<https://www.cnblogs.com/gongxijun/p/4702738.html>

Linux关于并发网络分为Apache模型（Process per Connection (进程连接) ） 和TPC ， 还有select模型，以及poll模型（一般是Epoll模型）

 Select模型极其作用：这文章讲述的很好，没必要重述已有的东西，就直接给链接

<http://blog.csdn.net/turkeyzhou/article/details/8609360>

        我的理解：

[复制代码](javascript:void(0);)

1 /\* According to POSIX.1-2001 \*/

2 #include <sys/select.h>

3

4 /\* According to earlier standards \*/

5 #include <sys/time.h>

6 #include <sys/types.h>

7 #include <unistd.h>

8

9 int select(int nfds, fd\_set \*readfds, fd\_set \*writefds,

10 fd\_set \*exceptfds, struct timeval \*timeout);

11

12 void FD\_CLR(int fd, fd\_set \*set);

13 int FD\_ISSET(int fd, fd\_set \*set);

14 void FD\_SET(int fd, fd\_set \*set);

15 void FD\_ZERO(fd\_set \*set);

[复制代码](javascript:void(0);)

    对于

int select(int nfds, fd\_set \*readfds, fd\_set \*writefds, fd\_set \*exceptfds, struct timeval \*timeout);  
 第一个参数 nfds： 第n个文件id的编号 （linux下，一切皆文件） 需要注意的是： nfds = fd+1 (fd 为 FD\_SET中的fd)  
 第二个参数： fd\_set \*readfds 读取文件编号，如果不需要读取的话 可以设置为NULL  
 第三 ，四个参数： 同上  
 第五个参数：为一个定义超时的结构体

1 struct timeval {

2 time\_t tv\_sec; /\* seconds \*/

3 suseconds\_t tv\_usec; /\* microseconds \*/

4 };

         该结构用来设定多少时间为超时 ,比如

struct timeval ss ;

ss.tv\_sec =3;

ss.tv\_usec =0;

//表示设定为3秒后为超时，select将会返回0

对于下面这几个函数：

    1 void FD\_CLR(int fd, fd\_set \*set);

      用来清除fd的fd\_set  ,比如fd为5  ，则表示set集中所有设定等于5的fd\_set 都将被清除

 1 int FD\_ISSET(int fd, fd\_set \*set);

     判断是否set 与fd是否绑定，如果没有绑定，则返回false，如果绑定了则返回True

void FD\_SET(int fd, fd\_set \*set);

        将fd值 和set绑定

void FD\_ZERO(fd\_set \*set);

   将set集全部清除

简单的例子：

判断是否有数据输入，有则直接打印出来

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 #include<stdio.h>

2 #include<string.h>

3 #include<sys/select.h>

4 #include<unistd.h>

5 #include<sys/time.h>

6 #include<sys/types.h>

7 #define maxn 6

8 #define EXIT\_FAILURE -1

9 #define EXIT\_SUCCESS 0

10

11 int main(int argc , char \* argv []){

12

13 fd\_set mtfd ;

14 int ffd =0;

15 struct timeval outtime ;

16 int retval ;

17 char redbuf[maxn];

18

19 FD\_ZERO(&mtfd);

20 FD\_SET(ffd , &mtfd);

21

22 // wait up to 5.5 s

23 outtime.tv\_sec = 5 ;

24 outtime.tv\_usec = 500 ;

25 retval = select(ffd+1 , &mtfd ,NULL , NULL , &outtime);

26 if(-1 == retval )

27 {

28 printf("error happened ! %d \n" ,\_\_LINE\_\_ );

29 return EXIT\_FAILURE ;

30 }

31 else if(0 == retval ){

32 printf(" timeout !! %d \n" ,\_\_LINE\_\_ );

33 return EXIT\_FAILURE ;

34 }

35 //means that is good !

36

37 printf("retval = %d \n", retval);

38

39 if( FD\_ISSET(ffd , &mtfd) ){

40

41 memset(redbuf ,0 , sizeof(redbuf));

42 printf("reading ... !! ");

43 // read(1 , redbuf ,sizeof(redbuf) ); //use the sys func

44 fread(redbuf , sizeof(redbuf) ,ffd+1 , stdin );

45 }

46 // fwrite(redbuf ,strlen(redbuf) , 1 , stdout );

47 // write(1 , redbuf , strlen(redbuf));

48 printf("buf = %s buf\_len = %d \n" , redbuf , strlen(redbuf));

49

50 return EXIT\_SUCCESS ;

51 }

[复制代码](javascript:void(0);)

makefile文件：

[复制代码](javascript:void(0);)

1 .SUFFIXES: .o.c

2 CC =gcc

3 SRC = Se\_keyboard.c

4 OBJ = $(SRC: .c =.o)

5 BIN = Se\_keyboard

6

7

8 .PHONY: start

9 start: $(OBJ)

10 $(CC) -o $(BIN) $(OBJ)

11 .o.c: $(SRC)

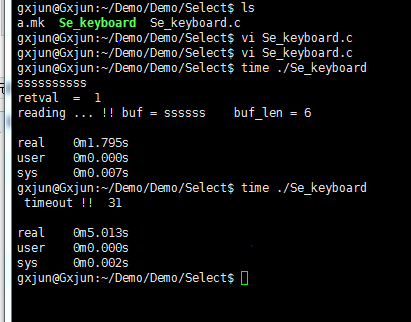
12 $(CC) -g -Wall $@ -c $<

13 .PHONY: clean

14 clean:

15 rm -f $(OBJ)

[复制代码](javascript:void(0);)



  虽然知道这么多，但是还是觉得Select并没有什么作用。

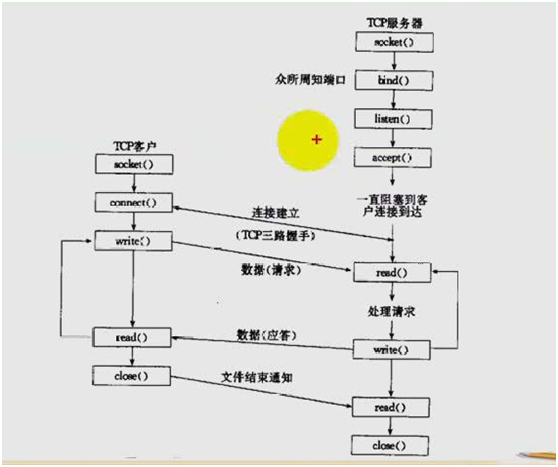
   Select一般是和Socket搭配使用，相当于线程池的效果，但是线程池有缺点，详情看这儿：

<http://blog.csdn.net/tianmohust/article/details/6677985>

<http://blog.csdn.net/xifeijian/article/details/17385831>

  关于Select的原理图：

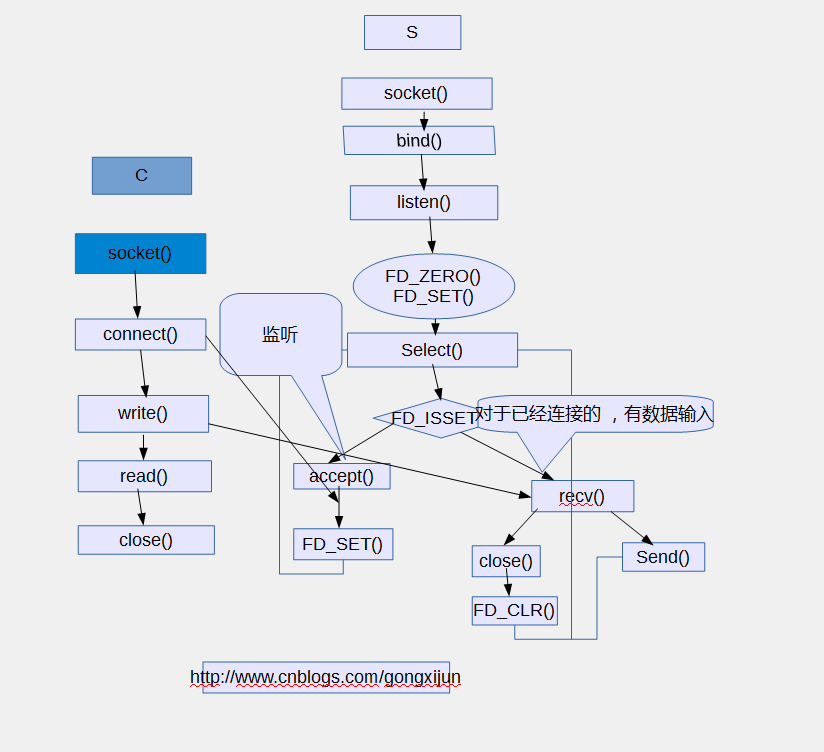
      首先来看下一对一的socket的c/s模式



 关于socket的一对一的详解

<http://www.cnblogs.com/gongxijun/p/4592864.html>

 看完这个之后，我们现在可以来看看这个图：



下面为举例：

          服务器创建一个socket并bind绑定一个本机地址和设定一个端口，然后进入listen监听状态。采用select模型而非传统apache模型（ppc）或者tpc模型 。 不过Select模型就是有这样一个特点

  一般我们default默认的SOMAXCONN为128 当然我们可以另外取一个设定（下面我们设定的是2048）作为最大连接数，虽然可以设置更大，但是缺点是，select模型是一个轮询模式，就是每一个都需要遍历一边所有的链接的fd

  查看是否在fd\_set集合中，这样，当SOMAXCONN取值非常大时，对于每一个客户端，访问时间都会延迟一点点，这样就是效率不是特别高！

 下面是一个简单的多路复用的网络并发Select模型

[复制代码](javascript:void(0);)

1 #include<stdio.h>

2 #include<string.h>

3 #include<stdlib.h>

4 #include<unistd.h>

5 #include<netinet/in.h>

6 #include<netinet/ip.h>

7 #include<sys/socket.h>

8 #include<sys/types.h>

9 #include<sys/time.h>

10 #include<arpa/inet.h>

11 #include<sys/select.h>

12 #include<assert.h>

13

14 #ifndef EXIT\_SUCCESS

15 #define EXIT\_SUCCESS 0

16 #endif

17

18 #ifndef EXIT\_FAILURE

19 #define EXIT\_FAILURE -1

20 #endif

21

22 #define Max\_connect 2048 //usually is SOMAXCONN =128 ,can be change

23

24 #define maxn 1024

25 #define Port 5567

26 #define Ser\_addr "192.168.132.128"

27

28 #define ERROR\_EXIT( inf ) \

29 do{ \

30 perror( inf ); \

31 printf("it happened in %d \n", \_\_LINE\_\_); \

32 exit(-1); \

33 }while(0);

34

35 #define Waring( inf ) \

36 do{ \

37 perror( inf ); \

38 printf("it happened in %d \n", \_\_LINE\_\_); \

39 }while(0);

40

41

42 int fds[Max\_connect];

43 int cnt = 1;

44

45 void

46 print (int fd, const char \*str)

47 {

48 assert (str != NULL);

49 printf ("the fd is %d \n", fd);

50 puts (str);

51 }

52

53 int

54 main (int argv, char \*argc[])

55 {

56

57 int ser\_sfd = -1, i;

58 struct sockaddr\_in ser\_addr;

59 struct sockaddr\_in client\_addr;

60 int setfd = 0, optval, maxsockfd;

61

62 char rebuf[maxn], wbuf[maxn];

63

64 // build a socket with ipv4 ans tcp

65

66 if ((ser\_sfd = socket (AF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0)

67 ERROR\_EXIT ("socket...!");

68

69 // set the type socket

70 /\*

71 level ={ SOL\_SOCKET ,IPPROTO\_TCP}

72 setsockopt is to cancle the jiangsi process

73 \*/

74

75 printf ("ser\_sfd = %d \n", ser\_sfd);

76 if (setsockopt (ser\_sfd, SOL\_SOCKET, SO\_REUSEADDR, &optval, sizeof (optval))

77 < 0)

78 ERROR\_EXIT ("setsockopt...!");

79

80 memset (&client\_addr, 0, sizeof (client\_addr));

81 memset (&ser\_addr, 0, sizeof (ser\_addr));

82 ser\_addr.sin\_family = AF\_INET;

83 ser\_addr.sin\_port = htons (Port);

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

85

86 if (bind (ser\_sfd, (struct sockaddr \*) &ser\_addr, sizeof (ser\_addr)) < 0)

87 ERROR\_EXIT ("bind..!");

88

89 if (listen (ser\_sfd, Max\_connect) < 0)

90 ERROR\_EXIT ("listen...!");

91

92 //user the select moduel

93 memset (fds, 0, sizeof (fds));

94 fd\_set fdset, wfd;

95 maxsockfd = ser\_sfd; //max socket fd

96

97 struct timeval tout;

98

99 tout.tv\_sec = 15;

100 tout.tv\_usec = 0;

101

102

103 while (1)

104 {

105

106 FD\_ZERO (&fdset); //clear

107 //FD\_ZERO (&wfd);

108

109 FD\_SET (ser\_sfd, &fdset); //bind

110 //FD\_SET (ser\_sfd, &wfd);

111

112 struct timeval tout;

113

114 tout.tv\_sec = 15;

115 tout.tv\_usec = 0;

116

117

118 for (i = 0; i < cnt; i++)

119 {

120 if (fds[i] != 0)

121 FD\_SET (fds[i], &fdset);

122 }

123

124 int tag = select (maxsockfd + 1, &fdset, NULL, NULL, &tout);

125

126 if (tag == 0)

127 {

128 Waring ("select wait timeout !");

129 continue;

130 }

131 else if (tag == -1)

132 ERROR\_EXIT ("Error select ...!");

133

134 //lunxun select

135 for (i = 0; i < cnt; i++)

136 {

137

138 if (FD\_ISSET (fds[i], &fdset))

139 {

140

141 int len = recv (fds[i], rebuf, sizeof (rebuf), 0);

142 if (len <= 0)

143 {

144

145 printf ("%d: \n", fds[i]);

146 close (fds[i]);

147 FD\_CLR (fds[i], &fdset);

148 Waring ("client is closed !");

149 continue;

150 }

151

152 printf ("the client\_ip : %s\n",

153 inet\_ntoa (client\_addr.sin\_addr));

154

155 print (fds[i], rebuf);

156

157 send (fds[i], rebuf, sizeof (rebuf), 0); //hui she

158 memset (rebuf, 0, sizeof (rebuf));

159 }

160 }

161 //if have a new connect happened

162 // memset(&client\_addr , 0 ,sizeof(client\_addr));

163 if (FD\_ISSET (ser\_sfd, &fdset))

164 {

165 // memset(&client\_addr , 0 ,sizeof(client\_addr));

166 int acplen = sizeof (client\_addr);

167 int acp = accept (ser\_sfd, (struct sockaddr \*) &client\_addr,

168 &acplen);

169

170 printf ("accept return acp=%d \n", acp);

171

172 if (acp < 0)

173 {

174

175 Waring ("waring accept acp<=0!");

176 continue;

177 }

178 //add to arr

179 if (cnt < maxn)

180 fds[cnt++] = acp;

181 else

182 {

183 ERROR\_EXIT ("cnt>maxn");

184 }

185 if (acp > maxsockfd)

186 maxsockfd = acp;

187 }

188 }

189

190 for (i = 0; i < cnt; i++)

191 {

192 close (fds[i]);

193 }

194

195 return EXIT\_SUCCESS;

196 }

[复制代码](javascript:void(0);)

makefile文件：

[复制代码](javascript:void(0);)

1 .SUFFIXES: .o.c

2 CC =gcc

3 SRC = server.c

4 OBJ = $(SRC: .c =.o)

5 BIN = Sez\_Server

6

7

8 .PHONY: start

9 start: $(OBJ)

10 $(CC) -o $(BIN) $(OBJ)

11 .o.c: $(SRC)

12 $(CC) -g -Wall $@ -c $<

13 .PHONY: clean

14 clean:

15 rm -f $(OBJ)

[复制代码](javascript:void(0);)

客户端：

[复制代码](javascript:void(0);)

1 #include<stdio.h>

2 #include<string.h>

3 #include<stdlib.h>

4 #include<netinet/in.h>

5 #include<arpa/inet.h>

6 #include<sys/socket.h>

7 #include<sys/types.h>

8 #include<assert.h>

9 #include<unistd.h>

10 #ifndef EXIT\_FAILURE

11 #define EXIT\_FAILURE -1 //exit failure

12 #endif

13 #ifndef EXIT\_SUCCESS

14 #define EXIT\_SUCCESS 0 // exit sucessful

15 #endif

16

17 #define Port 5567

18 #define IPADDR "192.168.132.128"

19 #define maxn 1024

20

21 #define ERROR\_EXIT( inf ) \

22 do{ \

23 perror( inf ); \

24 printf("it's happened in %d \n",\_\_LINE\_\_); \

25 }while(0) ;

26

27

28 //use the function to connect the server !!

29

30

31 int

32 main (int argv, char \*argc[])

33 {

34

35 int sfd = -1;

36 char rbuf[maxn], wbuf[maxn];

37 // int sfd=-1 ; //socket\_fd

38 int confd = -1; //connect\_fd

39 struct sockaddr\_in soaddr;

40 if ((sfd = socket (AF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0)

41 {

42 ERROR\_EXIT ("socket");

43 return EXIT\_FAILURE;

44 }

45

46 //memset a struct

47 memset (&soaddr, 0, sizeof (soaddr));

48

49 //int the struct sockaddr\_in

50

51 soaddr.sin\_family = AF\_INET;

52 soaddr.sin\_port = htons (Port);

53 soaddr.sin\_addr.s\_addr = inet\_addr (IPADDR);

54

55 if ((confd =

56 connect (sfd, (struct sockadrr \*) &soaddr, sizeof (soaddr))) < 0)

57 {

58 ERROR\_EXIT ("connect");

59 return EXIT\_FAILURE;

60 }

61

62 printf ("connect is sucessful !\n");

63

64 while (fgets (wbuf, sizeof (rbuf), stdin) != NULL)

65 {

66 write (sfd, wbuf, strlen (wbuf));

67 read (sfd, rbuf, sizeof (rbuf));

68 fputs (rbuf, stdout);

69 }

70

71 close (sfd);

72 close (confd);

73 return EXIT\_SUCCESS;

74 }

[复制代码](javascript:void(0);)

makefile文件：



[复制代码](javascript:void(0);)

1 .SUFFIXES: .o.c

2 CC =gcc

3 SRC = client.c

4 OBJ = $(SRC: .c =.o)

5 BIN = Se\_client

6

7

8 .PHONY: start

9 start: $(OBJ)

10 $(CC) -o $(BIN) $(OBJ)

11 .o.c: $(SRC)

12 $(CC) -g -Wall $@ -c $<

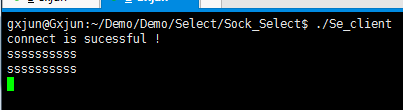
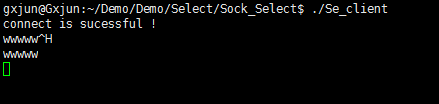
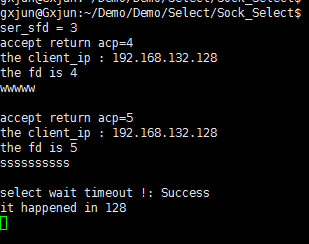
13 .PHONY: clean

14 clean:

15 rm -f $(OBJ)

[复制代码](javascript:void(0);)

效果图：

1. 

编程是一种快乐，享受代码带给我的乐趣！！！