**Linux操作系统大作业报告**

1. 简介：

本设计主要实现一个Linux下的局域网聊天工具的设计。该设计主要分为两部分，客户端部分和服务器部分。设计并实现一个简单的聊天室程序，可以是终端字符界面，支持用户管理，用户名密码注册和登录，客户端登陆到服务器聊天后，可以在聊天室内与其他用户交流(这个聊天室中的任何一-个用户输入一段字符后，室内的其他用户都可以看到这句话)。据此，聊天程序分为客户端和服务器端。客户端对应每一个参加聊天的用户，完成从终端.上输入采集并传递到服务器端和从服务器端接收信息输出显示的功能。

1. 涉及代码：

**1.首先是初始化服务器，使服务器进入监听状态:**  
sockfd = socket (AF\_ INET， SOCK\_ STREAM, 0) ;  
//首先建立一-个socket,族为AF\_ INET,类型为SOCK\_ STREAM。  
// AF\_ \_INET = ARPA Internet protocols，即使用TCP/IP协议族。// socK\_ STREAM 类型提供了顺序的，可靠的，基于字节流的全双工连接。//由干该协议族中只有一个协议，因此第三个参数为0。  
bind(sockfd，(struct sockaddr \*) &servaddr,  
sizeof (serv\_ \_addr) ) ;  
//再将这个socket与某个地址进行绑定。

// serv\_ addr包括sin\_ family=AF\_ INET协议族同socket。  
// sin\_ addr.s\_ \_addr = htonl (INADDR\_ ANY) 服务器所接收的所有其他地址//请求建立的连接。  
// sin\_ port = htons (SERV\_ TCP\_ PORT)服务器所监听的端口。  
listen ( socket，MAX\_ CLIENT) ;  
//地址绑定后，服务器进入监听状态。  
// MAX\_ CLIENT是可以同时建立连接的客户总数。  
服务器进入监听状态后，等待客户建立连接。若客户需要连接，也需要先进行网络部分的初始化工作:  
sockfd = socket (AF\_ INET，SOCK\_ STREAM, 0) ;  
//与服务器端相同。  
connect (sockfd，(struct sockaddr \*)  
&serv\_ addr, sizeof (serv\_ addr) )  
//客户使用connect建立一-个连接。  
// serv\_ addr中的变量分别设置为:  
// sin\_ family = AF\_ INET 协议族同socket  
// sin\_ \_addr.s\_ addr = inet\_ addr (SERV\_ HOST\_ ADDR)  
//地址为server 所在的计算机地址，本程序中为127.0.0.1  
// sin\_ port = htons (SERV\_ TCP\_ \_PORT) 端口为服务器监听的端口。

当客户建立新连接的时候，服务器使用accept来接收该连接:  
accept (sockfd, (struct sockaddr \*) &cli\_ addr, &cli\_ \_len) ;//在函数返回时，cli\_ addr 中保留的是该连接对方的信息  
//包括IP地址和对方使用的端口。  
// accept返回一个新的文件描述符。  
在服务器进入监听状态以后，使用select 方法实现，该方法中的所有描述符都是阻塞的。使用select 判断一组文件描述符中时候有一个可读(写)，如果没有就阻塞。直到有一个的时候被唤醒。

**2.客户端实现:**  
由于要处理两个文件描述符，因此需要判断是否有可读写的文件描述符需要加入两项:  
FD\_ ZERO (sockset) ;  
//将sockset清空。  
FD\_ SET (sockfd, sockset) ;  
//把sockfd加入到sockset集合中。  
FD\_ SET(0， sockset) ;  
//把标准输入(0)加入到sockset集合中。  
**然后客户处理如下:**  
while (不想退出) {  
select (sockfd+1, &sockset, NULL, NULL, NULL) ;  
//此时该函数使进程阻塞，直到标准输入或者sockfd 中有一个可读为//第一个参数是0和sockfd中的最大值加1  
//第二个参数是读集合，也就是sockset  
//第三、四个参数是写集，在本程序中都为空  
//第五个参数为超时时间，即在指定的时间仍然没有可读，则报错并返  
当该参数被设置为NULL时，超时时间为无限长。  
//当select因为可读返回时，sockset中包含的只是  
//可读的那些文件描述符。  
if (FD\_ ISSET (sockfd, &sockset) ) {  
  
// FD\_ IssET这个宏判断sockfd是否属于可读的文件描述符  
/\*  
\*从sockfd中读入，输出到标准输出.上去。\*/  
}  
if (FD\_ ISSET(0， &sockset) ) {/\*  
\*从标准读入中读入，输出到sockfd上去。\*/  
}

/\*

\*重新设置sockset。(即: 将sockset清空，并将sockfd和0加入)\*/  
下面是服务器的情况:设置sockset如下:FD\_ ZREO (sockset) ;  
FD\_ SET (sockfd, sockset) ;  
for (所有有效连接) {  
FD\_ SET (userfd[i]，sockset) ;  
max =最爱文件描述符+1;  
服务器处理如下:  
while(1) {  
select (maxfd, &sockset, NULL, NULL, NULL) ;if (FD\_ ISSET (sockfd, &sockset)) {  
//有新连接  
/\*  
\*建立新连接，并将该描述符加入到sockset中\*/  
for (所有有效连接) {  
if (FD\_ IsSET (userfd[i], sockset)) {//该连接中有字符可读  
/\*  
\*从该连接中读取字符，并发送给其他有效连接\*/}  
}  
/\*  
\*重新设置sockset;  
\*/  
读写文件函数如下:  
read (userfd[i]，line, MAX\_ LINE) ;  
// userfd[i] 是指第i个用户连接的文件描述符// line是指读出的字符存放位置  
// MAX\_ LINE 是一-次最多读出的字符数  
//返回值是实际读出的字符数  
write (userfd[i], line, strlen(line) ) ;// userfd[i]是指第i个用户连接的文件描述符//line是指要发送的字符串.  
// strlen(line) 是要发送的字符串长度//返回值是实际发送的字符数

1. 部分实现代码

**（1）客户端代码**

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>  
#define TRUE 1  
#define PORT 1000  
static int sockfd;  
void recvfromserver（)//接受服务器消息线程入口函数  
{  
char mes[ 1024];int nbytes=0;  
while(1)  
{  
memset(mes,0,sizeof(mes));  
nbytes=read(sockfd,mes,sizeof(mes));

if(nbytes>0)

{mes[nbytes]="0';

printf("%sln" ,mes);}

}  
pthread\_ exit(NULL);  
}

int main(int argc, char \*argv[])  
{  
// int sockfd;  
char buffer[ 1024];

struct sockaddr\_ in server\_ addr;

struct hostent \*host;  
int portnumber,nbytes;char \*strhost=" 127.0.0.1";  
char clientname[20];

char password[20];

char mes[ 1024];

Int thr\_ id;  
/\* thread ID for the newly created thread \*/  
pthread\_ \_t p. \_thread;/\* thread's structure\*/  
  
if(argc!=1)  
{  
fnitstsed."0sge:%s 'aln' ,rgv[0);

exit(1);}  
if(hosgethostbyamestrbos)ULL)  
{  
frintf(terte,"Gethostname ertn");

exit(1);

}

/\*客户程序开始建立sockfd 描述符\*/ .  
printf("Creating the Set inferface..n");  
if(sockfd=socket(AF\_ INET,SOCK\_ STREAM,0))==-1)  
{  
fprintf(stderr,"Socket Error:% s\a\n" ,strerror(errno));

exit(1);}  
  
/\*客户程序填充服务端的资料\*/  
bzero( &server\_ \_addr, sizeof(server\_ \_add));  
server\_ addr.sin\_ family=AF\_ INET;  
server\_ \_addr.sin\_ port=htons(PORT);  
server\_ \_addr .sin\_ addr=\*((struct in\_ \_addr \*)host->h\_ addr);  
printf("The successful landing\nWelcome to zhe chat room!\n");/\*客户程序发起连接请求\*/

if(connect(sockfd,(structsockaddr\*)(&server\_ addr),sizeof(structsockaddr))==-1)  
{  
fprintf(stderr, "Connect Error:%slaln" ,strerror(errno));  
exit(1);  
}  
/\*连接成功了\*/  
  
printf("The successful landinglnWelcome to zhe chat room!n");printf("Please enter your nickname:In");  
scanf("%s",clientname);  
printf("Please enter your password:1n");  
scanf("%s",password);  
printf("lnThe successful landing");  
//  
write(sockfd,clientname,sizeof(clientname));  
printf("lnNow you can chat with others!€" "Quit"CUT DOWN LANDINGInln");  
  
thr\_ \_id = pthread\_ \_create(&p\_ thread, NULL, recvfromserver, NULL);

while(1)

{

memset(buffer,0,sizeof(buffer));  
memset( mes,0,sizeof(mes));  
scanf("%s",buffer);  
strcat(mes,clientname);  
strcat(mes,":");  
strcat( mes,buffer);  
  
//printf(" main thread %sln",mes);  
if((write(sockfd,mes,sizeof(mes)))==-1)  
{  
fprintf(stderr,"Write Error:%sln",strerror(errno));

exit(1);  
}  
if(strcmp(buffer,"Quit")==0)

{break;}  
}

}  
/\*结束通讯\*/

close(sockfd);

exit(0);  
  
**(2服务器代码**

#include <stdlib.h>

#include <stdio.h>

#include <ermo.h>

#include <string.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>

#define MAXLINE 1000 //在一条消息中最大的输出字符数

#define LISTENQ 20//最大监听队列  
#define PORT 1000//监听端口.

#define MAXFD 20//最大的在线用户数  
void\*get\_ cien(void \*):  
int sockfd,i;  
static int maxi=0;//maxi表示当前ciou 数组中最大的用户的i值

static int client[MAXFDI];

void revandsend(void)//监听转发线程入口函数  
{  
int index=0;

int nbytes=0;

char bffrt1024];

int len;  
int outindex=0;

while(1)

{if(maxi>0)

{  
  
memset(buffer,0,sizeof(buer));  
nbytes=0;  
//index++;  
nbytes=read(client[index++ ],buffer,sizeof(buffer));

// prinf("%d,%dln",index,client[index]);  
if(nbytes>0)  
{  
buffer[nbytes]='0';  
printf(" %sln",buffer);  
outindex=0;  
while(outindex<maxi)  
{if(write(client[outindex++],buffer,sizeof(buffer))==-1)  
{  
fprintf(stderr, "Write Error:% sln" ,strerror(errno));

exit(1);  
}  
}}  
if(index>=maxi)

index=0;  
}  
pthread\_ exit(NULL);

}

int main(int argc, char \*argv[])  
{  
//int client\_ \_fd[LISTENQ],clientnum=0;;  
struct sockaddr\_ in server\_ addr;  
struct sockaddr\_ in client\_ \_addr;  
int sin\_ size,portnumber;  
char hello[]="Hello! Are You Fine?n";  
intthr\_ id;  
/\* thread ID for the newly created thread \*/  
pthread\_ t P\_ thread; ./\* . thread's structure\*/  
int new\_ fd=0;  
memset(client,0,sizeof(clien));  
if(argc!=1)  
{  
fprintf(sterr,"Usage:%s portnumberlaln" ,argv[0]);

exit(1);}  
/\*服务器端开始建立socket 描述符\*/  
if(sockfd=socket(AF\_ INET ,SOCK\_ STREAM,0))==-1)  
{  
fprintf(stderr,"Socket error:% s\n\a",strerror(errmo));

exit(1);}  
  
/\*服务器端填充sockaddr结构\*/  
bzero(& server\_ addr,sizeof(struct sockaddr\_ \_in));  
server\_ \_addr.sin\_ family=AF\_ INET;  
server\_ \_addr .sin\_ \_addr.s\_ \_addr=htonl(INADDR\_ ANY);  
server\_ addr.sin\_ port=htons(PORT);  
/\*捆绑sockfd描述符\*/  
if(bind(sockfd,(struct sockaddr \*)(&server\_ \_addr),sizeof(struct sockadd))==-1)  
{  
fprintf(stderr,"Bind error:%s\n\a" ,strerror(errno));

exit(1);

}

printf("服务器监听端口%d..\n" ,PORT);/\*监听. sockfd描述符\*/  
if(listen(sockfd,LISTENQ)==-1)  
{  
fprintf(stderr,"Listen error:% s\na,",strerror(ermo));

exit(1);}  
thr\_ id = pthread\_ \_create(&p\_ \_thread, NULL, reevandsend, NULL);

printf("NAME:Li Junnan No:80006 12030 Class:Ji ruan121\n");

printf("Welcome to the chat room!!!n");

while(1)  
{  
/\*服务器阻塞,直到客户程序建立连接\*/

if(maxi>=20)

{  
printf("Over the max peopleln");

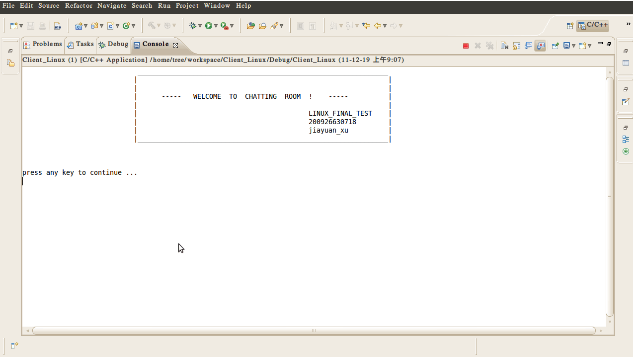
continue;  
}  
sin\_ \_size=sizeof(struct sockaddr\_ in);

if((new\_ fd=accept(sockfd,(struct\*)(&client\_ \_addr),&sin\_ size))==-1)  
{  
fprintf(stderr," 'Accept error: %slnla" ,strerror(errno));

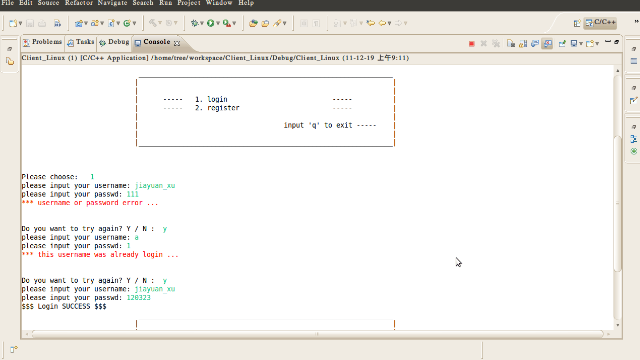
exit(1);}  
/\*fprintf(stderr,"Server get from %sln",inet\_ ntoa(client\_ \_addr.sin\_ \_addr));\*/  
client[maxi++]=new\_ fd;  
printf("lnNew %d user come to the chat roomln" ,new\_ fd-3);}  
close(sockfd);  
  
exit(O);

}

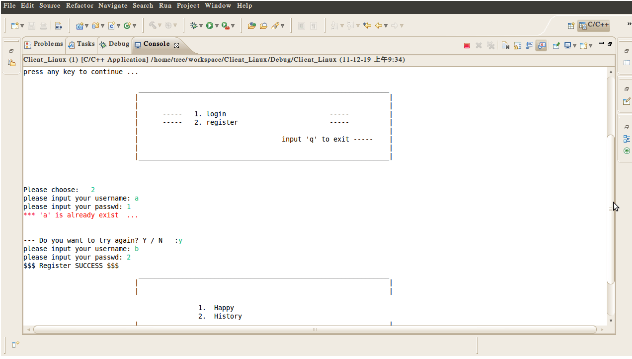
1. 运行结果：



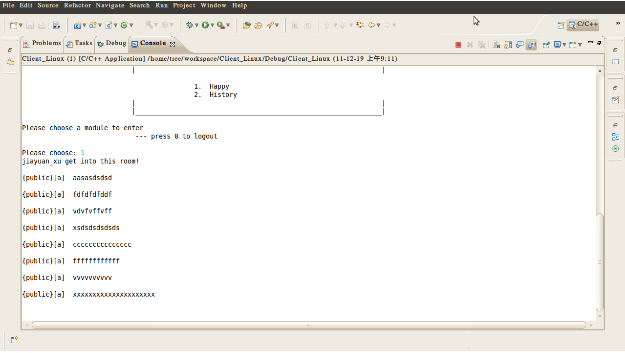
登录：



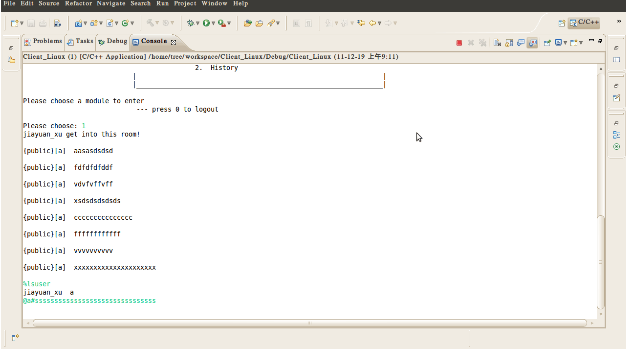
注册：

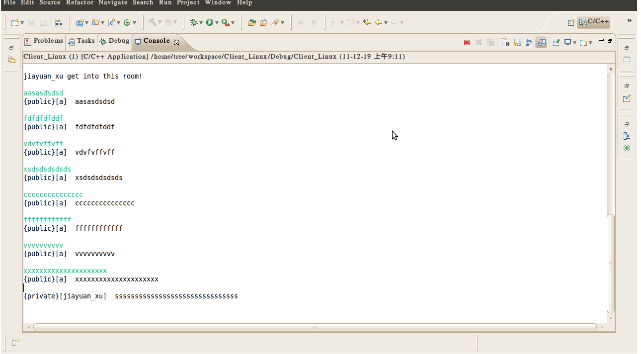


群聊：



私聊：





1. 项目总结：

借鉴网络以及同学的帮助下安装系统并成功运行出结果，但经过此项目，对Linux操作系统有了新的认知。