## 五子棋

#### Fivechess.c

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include"Message.h"

void show\_chess\_board(int (\*a)[20][20]);//显示棋盘 函数参数为全局的数组指针(传递指针，函数运行得更快)

void get\_inputchess();//获取用户的下棋的位置，并更新本机全局数组的内容

int win(int a);//判断胜负，根据a的值来判断是黑方还是白方胜利，a=1判断黑方，a=2判断白方

//上面函数在这个文件下方，下面三个函数在socket.c

int bind\_server\_socket(char \*s1,char \*s2);//主机用自己的ip地址建立sock，并bind（参数传递IP地址与端口号）

int client\_socket\_connect(char\* s1,char\* s2);//客户端连接

int decompre(MSGSTRUCT msg);//解析获取的消息

//服务器或者客户端解析收到的消息，并更新board数组（数组更新在里面实现）

void race\_servercircle();//循环执行，反复接受消息，判断胜负，接受用户下子，并发送数据，主机使用这个函数

void race\_clientcircle();//功能同上，但是这个函数是客户端使用

int player\_pick=5;//0 代表黑方主机 , 1 代表白方 客户端，初始化为5是为了避免出现因为没有初始化而出现0值

int server\_sock;//主机的listen sock

int server\_connect\_sock;//主机accept后获得的sock，用此sock与客户端通信

int client\_sock;//客户端使用的sock

MSGSTRUCT rece\_msg;//接收信息用的缓冲区

MSGSTRUCT send\_msg;//发送信息用的缓冲区

struct sockaddr\_in saddr;//主机使用

struct sockaddr\_in client\_saddr;//客户端使用

int x;//用来存储用户下棋的行数

int y;//用来存储用户下棋的列数

int board[20][20]= {0}; //全局的数组 存放棋子用的

//server and client 都使用如下方式启动，注意ip地址是主机的ip地址

//server ./a.out 192.168.9.179 80

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

{

if(argc!=3||atoi(argv[2])<=0)//判断命令行的参数是否足够和端口值是否非法并显示如何启动的提示信息，没有help请自行完整程序

{

printf("use ./program hostname port\n");

printf("for example a.out 192.168.9.179 80 \n");

printf("please note the ip address means your vitual machine 's ip in the vmware software. you can use ifconfig to check it\n");

exit(1);

}

//告知用户如何选择主机和客户端

printf("please choose the white or black party\n");

printf("the black party is the server\n");

printf("the white party is the client\n");

printf("input 0 means black and sever\n");

printf("input 1 means white and client\n");

//获取用户选择，并保存在全局变量play-pick内

while(1)

{

scanf("%d",&player\_pick);

getchar();//这个getchar是把用户敲击的回车从stdin文件缓冲里读取掉，避免后面使用scanf时候读到/n，程序无法正确工作的。

if(0==player\_pick||1==player\_pick)

break;//用户输入1或者0后，才退出循环，否则一直循环让用户选择

else

printf("wrong input,try again,only input 0 or 1\n");

}

//显示提示信息

printf("please input the position x/y then press enter\n");

printf("you can use the number to judge which row or line the position to put\n");

show\_chess\_board(&board);//显示棋盘

//主机（黑色）先行，客户端（白色）等待主机

if(player\_pick==0) //server创建通信

{

if((server\_sock=bind\_server\_socket(argv[1],argv[2]))==-1) //绑定socket

oops("socket");

if(listen(server\_sock,10)==-1) //10是设定监听队列为10，主机依次处理

oops("listen error");

server\_connect\_sock=accept(server\_sock,0,0);//后两个参数可以用来获取客户端的地址，这里用不到，设置为0

//sever\_sock建立通信使得许多客户端可以连接到主机（等待的队列长度由listen决定），而/>accept得到的sever\_connect\_sock是主机与每个客户端相连的描述符

if(server\_connect\_sock==-1) //获取server\_connect\_sock出粗accept出错返回-1,此时退出

oops("accept error");

race\_servercircle();//进入接收发送消息循环，开始游戏

}

else //客户端

{

if((client\_sock= client\_socket\_connect(argv[1],argv[2]))==-1) oops("socket client");

race\_clientcircle(); //进入接收发送消息循环，开始游戏

}

} //主函数到这里结束

void race\_clientcircle() //客户端接受与发送消息循环，因为后落字所以先接受消息

{

while(1)

{

do

{

if(recv(client\_sock,&rece\_msg,sizeof(rece\_msg),0)==-1) //接收消息，第四个参数一般设置为0此时相当于read

oops("recvfrom"); //如果接受消息错误，退出

}

while(decompre(rece\_msg)==-1); //解析数据结构体(读取得到信息直到得到正确的信息，否则一直循环)

//接收黑方消息后，判断黑方落子后是否分出胜负

if(win(1)) //判断server方是否已经取得胜利

{

printf("black win! pause!");

//pause();让程序暂停，以后补充用，这里不考虑

}

show\_chess\_board(&board);

printf("receive sever's info succeed!\n");

printf("Are you sure to contine?"); //提示用户选择消息类型，是继续落子还是退出

printf("Or you can choose give in!\n");

printf("type 1 contine,type 2 give in\n");

scanf("%u",&send\_msg.msgType); //输入消息类型

getchar();

if(1==send\_msg.msgType) //1表示继续下棋，其余类型不需要额外的信息

{

get\_inputchess(); //如果用户继续落子，封装信息结构体

send\_msg.x=x;

send\_msg.y=y;

send\_msg.msgType=1; //设置落子信息以及消息类型

}

if(win(2)) //判断落子后client方是否已经取得胜利

{

printf("white win! pause!");

//pause();同上

}

if(send(client\_sock,&send\_msg,sizeof(send\_msg),0)==-1) oops("send"); //如果发送消息失败提示错误信息并退出，第四个参数一般设置为0,，此时相当于write

show\_chess\_board(&board); //刷新棋盘输出，让用户看到自己的落子

printf("client send message success! waiting for receive message from sever\n");//发送成功后让用户知道

}

}

void race\_servercircle() //主机端接受消息与发出消息循环，因为先落子所以先发送消息

{

while(1)

{

printf("Are you sure to contine?"); //提示信息，让用户选择继续下棋还是退出

printf("Or you can choose give in!\n");

printf("type 1 contine,type 2 give in\n");

scanf("%u",&send\_msg.msgType); //执行循环，选1继续下棋，

getchar(); //getchar是把用户敲击的回车从stdin文件缓冲里读取掉，避免后面使用scanf时候读到/n，程序无法正确工作的

if(1==send\_msg.msgType) //判断用户是否决定继续落子

{

get\_inputchess();

//封装信息结构体

send\_msg.x=x;

send\_msg.y=y; //设置落子位置以及消息类型

send\_msg.msgType=1;

}

if(win(1)) //判断主机落子后是否产生胜负

{

printf("black win! pause!");

//pause();暂停程序，为了以后的补充

}

if(send(server\_connect\_sock,&send\_msg,sizeof(send\_msg),0)==-1) //服务器端先下棋，所以先send发送消息，客户端先recv接收消息

oops("oop sendto"); //如果消息发送失败，退出

show\_chess\_board(&board); //刷新棋盘输出，让用户看到自己的落子

printf("sever send message success,please waiting for client message\n"); //让用户知道发送消息已经成功，等待对方落子

//接受数据

do

{

if(recv(server\_connect\_sock,&rece\_msg,sizeof(rece\_msg),0)==-1)

oops("recvfrom"); //如果接受消息错误，退出

}

while( decompre(rece\_msg)==-1); //读取信息直到用户得到正确信息否则一直循环

if(win(2))

{

printf("white win! pause!"); //对方下子后是否已经分出胜负

//pause();同上

}

show\_chess\_board(&board); //刷新棋盘

printf("receive client's message succeed!\n");//让用户知道自己接受消息成功

}

}

void get\_inputchess()

{

int circle\_x=0;//用于判断是否进行循环，0代表输入正确不用循环，1代表输入错误继续循环

int circle\_y=0;

char local\_x,local\_y; //存放要落棋的坐标

do

{

circle\_x=0;//每一次循环开始前将其设置为0，如果没有进行到deafult代表输入正确跳出循环，否则circle\_x被设置为1执行循环

printf("please input the chess of line:");

scanf("%c",&local\_x); //获取行坐标

getchar(); //getchar是把用户敲击的回车从stdin文件缓冲里读取掉，避免后面使用scanf时候读到/n，程序无法正确工作的

switch(local\_x)

{

case '1':

x=0;

break;

case '2':

x=1;

break;

case '3':

x=2;

break;

case '4':

x=3;

break;

case '5':

x=4;

break;

case '6':

x=5;

break;

case '7':

x=6;

break;

case '8':

x=7;

break;

case '9':

x=8;

break;

case 'a':

x=9;

break;

case 'b':

x=10;

break;

case 'c':

x=11;

break;

case 'd':

x=12;

break;

case 'e':

x=13;

break;

case 'f':

x=14;

break;

case 'g':

x=15;

break;

case 'h':

x=16;

break;

case 'i':

x=17;

break;

case 'j':

x=18;

break;

case 'k':

x=19;

break;

default:

printf("wrong input\n"); //错误输入就一直执行循环

circle\_x=1;

break;

}

}

while(circle\_x==1); //正确输入跳出，否则循环

//x输入结束

printf("please input the chess of row:"); //y的处理同上

do

{

circle\_y=0;

scanf("%c",&local\_y); //获取列坐标

getchar();

switch(local\_y)

{

case '1':

y=0;

break;

case '2':

y=1;

break;

case '3':

y=2;

break;

case '4':

y=3;

break;

case '5':

y=4;

break;

case '6':

y=5;

break;

case '7':

y=6;

break;

case '8':

y=7;

break;

case '9':

y=8;

break;

case 'a':

y=9;

break;

case 'b':

y=10;

break;

case 'c':

y=11;

break;

case 'd':

y=12;

break;

case 'e':

y=13;

break;

case 'f':

y=14;

break;

case 'g':

y=15;

break;

case 'h':

y=16;

break;

case 'i':

y=17;

break;

case 'j':

y=18;

break;

case 'k':

y=19;

break;

default:

printf("wrong input\n");

circle\_y=1;

break;

}

}

while(circle\_y==1);

//y输入结束

if(player\_pick==0) //board赋值了两次，这里是给自己的棋子赋值，decompre函数里面是对面的棋子

board[x][y]=1; //根据主机，客户端不同确定下的棋子的样式

else

board[x][y]=2;

}

void show\_chess\_board(int (\*a)[20][20]) //这个函数可以把可执行文件打开对照显示效果看就很容易懂了

{ //a为三级指针

//显示最开头的环行

int i,j,k=0;//every call this fuction clear k to 0 //k用来控制显示空格，换行

printf("\n\n");

printf(" 12345 6789a bcdef ghijk\n"); //打印出列坐标

for(i=0; i<20; i++) //i为行

{

for(j=0; j<20; j++) //j为列

{

//实现每五个棋子就出现一个空格，方便查找位置去下子

if(k>0&&(k%5==0))

printf(" ");

//每20个棋子输出，就换行显示

if(k>0&&(k%20==0))

printf("\n");

//显示每行的开头的数字

if(j==0)

{

if (i<9) printf("%d",i+1); //开始9行用数字显示

else

{

switch(i)

{

case 9:

printf("%c",'a');

break;

case 10:

printf("%c",'b'); //1-9不够显示行坐标，9后用a-k表示

break;

case 11:

printf("%c",'c');

break;

case 12:

printf("%c",'d');

break;

case 13:

printf("%c",'e');

break;

case 14:

printf("%c",'f');

break;

case 15:

printf("%c",'g');

break;

case 16:

printf("%c",'h');

break;

case 17:

printf("%c",'i');

break;

case 18:

printf("%c",'j');

break;

case 19:

printf("%c",'k');

break;

}

}

}

/\* 0 print -, 1 black print X ,2 white print O \*/

if((\*a)[i][j]==0)

printf("-"); //此时代表未落子

else if((\*a)[i][j]==1)

printf("X"); //1代表主机落的子，2代表客户端落的子

else if((\*a)[i][j]==2)

printf("O");

k++; //每落一个子后k++来控制空格与换行

}

}

printf("\n"); //打印棋盘完后再输出一个换行，让用户看得更清楚

}

//this win function can improve in many aspects.

//we just need to draw a picture, then it is easy to feel that

//we must have five line or row, then can be lined in five.

//so in following lines we can jump several lines or row according to

//different direction in chess board. so the y<17! so the x<17!

//so the

int win(int a) //a=1 means black, a=2 means white //判断胜负

{

int x,y;//x means line,y means column

//judge line

for(y=0; y<17; y++) //last four column no need to judge

{

for(x=0; x<20; x++)

{

if(a==board[x][y]&&a==board[x+1][y]&&a==board[x+2][y]&&a==board[x+3][y]&&a==board[x+4][y])

return a;// win

}

}

//judge vitical

for(y=0; y<20; y++)

{

for(x=0; x<17; x++) //last four line no need to judge

{

if(a==board[x][y]&&a==board[x][y+1]&&a==board[x][y+2]&&a==board[x][y+3]&&a==board[x][y+4])

return a;// win

}

}

// judge "\"

for(y=0; y<20; y++)

{

for(x=0; x<17; x++) //last four line no need to judge

{

if(a==board[x][y]&&a==board[x+1][y+1]&&a==board[x+2][y+2]&&a==board[x+3][y+3]&&a==board[x+4][y+4])

return a;// win

}

}

// judge "/"

for(y=19; y>3; y--)

{

for(x=0; x<17; x++) //last four line no need to judge

{

if(a==board[x][y]&&a==board[x+1][y-1]&&a==board[x+2][y-2]&&a==board[x+3][y-3]&&a==board[x+4][y+4])

return a;// win

}

}

return 0;

}

#### Socket.c

#include"Message.h"

extern int board[20][20];

extern int player\_pick;

/\*

\*return -1 means error

\* This is server address

\*/

int bind\_server\_socket(char \*s1,char \*s2) //创建主机socket, s1表示IP地址，s2表示端口

{

unsigned long inaddr; //用来储存ip地址

struct sockaddr\_in saddr; //地址结构

memset(&saddr,0,sizeof(struct sockaddr\_in)); //开始将这一块内存清空

int sock\_id\_s;

sock\_id\_s=socket(AF\_INET,SOCK\_STREAM,0);

if(sock\_id\_s==-1)

return -1;

saddr.sin\_family=AF\_INET;

inaddr=inet\_addr(s1); //将点分地址转化为无符号长整型

memcpy(&saddr.sin\_addr,&inaddr,sizeof(inaddr));

saddr.sin\_port=htons(atoi(s2));//将端口号转化为网络字节序

if(bind(sock\_id\_s,(struct sockaddr \*)&saddr,sizeof(saddr))!=0)

return -1;

return sock\_id\_s;

}

int client\_socket\_connect(char \*s1,char \*s2) //创建客户端socket，s1表示IP地址，s2表示端口

{

unsigned long inaddr; //用来储存ip地址

struct sockaddr\_in saddr; //地址结构

memset(&saddr,0,sizeof(struct sockaddr\_in));//开始将这一块内存清空

int sock\_id\_c;

sock\_id\_c=socket(AF\_INET,SOCK\_STREAM,0);

if(sock\_id\_c==-1)

return -1;

saddr.sin\_family=AF\_INET;

inaddr=inet\_addr(s1); //将点分地址转化为无符号长整型

memcpy(&saddr.sin\_addr,&inaddr,sizeof(inaddr));

saddr.sin\_port=htons(atoi(s2)); //将端口号转化为网络字节序

if(connect(sock\_id\_c,(struct sockaddr\*)&saddr,sizeof(saddr)) != 0)

return -1;

return sock\_id\_c;

}

int decompre(MSGSTRUCT msg)

{

int res = 0;

switch(msg.msgType)

{

case 1: //更新对方刚下的棋

printf("get chess position\n");

if (player\_pick==0)

//player\_pick==0 means that the machine is server

//so the sever machine only recive the message from client.

//the value of client for board array is 2

board[msg.x][msg.y]=2;

else

//player\_pick!=0 means that the machine is client

//so the client machine only recive the message from sever.

//the value of sever for board array is 1

board[msg.x][msg.y]=1;

break;

case 2:

printf("对方向您提出和棋请求,您是否同意？\n");

break;

case 3:

printf("对方同意了你的的和棋请求\n");

break;

case 4:

printf("对方拒绝了您的和棋请求，所以您得继续完成本次博弈\n");

break;

case 5:

printf("%s",msg.byMsg);

break;

default:

res = -1; //如果获取的消息不正确返回-1

break;

}

return res;

}

#### Message.h

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<netdb.h>

#include<string.h>

#define oops(x) {perror(x);exit(1);}//程序出错时候使用

#ifndef MESSAGE\_H\_

#define MESSAGE\_H\_

#define HOSTLEN 256

#define MSG\_PUTSTEP 0x01

#define MSG\_DRAW 0x02

#define MSG\_AGREE\_DRAW 0x03

#define MSG\_DENY\_DRAW 0x04

#define MSG\_EXTERN 0x05

typedef struct Message

{

int len;

// 消息ID

unsigned int msgType;

// 落子信息

int x;

int y;

//书上的消息体结构，因为只是演示棋子落子，所以只用了msgType,x,y

//有些功能没有实现你们可以先看他的demo视频内容

int color;

// 其他消息内容

char byMsg[128];

}MSGSTRUCT;

#endif