## 《Linux 操作系统设计实践》实验三：网络编程

实验环境：Ubuntu 14.04.5 LTS

实验内容：

运行代码：

E3GOBANG.c 主程序，用以连接和收发数据，并处理下棋事件

#include <stdio.h>

#include <sys/socket.h>

#include <unistd.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "GOBANG.h"

#define PORT 16923

int main()

{

char inputbuf[100];

char recbuf[100];

int final = 0;

int over = 0;

int player = 1;

int x, y;

int role = 0;

int server\_sockfd, client\_sockfd;

struct sockaddr\_in server\_addr = {0};

struct sockaddr\_in client\_addr = {0};

int client\_sockaddr\_len;

int optval = 1;

do {

printf("(L)isten , (C)onnect or (E)xit:");

scanf("%s", inputbuf);

switch(inputbuf[0]) {

case 'L':case 'l':

server\_sockfd = socket(AF\_INET, SOCK\_STREAM, 0); //服务器建立套接字

if(server\_sockfd == -1) {

perror("");

return 1;

}

if((setsockopt(server\_sockfd, SOL\_SOCKET, SO\_REUSEADDR, &optval, sizeof(optval))) == -1) { //设置套接字选项

perror("Setsockopt Error");

return 1;

}

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

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

server\_addr.sin\_port = htons(PORT);

if(bind(server\_sockfd, (struct sockaddr \*) &server\_addr, sizeof(server\_addr)) == -1) { //套接字绑定

perror("Bind Error");

return 1;

}

if(listen(server\_sockfd, 5) == -1) { //进行客户端监听

perror("Listen Error");

return 1;

}

printf("Waiting client...\n");

client\_sockaddr\_len = sizeof(client\_addr);

client\_sockfd = accept(server\_sockfd, (struct sockaddr \*) &client\_addr, (socklen\_t \*\_\_restrict) &client\_sockaddr\_len); //等待客户端连接

if(client\_sockfd == -1) {

perror("");

return 1;

}

role = 1; //设定服务器对应玩家角色

break;

case 'C':case 'c':

printf("Server address:");

scanf("%s", inputbuf);

client\_sockfd = socket(AF\_INET, SOCK\_STREAM, 0); //客户端建立套接字

if(client\_sockfd == -1) {

perror("");

return 1;

}

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

client\_addr.sin\_port = htons(PORT);

client\_addr.sin\_addr.s\_addr = inet\_addr(inputbuf); //根据手动输入地址进行连接

if(client\_addr.sin\_addr.s\_addr == -1) {

printf("Address is invalid");

return 1;

}

server\_sockfd = connect(client\_sockfd, (struct sockaddr \*)(&client\_addr), sizeof(client\_addr)); //连接服务器

if(server\_sockfd == -1) {

perror("Connect failed");

return 1;

}

role = 2; //设定服务器对应玩家角色

break;

case 'E':case 'e':

return 0;

default:

role = 0;

}

} while(!role);

do {

printtable(); //打印棋盘

if(player == role) { //当前与对应端一致

do {

printf("Player %s (INPUT:Ln or (Q)uit):", tablechar[player]);

scanf("%s", inputbuf);

ipttoop(inputbuf, &x, &y, &over);

} while((!over) && (x < 0 || y < 0));

send(client\_sockfd, inputbuf, sizeof(inputbuf), 0); //发送己方指令

} else { //

printf("Waiting Player %s ...\n", tablechar[player]);

recv(client\_sockfd, recbuf, sizeof(recbuf), 0); //接收对方指令

ipttoop(recbuf, &x, &y, &over);

}

if(over)break;

if(set(x, y, player)) { //放置棋子

printf("Set Error!\n");//放置失败

} else {

player = 3 - player; //放置成功，切换玩家

}

final = check(); //检测游戏是否结束

} while(!final && !over);

if(!over) {

printtable();

switch(final) { //游戏结果

case 1:

printf("Player () WIN!\n");

break;

case 2:

printf("Player >< WIN!\n");

break;

case 3:

printf("DRAW!\n");

break;

default:

printf("ERROR");

}

}

close(server\_sockfd);

close(client\_sockfd);

return 0;

}

GOBANG..h 五子棋头文件，内含棋盘、落子、判断胜负等函数。

#ifndef \_GOBANG\_H\_

#define \_GOBANG\_H\_

#include <stdlib.h>

int table[15][15] = {0}; //棋盘

const char tablechar[][3] = {" ", "()", "><"}; //棋盘图案

int lastx, lasty, count = 0;

void printtable() //打印棋盘

{

int i, j;

printf(" A B C D E F G H I J K L M N O\n");

for(i = 0; i < 15; ++i) {

if(i)printf(" --+--+--+--+--+--+--+--+--+--+--+--+--+--+--\n");

printf("%2d ", i + 1);

for(j = 0; j < 15; ++j) {

if(j)printf("|");

printf("%s", tablechar[table[i][j]]);

}

printf("\n");

}

}

int set(int x, int y, int c) //放置棋子

{

if(x >= 15)return 1;

if(y >= 15)return 1;

if(table[y][x])return 1;

else table[y][x] = c;

lastx = x;

lasty = y;

++count;

return 0;

}

const int wayx[8] = { -1, +1, -1, +1, -1, +1, +0, +0};

const int wayy[8] = { -1, +1, +1, -1, +0, +0, -1, +1};

int check() //检测胜利

{

if(count == 15 \* 15)return 3;

int i, j, x, y, d;

int c = table[lasty][lastx];

for(i = 0; i < 8 ; ++i) {

x = lastx;

y = lasty;

for(j = 0; j < 4; ++j) {

x += wayx[i];

y += wayy[i];

if(x < 0 || x >= 15)break;

if(y < 0 || y >= 15)break;

if(table[y][x] != c)break;

}

if((i&1)==0)d=j;

else {

d+=j+1;

if(d >= 5)return c;

}

}

return 0;

}

void ipttoop(char \*ipt, int \*x, int \*y, int \*over) //将字符指令转为操作

{

char buf[3] = {0};

int tx, ty;

tx = ipt[0] - 'A';

if(tx > ('Z' - 'A'))tx = ipt[0] - 'a';

if(tx == ('Q' - 'A'))\*over = 1;

if(tx > ('O' - 'A'))tx = -1;

if(tx < 0)tx = -1;

buf[0] = ipt[1];

buf[1] = ipt[2];

buf[3] = 0;

ty = atoi(buf) - 1;

if(ty >= 15)ty = -1;

\*x = tx;

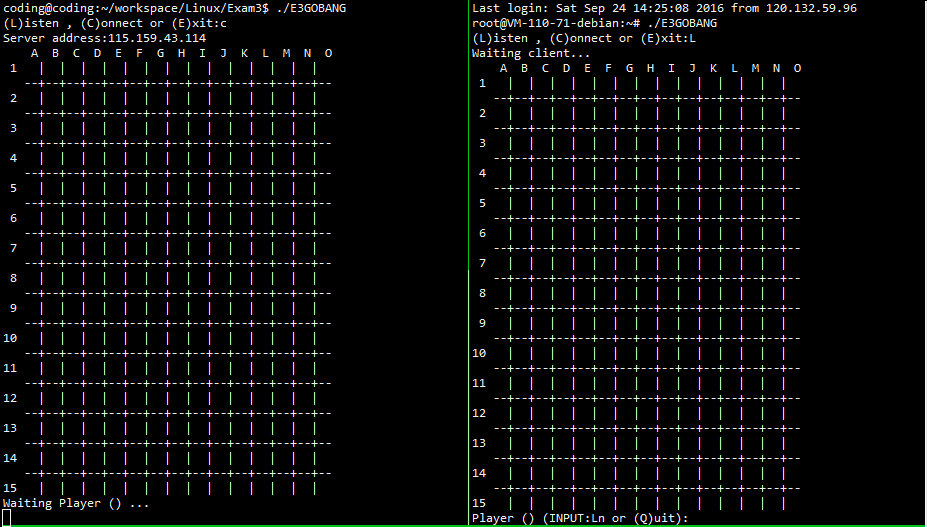
\*y = ty;

}

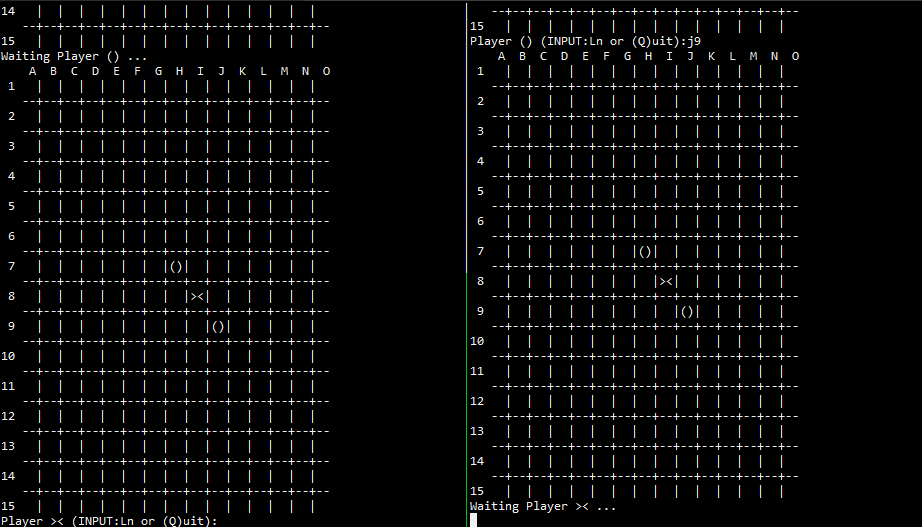
#endif

运行结果：

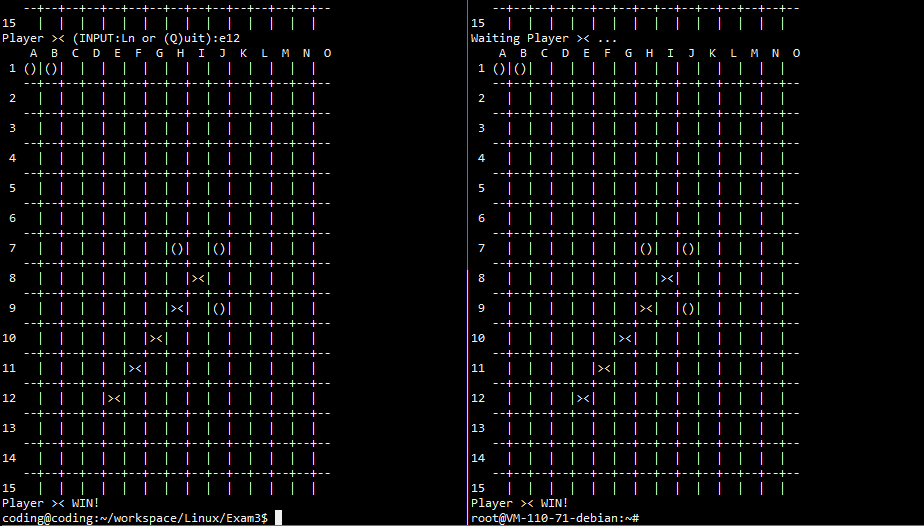
与远程服务器建立的五子棋游戏。



相互落子，并让对方收到结果。



游戏结束，某方胜利，同时连接中断。



实验总结：

之前编写的代码会导致一次游戏后连接不上，经检查是服务器端bind进行绑定失败，之后利用setsockopt函数设定套接字类型，使得连接中断后下次连接能够正常。

游戏虽用类似S/C机制进行连接，但对局的相关计算仍由服务器与客户机自行处理，网络仅发送对方所下位置。