TCP/IP网络编程(十六)

笔记本: 网络编程
创建时间: 2018/11/16 9:56 **更新时间:** 2018/11/19 21:13
作者: xiangkang94@outlook.com
标签: 第十六章(I/O流分离的相关内容)

- IO流分离方式（两种）
  - 第十章TCP IO routine分离，通过fork文件描述符区分输入输出，虽然文件描述符不会根据输入、输出进行区分，但是分开了两个文件描述符的用途（父进程负责读，子进程负责写）
  - 第十五章 调用fdopen创建FILE指针，分离输入工具和输出工具。
- 分离流的好处
  - //通过fork文件描述符区分输入输出
  - 通过分开输入过程和输出过程降低实现难度
  - 与输入无关的输出操作可以提高速度
  - //调用fdopen创建FILE指针
  - 将FILE指针按读模式、写模式加以区分
  - 可以通过区分读写模式降低难度
  - 通过区分IO缓冲提高缓冲性能
  - 将文件描述符转成文件指针后，可以使用标准IO函数
- 流分离带来的EOF问题
  - 在流分离的情况下半关闭连接需要额外处理
  - 读模式FILE指针、写模式FILE指针都是基于同一个文件描述符创建的，fclose关闭任意一个，都将关闭文件描述符，套接字终止

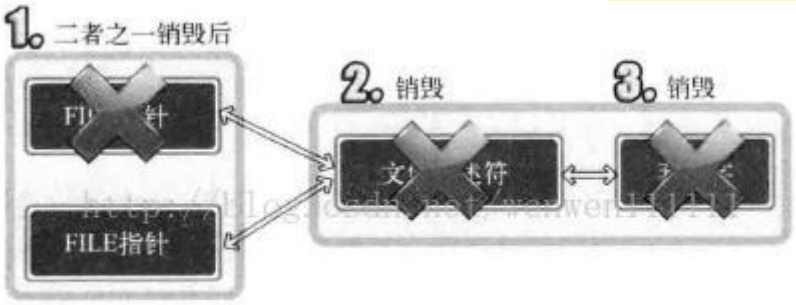
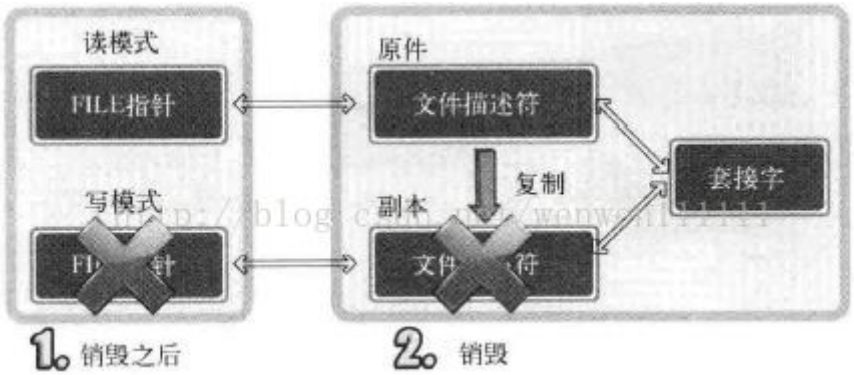


图16-2 调用fclose函数的结果

  - 解决如上问题，只需在创建FILE指针前复制文件描述符



  - 但是上图中虽然关闭了其中一个文件文件描述符，但是仍然没有进入半关闭状态，因为剩下的这个文件描述符，仍然可以同时进行I/O

- 复制文件描述符的方法dup/dup2

```
#include <unistd.h>
int dup(int fildes)
int dup2(int fildes, int fildes2)
```

```
//成功时返回复制的文件描述符，失败返回-1
//fildes：需要复制的文件描述符
//filders2：明确指定的文件描述符整数值
```

- **无论复制出多少文件描述符，均应该通过shutdown函数发送EOF并进入半关闭状态**

```
server.c：
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define BUF_SIZE 1024

int main(int argc, char * argv[]){
    int serv_sock,clnt_sock;
    FILE *readfp;
    FILE *writefp;
    struct sockaddr_in serv_adr,clnt_adr;
    socklen_t clnt_adr_sz;
    char buf[BUF_SIZE];

    serv_sock = socket(PF_INET,SOCK_STREAM,0);
    memset(&serv_adr,0,sizeof(serv_adr));
    serv_adr.sin_family = AF_INET;
    serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_adr.sin_port = htons(atoi(argv[1]));

    bind(serv_sock,(struct sockaddr *)&serv_adr,sizeof(serv_adr));
    listen(serv_sock,5);
    clnt_adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock,(struct sockaddr *)&clnt_adr,&clnt_adr_sz);

    readfp = fdopen(clnt_sock,"r");
    writefp = fdopen(dup(clnt_sock),"w");

    fputs("from server: Hi~ client? \n",writefp);
    fputs("I love all the world! \n",writefp);
    fflush(writefp);

    shutdown(fileno(writefp),SHUT_WR);   //keypoint,无论复制了多少文件描述符，只要调用了shutdown均会进入半状态
    fclose(writefp);
    fgets(buf,sizeof(buf),readfp);
    fputs(buf,stdout);
    fclose(readfp);
    return 0;
}
```