#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/select.h>

#include <sys/time.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <poll.h>

#include <errno.h>

#define OPEN\_MAX 100

int main(int argc, char \*argv[])

{

    //1.创建tcp监听套接字

    int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    //2.绑定sockfd

    struct sockaddr\_in my\_addr;

    bzero(&my\_addr, sizeof(my\_addr));

    my\_addr.sin\_family = AF\_INET;

    my\_addr.sin\_port = htons(8000);

    my\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(my\_addr));

    //3.监听listen

    listen(sockfd, 10);

    //4.poll相应参数准备

    struct pollfd client[OPEN\_MAX];

    int i = 0, maxi = 0;

    for(;i<OPEN\_MAX; i++)

        client[i].fd = -1;//初始化poll结构中的文件描述符fd

    client[0].fd = sockfd;//需要监测的描述符

    client[0].events = POLLIN;//普通或优先级带数据可读

    //5.对已连接的客户端的数据处理

    while(1)

    {

        int ret = poll(client, maxi+1, -1);//对加入poll结构体数组所有元素进行监测

        //5.1监测sockfd(监听套接字)是否存在连接

        if((client[0].revents & POLLIN) == POLLIN )

        {

            struct sockaddr\_in cli\_addr;

            int clilen = sizeof(cli\_addr);

            int connfd = 0;

            //5.1.1 从tcp完成连接中提取客户端

            connfd = accept(sockfd, (struct sockaddr \*)&cli\_addr, &clilen);

            //5.1.2 将提取到的connfd放入poll结构体数组中，以便于poll函数监测

            for(i=1; i<OPEN\_MAX; i++)

            {

                if(client[i].fd < 0)

                {

                    client[i].fd = connfd;

                    client[i].events = POLLIN;

                    break;

                }

            }

            //5.1.3 maxi更新

            if(i > maxi)

                maxi = i;

            //5.1.4 如果没有就绪的描述符，就继续poll监测，否则继续向下看

            if(--ret <= 0)

                continue;

        }

        //5.2继续响应就绪的描述符

        for(i=1; i<=maxi; i++)

        {

            if(client[i].fd < 0)

                continue;

            if(client[i].revents & (POLLIN | POLLERR))

            {

                int len = 0;

                char buf[128] = "";

                //5.2.1接受客户端数据

                if((len = recv(client[i].fd, buf, sizeof(buf), 0)) < 0)

                {

                    if(errno == ECONNRESET)//tcp连接超时、RST

                    {

                        close(client[i].fd);

                        client[i].fd = -1;

                    }

                    else

                        perror("read error:");

                }

                else if(len == 0)//客户端关闭连接

                {

                    close(client[i].fd);

                    client[i].fd = -1;

                }

                else//正常接收到服务器的数据

                    send(client[i].fd, buf, len, 0);

                //5.2.2所有的就绪描述符处理完了，就退出当前的for循环，继续poll监测

                if(--ret <= 0)

                    break;

            }

        }

    }

    return 0;

}