单，但是有几个普遍而且有些微妙的问题需要我们更仔细地看一看。第一个问题是当我们调用 pthread_create 时，如何将已连接描述符传递给对等线程。最明显的方法就是传递一个指向这个描述符的指针，就像下面这样

```
connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
Pthread_create(&tid, NULL, thread, &connfd);
```

然后，我们让对等线程间接引用这个指针，并将它赋值给一个局部变量，如下所示

```
void *thread(void *vargp) {
    int connfd = *((int *)vargp);
    .
    .
    .
}
```

code/conc/echoservert.c

```
1   #include "csapp.h"
2
3   void echo(int connfd);
4   void *thread(void *vargp);
5
6   int main(int argc, char **argv)
7   {
8       int listenfd, *connfdp;
9       socklen_t clientlen;
10      struct sockaddr_storage clientaddr;
11      pthread_t tid;
12
13      if (argc != 2) {
14          fprintf(stderr, "usage: %s <port>\n", argv[0]);
15          exit(0);
16      }
17      listenfd = Open_listenfd(argv[1]);
18
19      while (1) {
20          clientlen=sizeof(struct sockaddr_storage);
21          connfdp = Malloc(sizeof(int));
22          *connfdp = Accept(listenfd, (SA *) &clientaddr, &clientlen);
23          Pthread_create(&tid, NULL, thread, connfdp);
24      }
25  }
26
27  /* Thread routine */
28  void *thread(void *vargp)
29  {
30      int connfd = *((int *)vargp);
31      Pthread_detach(pthread_self());
32      Free(vargp);
33      echo(connfd);
34      Close(connfd);
35      return NULL;
36  }
```

code/conc/echoservert.c

图 12-14　基于线程的并发 echo 服务器