

# 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）

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <errno.h>
5  #include <netinet/in.h>
6  #include <sys/socket.h>
7  #include <arpa/inet.h>
8  #include <sys/epoll.h>
9  #include <unistd.h>
10 #include <sys/types.h>
11 #include <sys/stat.h>
12
13 #include <fcntl.h>
14
15
16 #define Port 80
17 #define IPADDRESS "192.168.1.9"
18 #define LISTEN_NUM 30
19 #define FDSIZE 1000
20 #define EPOLLEVENTS 100
21 static int socket_bind(const char IP,int PORT);
22 static int add_event(int epoll_fd,int fd,int state);
23 static void do_epoll(int listenfd);
24 static void handle_event(int epoll_fd,int num,struct epoll_event * events,int listenfd);
25 static void hand_accept(int epollfd,int listenfd);
26 static int add_event(int epoll_fd,int fd,int state);
27 static void do_read(int epoll_fd,int fd);
28 //static void do_write();
29 static int get_line(int cfd,char *buf,int size);
30 static void http_request(int cfd,const char *file);
31 static void send_respondHttp(int cfd,int StateNum,char *description,char *type);
32 static void send_file(int cfd,const char *file);
33
34 //void send_dir(cfd,file);
35
36
```

```

37 int socket_bind(const char IP,int PORT)
38 {
39     struct sockaddr_in server_addr;
40     int lfd;
41     int ret;
42     lfd = socket(AF_INET,SOCK_STREAM,0);
43     server_addr.sin_family = AF_INET;
44     server_addr.sin_port = htons(PORT);
45     //server_addr.sin_addr.s_addr = inet_pton(AF_INET,IPADDRESS,&server_addr.sin_addr);
46     server_addr.sin_addr.s_addr = INADDR_ANY;
47
48
49     ret = bind(lfd,(struct sockaddr*)&server_addr,sizeof(server_addr));
50     if(ret == -1){
51         perror("bind err!");
52         exit(1);
53     }
54     ret = listen(lfd,LISTEN_NUM);
55     if(ret == -1){
56         perror("listen err!");
57         exit(1);
58     }
59     return lfd;
60 }
61 static void do_epoll(int listenfd)
62 {
63     int epoll_fd;
64     struct epoll_event events[EPOLLEVENTS];
65     int num;
66     epoll_fd = epoll_create(FDSIZE);
67     if(epoll_fd == -1 ){
68         perror("epoll err!");
69         exit(1);
70     }
71     add_event(epoll_fd,listenfd,E POLLIN);
72     for( ;; )
73     {
74         num = epoll_wait(epoll_fd,&events,EPOLLEVENTS,-1);
75
76         handle_event(epoll_fd,num,events,listenfd);

```

```

77     num = 0;
78 }
79 }
80
81 static void handle_event(int epoll_fd,int num,struct epoll_event * events,int listenfd)
82 {
83     int i = 0;
84     int fd;
85     for(i=0;i<num;i++)
86     {
87
88         fd = events[i].data.fd;
89         if((fd == listenfd))
90             hand_accept(epoll_fd,listenfd);
91         else if(events[i].events == EPOLLIN)
92             do_read(epoll_fd,fd);
93         // else if(events[i].events == EPOLLOUT)
94         //     do_write();
95     }
96 }
97 static void hand_accept(int epollfd,int listenfd)
98 {
99     int clifd;
100     struct sockaddr_in cli_sockaddr;
101     socklen_t socklen;
102     socklen = sizeof(cli_sockaddr);
103     memset(&cli_sockaddr,0,sizeof(cli_sockaddr));
104     clifd = accept(listenfd,(struct sockaddr*)&cli_sockaddr,&socklen);
105
106     if(clifd == -1){
107         perror("accept err");
108         // exit(1);
109     }
110     else{
111         printf("client ip is %s port is %d \n",inet_ntoa(cli_sockaddr.sin_addr),cli_sockaddr.sin_port);
112     }
113     add_event(epollfd,clifd,EPOLLIN);
114 }
115
116 static int add_event(int epoll_fd,int fd,int state)

```

```
117 {
118     struct epoll_event events;
119     events.data.fd = fd;
120     events.events = state;
121     epoll_ctl(epoll_fd, EPOLL_CTL_ADD, fd, &events);
122 }
123
124 static int delete_event(int epoll_fd, int fd, int state)
125 {
126     struct epoll_event events;
127     events.data.fd = fd;
128     events.events = state;
129     epoll_ctl(epoll_fd, EPOLL_CTL_DEL, fd, &events);
130 }
131
132 static void do_read(int epoll_fd, int fd)
133 {
134     int len;
135     char line[1024] = {0};
136     char method[16], path[64], protocol[16];
137     //读取一行进行http拆分 获取get方法以及文件名
138     len = get_line(fd, line, sizeof(line));
139
140     if(len == 0){
141         printf("服务器, 检测到客户端关闭...\n");
142     }
143     else{
144
145         sscanf(line, "%[^ ] %[^ ] %[^ ]", method, path, protocol);
146         printf("method = %s, path = %s, protocol = %s", method, path, protocol);
147     }
148     //检测GET方法
149     if(strncasecmp(method, "GET", 3) == 0){
150         //处理http请求
151         char *file = path+1; //这里+1是因为 文件名称里面有代表路径的/
152         printf("-----%s----\n", file);
153         if(strcmp(path, "/") == 0)
154             file = "./";
155         http_request(fd, file); //发送HTTP数据的表头
156         //发送数据
```

```

157     send_file(fd,file);
158 }
159
160     delete_event(epoll_fd,fd,EPOLLIN);
161 }
162 //发送服务器本地文件给客户端
163 static void send_file(int cfd,const char *file)
164 {
165     int n = 0;
166     int buf[1024] = {0};
167     int fd = open(file,O_RDONLY);
168     if(fd == -1){
169         perror("sendfile open fail:");
170         exit(1);
171     }
172     while((n = read(fd,buf,sizeof(buf))) > 0){
173         send(cfd,buf,n,0);
174     }
175     close(fd);
176 }
177 static void http_request(int cfd,const char *file)
178 {
179
180     int ret;
181     //判断文件是否存在
182     struct stat sbuf;
183     ret = stat(file,&sbuf);
184     if(ret != 0){
185         //回发404页面
186         perror("stat file compare fail:");
187         //exit(1);
188     }
189
190     if(S_ISREG(sbuf.st_mode)){//满足条件的话说明是一个普通文件
191
192         //回发http协议应答
193         send_respondHttp(cfd,200,"OK","Content-Type:text/plain;charset=iso-8859-1");
194         //回发给客户端请求数据内容
195     }
196
197     else if(S_ISDIR(sbuf.st_mode)){

```

```

197         send_responseHttp(cfd,200,"OK","Content-Type:text/html;charset=iso-8859-1");
198         //send_dir(cfd,file);
199     }
200 }
201 //HTTP/1.1 200(状态码) Ok (对状态码的描述)
202 //Content-Type:text/plain;charset=iso-8859-1(必写项): 文本类型, 编码类型
203
204 static void send_responseHttp(int cfd,int StateNum,char *description,char *type)
205 {
206     //将要send出去的信息先拼起来
207     char buf[1024] = {0};
208     sprintf(buf,"HTTP/1.1 %d %s\r\n",StateNum,description);//注意http协议是以\r\n结尾的
209     sprintf(buf+strlen(buf),"%s\r\n",type);
210     send(cfd,buf,strlen(buf),0);
211     send(cfd,"\r\n",2,0);
212 }
213 #if 0
214 void send_dir(cfd,file)
215 {
216     //打开目录
217     DIR* dir = opendir(file);
218     if(dir == NULL){
219         perror("opendir err");
220         exit(1);
221     }
222     //读目录
223     struct dirent* ptr = NULL;
224     while((ptr = readdir(dir)) != NULL)
225     {
226         char* name = ptr->d_name;
227     }
228     closedir(dir);
229 }
230 #endif
231 static int get_line(int cfd,char *buf,int size)
232 {
233     int i = 0;
234     char c = '\0';
235     int n = 0;
236     while((i < size-1) && (c != '\n')){

```

```

237     n = recv(cfd,&c,1,0);
238     if(n>0){
239
240         if(c == '\r'){
241             n = recv(cfd,&c,1,MSG_PEEK);
242             if((n > 0) && (c == '\n'))
243                 {
244                     n = recv(cfd,&c,1,0);
245                 }
246             else{
247                 c = '\n';
248             }
249         }
250         buf[i] = c;
251         i++;
252     }
253     else{
254         c = '\n';
255     }
256 }
257 //buf[i] = '\0';
258 return i;
259 }
260 int main(int argc,char *argv[])
261 {
262     int listenfd;
263     int ret;
264     if(argc < 3)
265     {
266
267         printf("main parameter is err");
268         exit(1);
269     }
270     printf("argc is %d\n",argc);
271     ret = chdir(argv[2]);
272     if(ret !=0){
273         perror("chdir err\n");
274         exit(1);
275     }
276     listenfd = socket_bind(IPADDRESS,Port);

```

```
277     do_epoll(listenfd);  
278     return 0;  
279 }
```