clientfd 数组中找到一个空槽位后，服务器将这个已连接描述符添加到数组中，并初始化相应的 RIO 读缓冲区，这样一来我们就能够对这个描述符调用 rio_readlineb(第 8～9 行)。然