http://blog.chinaunix.net/uid-8196371-id-1642035.html

linux网络编程：用C语言实现的聊天程序（异步通信） 2011-07-08 17:01:02

分类： LINUX

本片文章，在上一篇：linux网络编程：用C语言实现的聊天程序（同步通信） 的基础上，增加了IO复用的功能，实现了聊天程序的异步通讯！

1、使用IO复用可以在等待的时候加入了超时的时间，如果等待的时间没有达到超时时间，那么该情况与阻塞的情况一致。而当超时的时间到达时，仍没有数据接收到，系统回返回0，不再等待。select函数就实现了这个功能。

2、select函数原型

int select (int maxfdp1, fd\_set \*readset, fd\_set \*writeset, fd\_set \*exceptset, const struct timeval \* timeout);

a）返回值

>0：就绪的描述符

-1：出错

0 ：超时

struct timeval{

long tv\_sec; // seconds

long tv\_usec; // microseconds

}

b）具体解释select的参数

int maxfdp是一个整数值，是指集合中所有文件描述符的范围，即所有文件描述符的最大值加1；

fd\_set\*readfds 文件描述符集合内，是否有数据可读；

fd\_set\*writefds 文件描述符集合内，是否有数据可写；

fd\_set \*errorfds 文件描述符集合内，是否有文件发生错误；

struct timeval \*timeout是select的超时时间，它可以使select处于三种状态，第一，若将NULL以形参传入，就是将select置于阻塞状态；第二，若将时间值设为0秒0毫秒，就变成一个纯粹的非阻塞函数，不管文件描述符是否有变化，都立刻返回继续执行，文件无变化返回0，有变化返回一个正值；第三，timeout的值大于0，这就是等待的超时时间，即select在timeout时间内阻塞。

c）4个宏可以操作文件描述符的集合

void FD\_ZERO (fd\_set \*fdset); // 初始化文件描述集合

void FD\_SET (int fd, fd\_set \*fdset); // 将描述符加入到集合中

void FD\_CLR (int fd, fd\_set \*fdset); // 将描述符中集合中删除

int FD\_ISSET (int fd, fd\_set \*fdset); // 检查描述符集合中指定的文件描述符是否可读写

3、客户端源代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/time.h>

#define BUFLEN 1024

int main(int argc, char \*\*argv)

{

int sockfd;

struct sockaddr\_in s\_addr;

socklen\_t len;

unsigned int port;

char buf[BUFLEN];

fd\_set rfds;

struct timeval tv;

int retval, maxfd;

/\*建立socket\*/

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1){

perror("socket");

exit(errno);

}else

printf("socket create success!\n");

/\*设置服务器端口\*/

if(argv[2])

port = atoi(argv[2]);

else

port = 4567;

/\*设置服务器ip\*/

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

s\_addr.sin\_family = AF\_INET;

s\_addr.sin\_port = htons(port);

if (inet\_aton(argv[1], (struct in\_addr \*)&s\_addr.sin\_addr.s\_addr) == 0) {

perror(argv[1]);

exit(errno);

}

/\*开始连接服务器\*/

if(connect(sockfd,(struct sockaddr\*)&s\_addr,sizeof(struct sockaddr)) == -1){

perror("connect");

exit(errno);

}else

printf("conncet success!\n");

while(1){

/\*把可读文件描述符的集合清空\*/

FD\_ZERO(&rfds);

/\*把标准输入的文件描述符加入到集合中\*/

FD\_SET(0, &rfds);

maxfd = 0;

/\*把当前连接的文件描述符加入到集合中\*/

FD\_SET(sockfd, &rfds);

/\*找出文件描述符集合中最大的文件描述符\*/

if(maxfd < sockfd)

maxfd = sockfd;

/\*设置超时时间\*/

tv.tv\_sec = 5;

tv.tv\_usec = 0;

/\*等待聊天\*/

retval = select(maxfd+1, &rfds, NULL, NULL, &tv);

if(retval == -1){

printf("select出错，客户端程序退出\n");

break;

}else if(retval == 0){

printf("客户端没有任何输入信息，并且服务器也没有信息到来，waiting...\n");

continue;

}else{

/\*服务器发来了消息\*/

if(FD\_ISSET(sockfd,&rfds)){

/\*\*\*\*\*\*接收消息\*\*\*\*\*\*\*/

bzero(buf,BUFLEN);

len = recv(sockfd,buf,BUFLEN,0);

if(len > 0)

printf("服务器发来的消息是：%s,共有字节数是: %d\n",buf,len);

else{

if(len < 0 )

printf("接受消息失败！\n");

else

printf("服务器退出了，聊天终止！\n");

break;

}

}

/\*用户输入信息了,开始处理信息并发送\*/

if(FD\_ISSET(0, &rfds)){

\_retry:

/\*\*\*\*\*\*发送消息\*\*\*\*\*\*\*/

bzero(buf,BUFLEN);

/\*fgets函数：从流中读取BUFLEN-1个字符\*/

fgets(buf,BUFLEN,stdin);

/\*打印发送的消息\*/

//fputs(buf,stdout);

if(!strncasecmp(buf,"quit",4)){

printf("client 请求终止聊天!\n");

break;

}

/\*如果输入的字符串只有"\n"，即回车，那么请重新输入\*/

if(!strncmp(buf,"\n",1)){

printf("输入的字符只有回车，这个是不正确的！！！\n");

goto \_retry;

}

/\*如果buf中含有'\n'，那么要用strlen(buf)-1，去掉'\n'\*/

if(strchr(buf,'\n'))

len = send(sockfd,buf,strlen(buf)-1,0);

/\*如果buf中没有'\n'，则用buf的真正长度strlen(buf)\*/

else

len = send(sockfd,buf,strlen(buf),0);

if(len > 0)

printf("\t消息发送成功，本次共发送的字节数是：%d\n",len);

else{

printf("消息发送失败!\n");

break;

}

}

}

}

/\*关闭连接\*/

close(sockfd);

return 0;

}

4、服务器源代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/time.h>

#define BUFLEN 1024

int main(int argc, char \*\*argv)

{

int sockfd, newfd;

struct sockaddr\_in s\_addr, c\_addr;

char buf[BUFLEN];

socklen\_t len;

unsigned int port, listnum;

fd\_set rfds;

struct timeval tv;

int retval,maxfd;

/\*建立socket\*/

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1){

perror("socket");

exit(errno);

}else

printf("socket create success!\n");

/\*设置服务器端口\*/

if(argv[2])

port = atoi(argv[2]);

else

port = 4567;

/\*设置侦听队列长度\*/

if(argv[3])

listnum = atoi(argv[3]);

else

listnum = 3;

/\*设置服务器ip\*/

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

s\_addr.sin\_family = AF\_INET;

s\_addr.sin\_port = htons(port);

if(argv[1])

s\_addr.sin\_addr.s\_addr = inet\_addr(argv[1]);

else

s\_addr.sin\_addr.s\_addr = INADDR\_ANY;

/\*把地址和端口帮定到套接字上\*/

if((bind(sockfd, (struct sockaddr\*) &s\_addr,sizeof(struct sockaddr))) == -1){

perror("bind");

exit(errno);

}else

printf("bind success!\n");

/\*侦听本地端口\*/

if(listen(sockfd,listnum) == -1){

perror("listen");

exit(errno);

}else

printf("the server is listening!\n");

while(1){

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*聊天开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

len = sizeof(struct sockaddr);

if((newfd = accept(sockfd,(struct sockaddr\*) &c\_addr, &len)) == -1){

perror("accept");

exit(errno);

}else

printf("正在与您聊天的客户端是：%s: %d\n",inet\_ntoa(c\_addr.sin\_addr),ntohs(c\_addr.sin\_port));

while(1){

/\*把可读文件描述符的集合清空\*/

FD\_ZERO(&rfds);

/\*把标准输入的文件描述符加入到集合中\*/

FD\_SET(0, &rfds);

maxfd = 0;

/\*把当前连接的文件描述符加入到集合中\*/

FD\_SET(newfd, &rfds);

/\*找出文件描述符集合中最大的文件描述符\*/

if(maxfd < newfd)

maxfd = newfd;

/\*设置超时时间\*/

tv.tv\_sec = 5;

tv.tv\_usec = 0;

/\*等待聊天\*/

retval = select(maxfd+1, &rfds, NULL, NULL, &tv);

if(retval == -1){

printf("select出错，与该客户端连接的程序将退出\n");

break;

}else if(retval == 0){

printf("服务器没有任何输入信息，并且客户端也没有信息到来，waiting...\n");

continue;

}else{

/\*用户输入信息了,开始处理信息并发送\*/

if(FD\_ISSET(0, &rfds)){

\_retry:

/\*\*\*\*\*\*发送消息\*\*\*\*\*\*\*/

bzero(buf,BUFLEN);

/\*fgets函数：从流中读取BUFLEN-1个字符\*/

fgets(buf,BUFLEN,stdin);

/\*打印发送的消息\*/

//fputs(buf,stdout);

if(!strncasecmp(buf,"quit",4)){

printf("server 请求终止聊天!\n");

break;

}

/\*如果输入的字符串只有"\n"，即回车，那么请重新输入\*/

if(!strncmp(buf,"\n",1)){

printf("输入的字符只有回车，这个是不正确的！！！\n");

goto \_retry;

}

/\*如果buf中含有'\n'，那么要用strlen(buf)-1，去掉'\n'\*/

if(strchr(buf,'\n'))

len = send(newfd,buf,strlen(buf)-1,0);

/\*如果buf中没有'\n'，则用buf的真正长度strlen(buf)\*/

else

len = send(newfd,buf,strlen(buf),0);

if(len > 0)

printf("\t消息发送成功，本次共发送的字节数是：%d\n",len);

else{

printf("消息发送失败!\n");

break;

}

}

/\*客户端发来了消息\*/

if(FD\_ISSET(newfd, &rfds)){

/\*\*\*\*\*\*接收消息\*\*\*\*\*\*\*/

bzero(buf,BUFLEN);

len = recv(newfd,buf,BUFLEN,0);

if(len > 0)

printf("客户端发来的信息是：%s,共有字节数是: %d\n",buf,len);

else{

if(len < 0 )

printf("接受消息失败！\n");

else

printf("客户端退出了，聊天终止！\n");

break;

}

}

}

}

/\*关闭聊天的套接字\*/

close(newfd);

/\*是否退出服务器\*/

printf("服务器是否退出程序：y->是；n->否? ");

bzero(buf, BUFLEN);

fgets(buf,BUFLEN, stdin);

if(!strncasecmp(buf,"y",1)){

printf("server 退出!\n");

break;

}

}

/\*关闭服务器的套接字\*/

close(sockfd);

return 0;

}

5、编译源代码：

new@new-desktop:~/linux/c$ gcc -Wall async-server.c -o async-server

new@new-desktop:~/linux/c$ gcc -Wall async-client.c -o async-client

6、服务器运行：

new@new-desktop:~/linux/c$ ./async-server 127.0.0.1

socket create success!

bind success!

the server is listening!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*聊天开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正在与您聊天的客户端是：127.0.0.1: 37648

服务器没有任何输入信息，并且客户端也没有信息到来，waiting...

hello

消息发送成功，本次共发送的字节数是：5

服务器没有任何输入信息，并且客户端也没有信息到来，waiting...

客户端发来的信息是：renwen,共有字节数是: 6

7、客户端运行：

new@new-desktop:~/linux/c$ ./async-client 127.0.0.1

socket create success!

conncet success!

客户端没有任何输入信息，并且服务器也没有信息到来，waiting...

服务器发来的消息是：hello,共有字节数是: 5

客户端没有任何输入信息，并且服务器也没有信息到来，waiting...

renwen

消息发送成功，本次共发送的字节数是：6

客户端没有任何输入信息，并且服务器也没有信息到来，waiting...

客户端没有任何输入信息，并且服务器也没有信息到来，waiting...

客户端没有任何输入信息，并且服务器也没有信息到来，waiting...