状态机(state machine)就是一组状态(state)、输入事件(input event)和转移(transition)，其中转移是将状态和输入事件映射到状态。每个转移是将一个(输入状态，输入事件)对映射到一个输出状态。自循环(self-loop)是同一输入和输出状态之间的转移。通常把状态机画成有向图，其中节点表示状态，有向弧表示转移，而弧上的标号表示输入事件。一个状态机从某种初始状态开始执行。每个输入事件都会引发一个从当前状态到下一状态的转移。

对于每个新的客户端 k，基于 I/O 多路复用的并发服务器会创建一个新的状态机 s_k，并将它和已连接描述符 d_k 联系起来。如图 12-7 所示，每个状态机 s_k 都有一个状态(“等待描述符 d_k 准备好可读”)、一个输入事件(“描述符 d_k 准备好可以读了”)和一个转移(“从描述符 d_k 读一个文本行”)。

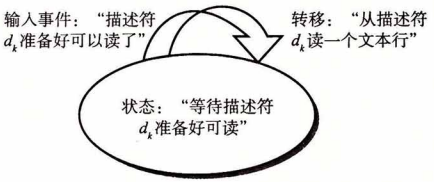

图 12-7　并发事件驱动 echo 服务器中逻辑流的状态机

服务器使用 I/O 多路复用，借助 select 函数检测输入事件的发生。当每个已连接描述符准备好可读时，服务器就为相应的状态机执行转移，在这里就是从描述符读和写回一个文本行。

图 12-8 展示了一个基于 I/O 多路复用的并发事件驱动服务器的完整示例代码。一个 pool 结构里维护着活动客户端的集合(第 3～11 行)。在调用 init_pool 初始化池(第 27 行)之后，服务器进入一个无限循环。在循环的每次迭代中，服务器调用 select 函数来检测两种不同类型的输入事件：a)来自一个新客户端的连接请求到达，b)一个已存在的客户端的已连接描述符准备好可以读了。当一个连接请求到达时(第 35 行)，服务器打开连接(第 37 行)，并调用 add_client 函数，将该客户端添加到池里(第 38 行)。最后，服务器调用 check_clients 函数，把来自每个准备好的已连接描述符的一个文本行回送回去(第 42 行)。

code/conc/echoservers.c

```
1   #include "csapp.h"
2
3   typedef struct { /* Represents a pool of connected descriptors */
4       int maxfd;        /* Largest descriptor in read_set */
5       fd_set read_set;  /* Set of all active descriptors */
6       fd_set ready_set; /* Subset of descriptors ready for reading  */
7       int nready;       /* Number of ready descriptors from select */
8       int maxi;         /* High water index into client array */
9       int clientfd[FD_SETSIZE];    /* Set of active descriptors */
10      rio_t clientrio[FD_SETSIZE]; /* Set of active read buffers */
11  } pool;
12
13  int byte_cnt = 0; /* Counts total bytes received by server */
14
15  int main(int argc, char **argv)
16  {
17      int listenfd, connfd;
18      socklen_t clientlen;
19      struct sockaddr_storage clientaddr;
```

图 12-8　基于 I/O 多路复用的并发 echo 服务器。每次服务器迭代都回送来自每个准备好的描述符的文本行