# 第四次上机作业

姓名：蒋世豪 学号：15051325 专业：计算机科学与技术 班级：15052313

一、实验内容：

学习并了解UDP的相关概念。完成一个聊天室客户端服务器程序，在例程的基础上加入私聊功能。

二、实验文件：

Udpmulclt.c

Udpmulsrv.c

三、实验结果：

3.1程序的运行

首先，运行服务器.

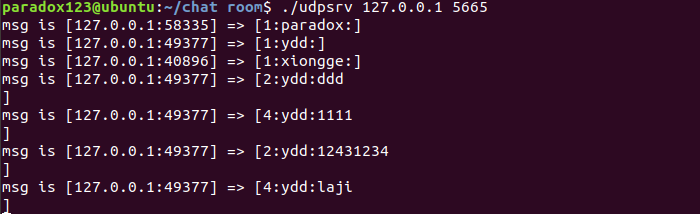


图1 服务器端打开的截图

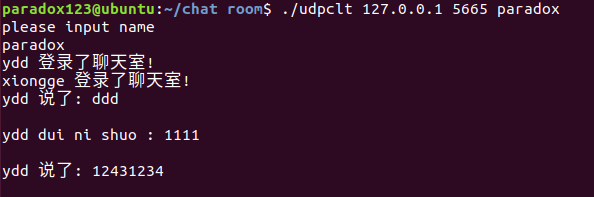


图2 Paradox客户端的截图,其中看到YDD的私聊和广播对话

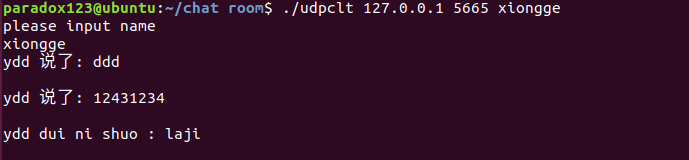


图3 Xiongge客户端的截图,其中看到YDD的私聊和广播对话

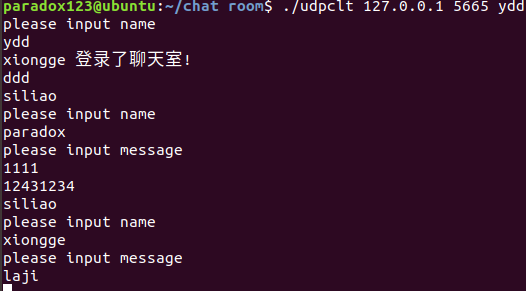


图4 ydd客户端，输入私聊对象以及对话。

四、实验源代码：

4.1服务器代码

文件 udpsrv.c

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/wait.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/socket.h>

// 名字长度包含'\0'

#define \_INT\_NAME (64)

// 报文最大长度,包含'\0'

#define \_INT\_TEXT (512)

//4.0 控制台打印错误信息, fmt必须是双引号括起来的宏

#define CERR(fmt, ...) \

fprintf(stderr,"[%s:%s:%d][error %d:%s]" fmt "\r\n",\

\_\_FILE\_\_, \_\_func\_\_, \_\_LINE\_\_, errno, strerror(errno),##\_\_VA\_ARGS\_\_)

//4.1 控制台打印错误信息并退出, t同样fmt必须是 ""括起来的字符串常量

#define CERR\_EXIT(fmt,...) \

CERR(fmt,##\_\_VA\_ARGS\_\_),exit(EXIT\_FAILURE)

/\*

\* 简单的Linux上API错误判断检测宏, 好用值得使用

\*/

#define IF\_CHECK(code) \

if((code) < 0) \

CERR\_EXIT(#code)

// 发送和接收的信息体

struct umsg{

char type; //协议 '1' => 向服务器发送名字, '2' => 向服务器发送信息, '3' => 向服务器发送退出信息

char name[\_INT\_NAME]; //保存用户名字

char text[\_INT\_TEXT]; //得到文本信息,空间换时间

char siliao[\_INT\_NAME];

};

// 维护一个客户端链表信息,记录登录信息

typedef struct ucnode {

struct sockaddr\_in addr;

struct ucnode\* next;

} \*ucnode\_t ;

// 新建一个结点对象

static inline ucnode\_t \_new\_ucnode(struct sockaddr\_in\* pa){

ucnode\_t node = calloc(sizeof(struct ucnode), 1);

if(NULL == node)

CERR\_EXIT("calloc sizeof struct ucnode is error. ");

node->addr = \*pa;

return node;

}

// 插入数据,这里head默认头结点是当前服务器结点

static inline void \_insert\_ucnode(ucnode\_t head, struct sockaddr\_in\* pa) {

ucnode\_t node = \_new\_ucnode(pa);

node->next = head->next;

head->next = node;

}

// 这里是有用户登录处理

static void \_login\_ucnode(ucnode\_t head, int sd, struct sockaddr\_in\* pa, struct umsg\* msg) {

\_insert\_ucnode(head, pa);

head = head->next;

// 从此之后才为以前的链表

while(head->next){

head = head->next;

IF\_CHECK(sendto(sd, msg, sizeof(\*msg), 0, (struct sockaddr\*)&head->addr, sizeof(struct sockaddr\_in)));

}

}

// 信息广播

static void \_broadcast\_ucnode(ucnode\_t head, int sd, struct sockaddr\_in\* pa, struct umsg\* msg) {

int flag = 0; //1表示已经找到了

while(head->next) {

head = head->next;

if((flag) || !(flag=memcmp(pa, &head->addr, sizeof(struct sockaddr\_in))==0)){

IF\_CHECK(sendto(sd, msg, sizeof(\*msg), 0, (struct sockaddr\*)&head->addr, sizeof(struct sockaddr\_in)));

}

}

}

// 有人退出群聊

static void \_quit\_ucnode(ucnode\_t head, int sd, struct sockaddr\_in\* pa, struct umsg\* msg) {

int flag = 0;//1表示已经找到

while(head->next) {

if((flag) || !(flag = memcmp(pa, &head->next->addr, sizeof(struct sockaddr\_in))==0)){

IF\_CHECK(sendto(sd, msg, sizeof(\*msg), 0, (struct sockaddr\*)&head->next->addr, sizeof(struct sockaddr\_in)));

head = head->next;

}

else { //删除这个退出的用户

ucnode\_t tmp = head->next;

head->next = tmp->next;

free(tmp);

}

}

}

// 销毁维护的对象池,没有往复杂的考虑了简单处理退出了

static void \_destroy\_ucnode(ucnode\_t\* phead) {

ucnode\_t head;

if((!phead) || !(head=\*phead)) return;

while(head){

ucnode\_t tmp = head->next;

free(head);

head = tmp;

}

\*phead = NULL;

}

/\*

\* udp聊天室的服务器, 子进程广播信息,父进程接受信息

\*/

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

int sd, rt;

struct sockaddr\_in addr = { AF\_INET };

socklen\_t alen = sizeof addr;

struct umsg msg;

ucnode\_t head;

// 这里简单检测

if(argc != 3) {

fprintf(stderr, "uage : %s [ip] [port]\n", argv[0]);

exit(-1);

}

// 下面对接数据

if((rt = atoi(argv[2]))<1024 || rt > 65535)

CERR("atoi port = %s is error!", argv[2]);

// 接着判断ip数据

IF\_CHECK(inet\_aton(argv[1], &addr.sin\_addr));

addr.sin\_port = htons(rt); //端口要采用网络字节序

// 创建socket

IF\_CHECK(sd = socket(PF\_INET, SOCK\_DGRAM, 0));

// 这里bind绑定设置的地址

IF\_CHECK(bind(sd, (struct sockaddr\*)&addr, alen));

//开始监听了

head = \_new\_ucnode(&addr);

for(;;){

bzero(&msg, sizeof msg);

IF\_CHECK(recvfrom(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, &alen));

msg.name[\_INT\_NAME-1] = msg.text[\_INT\_TEXT-1] = '\0';

fprintf(stdout, "msg is [%s:%d] => [%c:%s:%s]\n", inet\_ntoa(addr.sin\_addr),

ntohs(addr.sin\_port), msg.type, msg.name, msg.text);

// 开始判断处理

switch(msg.type) {

case '1':\_login\_ucnode(head, sd, &addr, &msg);break;

case '2':\_broadcast\_ucnode(head, sd, &addr, &msg);break;

case '3':\_quit\_ucnode(head, sd, &addr, &msg);break;

case '4':\_broadcast\_ucnode(head, sd, &addr, &msg);break;

default://未识别的异常报文,程序把其踢走

fprintf(stderr, "msg is error! [%s:%d] => [%c:%s:%s]\n", inet\_ntoa(addr.sin\_addr),

ntohs(addr.sin\_port), msg.type, msg.name, msg.text);

\_quit\_ucnode(head, sd, &addr, &msg);

break;

}

}

// 这段代码是不会执行到这的, 可以加一些控制让其走到这. 看人

close(sd);

\_destroy\_ucnode(&head);

return 0;

}

4.2 客户端代码

文件 udpclt.c

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/wait.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/socket.h>

// 名字长度包含'\0'

#define \_INT\_NAME (64)

// 报文最大长度,包含'\0'

#define \_INT\_TEXT (512)

//4.0 控制台打印错误信息, fmt必须是双引号括起来的宏

#define CERR(fmt, ...) \

fprintf(stderr,"[%s:%s:%d][error %d:%s]" fmt "\r\n",\

\_\_FILE\_\_, \_\_func\_\_, \_\_LINE\_\_, errno, strerror(errno),##\_\_VA\_ARGS\_\_)

//4.1 控制台打印错误信息并退出, t同样fmt必须是 ""括起来的字符串常量

#define CERR\_EXIT(fmt,...) \

CERR(fmt,##\_\_VA\_ARGS\_\_),exit(EXIT\_FAILURE)

/\*

\* 简单的Linux上API错误判断检测宏, 好用值得使用

\*/

#define IF\_CHECK(code) \

if((code) < 0) \

CERR\_EXIT(#code)

// 发送和接收的信息体

struct umsg{

char type; //协议 '1' => 向服务器发送名字, '2' => 向服务器发送信息, '3' => 向服务器发送退出信息

char name[\_INT\_NAME]; //保存用户名字

char text[\_INT\_TEXT]; //得到文本信息,空间换时间

char siliao[\_INT\_NAME];

};

/\*

\* udp聊天室的客户端, 子进程发送信息,父进程接受信息

\*/

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

int sd, rt;

struct sockaddr\_in addr = { AF\_INET };

socklen\_t alen = sizeof addr;

pid\_t pid;

struct umsg msg = { '1' };

// 这里简单检测

if(argc != 4) {

fprintf(stderr, "uage : %s [ip] [port] [name]\n", argv[0]);

exit(-1);

}

// 下面对接数据

if((rt = atoi(argv[2]))<1024 || rt > 65535)

CERR("atoi port = %s is error!", argv[2]);

// 接着判断ip数据

IF\_CHECK(inet\_aton(argv[1], &addr.sin\_addr));

addr.sin\_port = htons(rt);

// 这里拼接用户名字

strncpy(msg.name, argv[3], \_INT\_NAME - 1);

char benji[\_INT\_NAME];

fgets(benji,\_INE\_NAME,stdin);

//创建socket 连接

IF\_CHECK(sd = socket(PF\_INET, SOCK\_DGRAM, 0));

// 这里就是发送登录信息给udp聊天服务器了

IF\_CHECK(sendto(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, alen));

//开启一个进程, 子进程处理发送信息, 父进程接收信息

IF\_CHECK(pid = fork());

if(pid == 0) { //子进程,先忽略退出处理防止成为僵尸进程

signal(SIGCHLD, SIG\_IGN);

while(fgets(msg.text, \_INT\_TEXT, stdin)){

if(strcasecmp(msg.text, "quit\n") == 0){ //表示退出

msg.type = '3';

// 发送数据并检测

IF\_CHECK(sendto(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, alen));

break;

}

else if(strcasecmp(msg.text,"siliao\n") == 0)

{

msg.type= '4';

printf("please input name\n");

fgets(msg.siliao,\_INT\_NAME,stdin);

printf("please input message\n");

fgets(msg.text,\_INT\_TEXT,stdin);

IF\_CHECK(sendto(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, alen));

}

else

// 洗唛按发送普通信息

{msg.type = '2';

IF\_CHECK(sendto(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, alen));

}

}

// 处理结算操作,并杀死父进程

close(sd);

kill(getppid(), SIGKILL);

exit(0);

}

// 这里是父进程处理数据的读取

for(;;){

bzero(&msg, sizeof msg);

IF\_CHECK(recvfrom(sd, &msg, sizeof msg, 0, (struct sockaddr\*)&addr, &alen));

msg.name[\_INT\_NAME-1] = msg.text[\_INT\_TEXT-1] = '\0';

switch(msg.type){

case '1':printf("%s 登录了聊天室!\n", msg.name);break;

case '2':printf("%s 说了: %s\n", msg.name, msg.text);break;

case '3':printf("%s 退出了聊天室!\n", msg.name);break;

case '4':{

if(strcmp(benji,msg.siliao)==0)

printf("%s dui ni shuo : %s\n",msg.name,msg.text);

}break;

default://未识别的异常报文,程序直接退出

fprintf(stderr, "msg is error! [%s:%d] => [%c:%s:%s]\n", inet\_ntoa(addr.sin\_addr),

ntohs(addr.sin\_port), msg.type, msg.name, msg.text);

goto \_\_exit;

}

}

\_\_exit:

// 杀死并等待子进程退出

close(sd);

kill(pid, SIGKILL);

waitpid(pid, NULL, -1);

return 0;

}

五、遇到的问题：

5.1 ID不匹配

需要重新输入本机的名字，才能使用函数进行判断