epoll的web服务器

实现了在web端可以查看服务器上的文件（已经支持交叉编译后在hisi3559上运行）

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/epoll.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

#include <fcntl.h>


#define Port 80
#define IPADDRESS  "192.168.1.9"
#define LISTEN_NUM 30
#define FDSIZE 1000
#define EPOLLEVENTS 100
static int socket_bind(const char IP,int PORT);
static int add_event(int epoll_fd,int fd,int state);
static void do_epoll(int listenfd);
static void handle_event(int epoll_fd,int num,struct epoll_event * events,int listenfd);
static void hand_accept(int epollfd,int listenfd);
static int add_event(int epoll_fd,int fd,int state);
static void do_read(int epoll_fd,int fd);
//static void do_write();
static int get_line(int cfd,char *buf,int size);
static void http_request(int cfd,const char *file);
static void send_respondHttp(int cfd,int StateNum,char *description,char *type);
static void send_file(int cfd,const char *file);

//void send_dir(cfd,file);


```

```c
int socket_bind(const char IP,int PORT)
{
    struct sockaddr_in server_addr;
    int lfd;
    int ret;
    lfd = socket(AF_INET,SOCK_STREAM,0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    //server_addr.sin_addr.s_addr = inet_pton(AF_INET,IPADDRESS,&server_addr.sin_addr);
   server_addr.sin_addr.s_addr = INADDR_ANY;


   ret = bind(lfd,(struct sockaddr*)&server_addr,sizeof(server_addr));
    if(ret == -1){
        perror("bind err!");
        exit(1);
    }
    ret = listen(lfd,LISTEN_NUM);
if(ret == -1){
        perror("listen err!");
        exit(1);
    }
    return lfd;
}
static void do_epoll(int listenfd)
{
    int epoll_fd;
    struct epoll_event events[EPOLLEVENTS];
    int num;
    epoll_fd = epoll_create(FDSIZE);
    if(epoll_fd == -1 ){
        perror("epoll err!");
        exit(1);
    }
    add_event(epoll_fd,listenfd,EPOLLIN);
    for( ;; )
    {
       num = epoll_wait(epoll_fd,&events,EPOLLEVENTS,-1);

       handle_event(epoll_fd,num,events,listenfd);
```

```c
        num = 0;
    }
}

static void handle_event(int epoll_fd,int num,struct epoll_event * events,int listenfd)
{
    int i = 0;
    int fd;
    for(i=0;i<num;i++)
    {

    fd = events[i].data.fd;
    if((fd == listenfd))
        hand_accept(epoll_fd,listenfd);
    else if(events[i].events == EPOLLIN)
        do_read(epoll_fd,fd);
   // else if(events[i].events == EPOLLOUT)
   //    do_write();
    }
}
static void hand_accept(int epollfd,int listenfd)
{
    int clifd;
    struct sockaddr_in cli_sockaddr;
    socklen_t socklen;
    socklen = sizeof(cli_sockaddr);
    memset(&cli_sockaddr,0,sizeof(cli_sockaddr));
    clifd = accept(listenfd,(struct sockaddr*)&cli_sockaddr,&socklen);

    if(clifd == -1){
        perror("accept err");
     //  exit(1);
    }
    else{
        printf("client ip is %s port is %d \n",inet_ntoa(cli_sockaddr.sin_addr),cli_socka
    }
    add_event(epollfd,clifd,EPOLLIN);
}

static int add_event(int epoll_fd,int fd,int state)
```

```c
{
    struct epoll_event events;
    events.data.fd = fd;
    events.events = state;
    epoll_ctl(epoll_fd,EPOLL_CTL_ADD,fd,&events);
}

static int delete_event(int epoll_fd,int fd,int state)
{
    struct epoll_event events;
    events.data.fd = fd;
    events.events = state;
    epoll_ctl(epoll_fd,EPOLL_CTL_DEL,fd,&events);
}

static void do_read(int epoll_fd,int fd)
{
   int len;
   char line[1024] = {0};
   char method[16],path[64],protocol[16];
   //读取一行进行http拆分 获取get方法以及文件名
   len = get_line(fd,line,sizeof(line));

   if(len == 0){
   printf("服务器，检查到客户端关闭...\n");
   }
   else{

   sscanf(line,"%[^ ] %[^ ] %[^ ]",method,path,protocol);
   printf("method = %s,path = %s,protocol =%s",method,path,protocol);
   }
   //检测GET方法
   if(strncasecmp(method,"GET",3) == 0){
           //处理http请求
           char *file = path+1;//这里+1是因为 文件名称里面有代表路径的/
           printf("-------%s----\n",file);
           if(strcmp(path,"/") == 0)
                   file = "./";
   http_request(fd,file);//发送HTTP数据的表头
   //发送数据
```

```c
    send_file(fd,file);
    }

    delete_event(epoll_fd,fd,EPOLLIN);
}
//发送服务器本地文件给客户端
static void send_file(int cfd,const char *file)
{
    int n = 0;
    int buf[1024] = {0};
    int fd = open(file,O_RDONLY);
    if(fd == -1){
    perror("sendfile open fail:");
    exit(1);
    }
    while((n = read(fd,buf,sizeof(buf))) > 0){
    send(cfd,buf,n,0);
    }
    close(fd);
}
static void http_request(int cfd,const char *file)
{

    int ret;
    //判断文件是否存在
    struct stat sbuf;
    ret = stat(file,&sbuf);
    if(ret != 0){
    //回发404页面
    perror("stat file compare fail:");
    //exit(1);
    }

    if(S_ISREG(sbuf.st_mode)){//满足条件的话说明是一个普通文件

            //回发http协议应答
            send_respondHttp(cfd,200,"OK","Content-Type:text/plain;charset=iso-8859-1");
            //回发给客户端请求数据内容
    }
    else if(S_ISDIR(sbuf.st_mode)){
```

```c
                send_respondHttp(cfd,200,"OK","Content-Type:text/html;charset=iso-8859-1");
                //send_dir(cfd,file);
        }
}
//HTTP/1.1 200(状态码) Ok（对状态码的描述）
//Content-Type:text/plain;charset=iso-8859-1(必写项): 文本类型，编码类型

static void send_respondHttp(int cfd,int StateNum,char *description,char *type)
{
 //将要send出去的信息先拼起来
 char buf[1024] = {0};
 sprintf(buf,"HTTP/1.1 %d %s\r\n",StateNum,description);//注意http协议是以/r/n结尾的
 sprintf(buf+strlen(buf),"%s\r\n",type);
 send(cfd,buf,strlen(buf),0);
 send(cfd,"\r\n",2,0);
}
#if 0
void send_dir(cfd,file)
{
    //打开目录
    DIR* dir = opendir(file);
    if(dir == NULL){
    perror("opendir err");
    exit(1);
    }
    //读目录
    struct dirent* ptr = NULL;
    while((ptr = readdir(dir)) != NULL)
            {
            char* name = ptr->d_name;
    }
    closedir(dir);
}
#endif
static int get_line(int cfd,char *buf,int size)
{
    int i = 0;
    char c = '\0';
    int n = 0;
    while((i < size-1) && (c != '\n')){
```

```
    n = recv(cfd,&c,1,0);
    if(n>0){

            if(c == '\r'){
                    n = recv(cfd,&c,1,MSG_PEEK);
                    if((n > 0) && (c == '\n'))
                            {
                            n = recv(cfd,&c,1,0);
                            }
                    else{
                            c = '\n';
                            }
            }
            buf[i] = c;
            i++;
    }
    else{
    c = '\n';
    }
    }
    //buf[i] = '\0';
    return i;
}
int main(int argc,char *argv[])
{
     int listenfd;
    int ret;
    if(argc < 3)
            {

    printf("main parameter is err");
    exit(1);
    }
    printf("argc is %d\n",argc);
    ret = chdir(argv[2]);
    if(ret !=0){
    perror("chdir err\n");
    exit(1);
    }
    listenfd = socket_bind(IPADDRESS,Port);
```

```c
    do_epoll(listenfd);
    return 0;
}
```