**实验12 虚拟字符驱动程序设计**

**一、实验目的**

1.掌握简单游戏程序的编写

2.掌握驱动程序的编译方法；

**二、**实验内容

任务一：编译运行书本P：442扩展程序（相关资料见钉钉目录）。

#include <SDL.h>

#include <stdio.h>

#include <stdlib.h>

#include <SDL\_ttf.h>

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

{

    SDL\_Surface \*screen, \*text; /\*指针text指向文字屏幕\*/ /\*图像指针, image2代表人物，image代表怪物\*/

    SDL\_Surface \*image2, \*image;

    SDL\_Event event;                        /\*事件对象\*/

    TTE\_Font \*Nfont;                        /\*文字样式对象\*/

    Uint32 color;                           /\*定义一个颜色值\*/

    int bAppRun = l, right, left, up, down; /\*程序运行和人物运动方向标志\*/

    int bTopBottom = l;                     /\*屏幕上下\*/

    int bLeftRight = l;                     /\*屏幕左右 \*/

    Uint32Tstart, Tstop;                    /\*时间开始和结束\*/

    SDL\_Rect dRect, dRect2, dRect3;         /\*目标矩形，分别代表怪物、人物、文字\*/

    if(SDL\_Init (SDL\_INIT\_VIDEO ( SDL\_INIT\_TIMER) ==-l)

    {

        /\*初始化视频子系统和计时器子系统\*/

        fprintf(stderr, "不能初始化%s\n", SDL\_GetError());

        exit(1);

    }

    atexit(SDL\_Quit);                                       /\*退出\*/

    screen = SDL\_SetVideoMode(960, 720, 16, SDL\_SWSURFACE); /\*设置视频模式\*/

    if (screen == NULL)

    {

        fprintf(stderr, "不能初始化960\*720，16位色的视频模式: %s \n", SDLGetError());

        exit(1);

    }

    image2=SDL\_LoadBMP (" . / 1 . bmp" ) ; /\*加载人物图片\*/

    image = SDL\_LoadBMP("./ 2 .bmp"); /\*加载怪物图片\*/

    if (image == NULL|| image2 ==NULL)

    {

        fprintf(stderr, "Couldn 't load BMP,%s\n", SDL\_GetError());

        exit(1);

    }

    dRect.x=0;/\*怪物目标矩形的左上角坐标\*/

    dRect.y=0;

    dRect.w=image->w ;/\*目标矩形的宽和高是位图的宽和高\*/

    dRect.h=image->h;

    if(SDL\_BlitSurface (image ,NULL,screen , &dRect)<0)

    {

        fprintf(stderr, "BlitSurface error: %s \n", SDL\_GetError());

        SDL\_Freesurface(image);

        exit(l);

    }

    dRect2.x = 0; /\*人物目标矩形的左上角坐标\*/

    dRect2.y = 480;

    dRect2.w = image2->w;

    dRect2.h = image2->h;

    if (SDL\_BlitSurface(image2, NULL, screen, &dRect2) < 0)

    {

        fprintf(stderr, "BlitSurface error: 品s \n ", SDL\_GetError());

        SDL\_Freesurface(image2);

        exit(l);

    }

    SDL\_UpdateRect(screen, 0, 0, 0, 0); /\*更新屏幕\*/

    Tstart = SDLGetTicks();             /\*计时开始\*/

    while (bAppRun == 1)

    {

        while (SDL\_PollEvent(&event))

        {

            if (event.type == SDL\_KEYDOWN) /\*判断是否有按键\*/

                switch (event.key.keysym.sym)

                {                 /\*判断按键类型\*/

                case SDLK\_ESCAPE: /\*按下ESC键直接退出\*/

                    bAppRun = 0;

                    break;

                case SDLK\_RIGHT: /\*按下右键，右运动标志参数为真，其余为假\*/

                    right = l;

                    left = 0;

                    up = 0;

                    down = 0;

                    break;

                case SDLK\_LEFT: /\*按下左键，左运动标志参数为真，其余为假\*/

                    left = 1;

                    right = 0;

                    up = 0;

                    down = 0;

                    break;

                case SDLK\_UP: /\*按下上键，上运动标志参数为真，其余为假\*/

                    up = l;

                    right = 0;

                    left = 0;

                    down = 0;

                    break;

                case SDLK\_DOWN: /\*按下下键，下运动标志参数为真，其余为假\*/

                    down = l;

                    right = 0;

                    left = 0;

                    up = 0;

                    break;

                }

        }

        Tstop = SDL\_GetTicks();

        if ((Tstop - Tstart) > 5)

        {

            SDL\_FillRect(screen, &dRect, 0);

            if (bTopBottom == 1)

            { /\*如果碰到屏幕上部或下部\*/

                if ((dRect.y + dRect.h + 3) < screen->h)

                {

                    dRect.y += 3;

                }

                else

                {

                    bTopBottom = 0;

                    dRect.y = screen->h - dRect.h;

                }

            }

            else

            {

                if ((dRect.y - 3) > 0)

                {

                    dRect.y -= 3;

                }

                else

                {

                    bTopBottom = 1;

                    dRect.y = 0;

                }

            }

            if (bLeftRight == 1)

            { /\*如果碰到右边或是左边\*/

                if ((dRect.x + dRect.w + 3) < screen->w)

                {

                    .dRect.x += 3;

                }

                else

                {

                    bLeftRight = 0;

                    dRect.x = screen->w - dRect.w;

                }

            }

            else

            {

                if ((dRect.x - 3) > 0)

                    dRect.x -= 3;

                else

                {

                    bLeftRight = l;

                    dRect.x = 0;

                }

            }

            if (SDL\_BlitSurface(image, NULL, screen, &dRect) < 0)

            {

                /\*把目标快速转化\*/

                fprintf(stderr, "BlitSurface error: %s\n", SDL\_GetError());

                SDL\_FreeSurface(image);

            }

            SDL\_Flip(screen); /\*屏幕缓冲\*/

        }

        if ((Tstop - Tstart) > 15)

        {

            Tstart = Tstop;

            SDL\_FillRect(screen, &dRect2, 0);

            if (right == 1)

            {

                /\*判断是否已经碰到屏幕右边沿 \*/

                if ((dRect2.x + dReet2.w + 3) < screen->w)

                    dRect2.x += 3;

            }

            if (left == 1)

            {

                if ((dRect2.x - 3) > 0) /\*判断是否已经碰到屏幕左边沿 \*/

                    dRect2.x -= 3;

            }

            if (up == 1)

            {

                if ((dRect2.y - 3) > 0) /\*判断是否已经碰到屏幕上边沿\*/

                    dRect2.y -= 3;

            }

            if (down == 1)

            {

                /\*判断是否已经碰到屏幕下边沿\*/

                if ((idRect2.y + dRect2.h + 3) < screen->h)

                    dRect2 - y += 3;

            }

            if (SDL\_Blitsurface(image2, NULL, screen, &dRect2) < 0)

            {

                /\*把目标快速转化\*/

                fprintf(stderr, "BlitSurface error : ts}n", SDL\_GetError());

                SDL\_FreeSurface(image2);

                exit(l);

            }

            SDL\_Flip(screen);

        }

        if ((dRect2.x >= dPect.x - image2->w) & &(dRect2.x <= dRect.x + image->w))

        {

            /\*判断怪物是否和人物接触\*/

            if ((dRect2.y >= dRect.y - image2->h) && (dRect2.y <= dRect.y + image->h))

                break;

        }

        SDL\_FreeSurface(image);

        SDL\_FreeSurface(image2); /\*释放图像\*/

        int x;

        for (x = 0; x <= 255; x++) /\*用循环来实现背景色渐变\*/

        {

            color = SDL\_MapRGB(screen->format, 255, x, x); /\*设置颜色\*/

            SDL\_FillRect(screen, NULL, color);             /\*填充整个屏幕\*/

            SDL\_UpdateRect(screen, 0, 0, 0, 0);            /\*更新整个屏幕\*/

            SDL\_Delay(5);

        }

        SDL\_Csolor red = {255, 0, 0, 0}; /\*设置字体颜色\*/

        int fontsize = 50;               /\*设置字体大小\*/

        if (TTF\_Init() != 0)

        {

            fprintf(stderr, "can't init ttf font! n");

            exit(1);

        }

        /\*打开字体库\*/

        Nfont = TTF\_OpenFont("/usr/share/fonts/truetype/droid/Droidsans.ttf"，fontsize);

        TTF\_SetFontstyle(Nfont, TTF\_STYLE\_NORMAL); /\*设置字体样式\*/

        text = TTF\_RenderUTF8\_Blended(Nfont, "GAME OVER", red);

        TTF\_CloseFont(Nfont); /\*关闭字体库\*/

        TTF\_Quit();           /\*退出\*/

        dRect3.x = 200;       /\*在点(200,300)处开始写\*/

        dRect3.y = 300;

        dRect3.w = text->w; /\*目标矩形的宽和高分别的是所写字的宽和高\*/

        dRect3.h = text->h;

        SDL\_BlitSurface(text, NULL, screen, &dRect3); /\*把目标快速转化\*/

        SDL\_UpdateRect(screen, 0, 0, 0, o);

        sDL\_Freesurface(text);

        SDL\_Delay(3000); /\*停留3秒\*/

        return 0;

}

任务二：（1）编译运行书本P：454页程序。（相关图片资源见钉钉目录）。（2）在原有程序功能上，加上记分功能，每击中一架敌机，总分加100分，并在屏幕上显示。

图形用户界面, 文本, 应用程序

描述已自动生成

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <SDL.h>

#include <SDL\_ttf.h>

#ifdef macintosh

#define DIR\_SEP ":"

#define DIR\_CUR ":"

#else

#define DIR\_SEP "/"

#define DIR\_CUR ""

#endif

#define ROOT\_DIRECTORY "/home/ljr/Desktop/Exp12/data"

#define DATAFILE(X) ROOT\_DIRECTORY DIR\_SEP X

#define FRAME\_RATES 10                             //每秒钟的帧数

#define CANNON\_SPEED 5                             //大炮运动速度

#define MAX\_SHOTS 1                                //最多可以发射多少发炮弹

#define SHOT\_SPEED 10                              //炮弹的速度

#define MAX\_AIRS 2                                 //最多出现多少架飞机

#define AIR\_SPEED 5                                //飞机的速度

#define AIR\_ODDS (1 \* FRAME\_RATES)                 //剩下的飞机数

#define EXPLODE\_TIME 4                             //爆炸时间

#define MAX\_UPDATES 3 \* (1 + MAX\_SHOTS + MAX\_AIRS) //最大更新次数

#define SCORE\_PER\_AIR 5

int score = 0;

typedef struct object

{                       //对象结构体

    int alive;          //是否存活标志

    int facing;         //运动方向

    int x, y;           //坐标

    SDL\_Surface \*image; //图像指针

} object;

struct blit

{ //定义快速重绘对象结构体

    SDL\_Surface \*src;

    SDL\_Rect \*srcrect;

    SDL\_Rect \*dstrect;

} blits[MAX\_UPDATES];

SDL\_Surface \*screen, \*background; //背景指针

object cannon;                    //大炮对象

object shots[MAX\_SHOTS];          //炮弹对象

object airs[MAX\_AIRS];            //飞机对象

object explosions[MAX\_AIRS + 1];  //爆炸对象

SDL\_Rect srcUpdate[MAX\_UPDATES];  //源目标更新

SDL\_Rect dstUpdate[MAX\_UPDATES];  //目标更新

int reloading;                    //重新加载标志

int numUpdates;                   //更新次数的变量

SDL\_Surface \*loadImage(char \*datafile)

{

    SDL\_Surface \*image, \*surface;

    image = SDL\_LoadBMP(datafile);

    if (image == NULL)

    {

        fprintf(stderr, "Couldn't load image %s: %s\n", datafile, SDL\_GetError());

        return (NULL);

    }

    surface = SDL\_DisplayFormat(image);

    SDL\_FreeSurface(image);

    return (surface);

}

void printScore()

{

    SDL\_Color green = {0, 255, 0, 0};

    int fontSize = 30;

    TTF\_Font \*Nfont;

    SDL\_Surface \*text;

    SDL\_Rect fontRect;

    if (TTF\_Init() != 0)

    {

        fprintf(stderr, "Can not initialize TTF font.\n");

    }

    SDL\_FillRect(screen, NULL, SDL\_MapRGB(screen->format, 255, 255, 255));

    Nfont = TTF\_OpenFont("/usr/share/fonts/simsunb.ttf", fontSize);

    TTF\_SetFontStyle(Nfont, TTF\_STYLE\_NORMAL);

    char end[50];

    sprintf(end, "You got %d scores.", score);

    text = TTF\_RenderUTF8\_Blended(Nfont, end, green);

    TTF\_CloseFont(Nfont);

    TTF\_Quit();

    fontRect.x = 200;

    fontRect.y = 50;

    fontRect.w = text->w;

    fontRect.h = text->h;

    SDL\_BlitSurface(text, NULL, screen, &fontRect);

    SDL\_UpdateRect(screen, 0, 0, 0, 0);

    SDL\_FreeSurface(text);

    SDL\_Delay(5000);

}

int loadData(void)

{

    int i;

    //加载大炮图片

    cannon.image = loadImage(DATAFILE("cannon.bmp"));

    if (cannon.image == NULL)

        return 0;

    //加载炮弹图片

    shots[0].image = loadImage(DATAFILE("shot.bmp"));

    if (shots[0].image == NULL)

        return 0;

    for (i = 1; i < MAX\_SHOTS; ++i)

    {

        shots[i].image = shots[0].image;

    }

    //加载飞机图片

    airs[0].image = loadImage(DATAFILE("air.bmp"));

    if (airs[0].image == NULL)

        return 0;

    for (i = 1; i < MAX\_AIRS; ++i)

    {

        airs[i].image = airs[0].image;

    }

    //加载爆炸图片

    explosions[0].image = loadImage(DATAFILE("explosion.bmp"));

    if (explosions[0].image == NULL)

        return 0;

    for (i = 1; i < MAX\_AIRS + 1; ++i)

    {

        explosions[i].image = explosions[0].image;

    }

    //加载背景图片

    background = loadImage(DATAFILE("background.bmp"));

    if (background == NULL)

        return 0;

    //设置更新目标矩形的指针

    for (i = 0; i < MAX\_UPDATES; ++i)

    {

        blits[i].srcrect = &srcUpdate[i];

        blits[i].dstrect = &dstUpdate[i];

    }

    return 1;

}

void freeObject(void)

{

    int i;

    SDL\_FreeSurface(cannon.image);

    SDL\_FreeSurface(shots[0].image);

    SDL\_FreeSurface(airs[0].image);

    SDL\_FreeSurface(explosions[0].image);

}

void createAir(void)

{

    int i;

    for (i = 0; i < MAX\_AIRS; ++i)

    { //当飞机数没有达到最大值，并且存活值为1时，产生一架新的飞机

        if (!airs[i].alive)

            break;

    }

    if (i == MAX\_AIRS)

    {

        return;

    }

    //用随机函数产生三个数，分别用来标记运动方向

    do

    {

        airs[i].facing = (rand() % 3) - 1;

    } while (airs[i].facing == 0);

    //确定飞机初始时的y方向上的位置，0即表示屏幕的上部

    airs[i].y = 0;

    if (airs[i].facing < 0)

    { //如果飞机方向小于0，即反方向，飞机往左边移动

        airs[i].x = screen->w - airs[i].image->w - 1;

    }

    else

    {

        airs[i].x = 0;

    }

    airs[i].alive = 1;

}

void drawObject(object \*sprite) //画对象

{

    struct blit \*update;

    update = &blits[numUpdates++];

    update->src = sprite->image;

    update->srcrect->x = 0; //根据源目标的大小和位置坐标来画

    update->srcrect->y = 0;

    update->srcrect->w = sprite->image->w;

    update->srcrect->h = sprite->image->h;

    update->dstrect->x = sprite->x;

    update->dstrect->y = sprite->y;

    update->dstrect->w = sprite->image->w;

    update->dstrect->h = sprite->image->h;

}

void eraseObject(object \*sprite) //消除目标

{

    struct blit \*update;

    int wrap;

    //背景水平重叠达到清除画面效果

    update = &blits[numUpdates++];

    update->src = background;

    update->srcrect->x = sprite->x % background->w;

    update->srcrect->y = sprite->y;

    update->srcrect->w = sprite->image->w;

    update->srcrect->h = sprite->image->h;

    wrap = (update->srcrect->x + update->srcrect->w) - (background->w);

    if (wrap > 0)

    {

        update->srcrect->w -= wrap;

    }

    update->dstrect->x = sprite->x;

    update->dstrect->y = sprite->y;

    update->dstrect->w = update->srcrect->w;

    update->dstrect->h = update->srcrect->h;

    //一个背景一个背景的把屏幕重绘

    if (wrap > 0)

    {

        update = &blits[numUpdates++];

        update->src = background;

        update->srcrect->x = 0;

        update->srcrect->y = sprite->y;

        update->srcrect->w = wrap;

        update->srcrect->h = sprite->image->h;

        update->dstrect->x = ((sprite->x / background->w) + 1) \* background->w;

        update->dstrect->y = sprite->y;

        update->dstrect->w = update->srcrect->w;

        update->dstrect->h = update->srcrect->h;

    }

}

void UpdateScreen(void)

{

    int i;

    for (i = 0; i < numUpdates; ++i)

    {

        SDL\_LowerBlit(blits[i].src, blits[i].srcrect,

                      screen, blits[i].dstrect);

    }

    SDL\_UpdateRects(screen, numUpdates, dstUpdate);

    numUpdates = 0;

}

//碰撞检测

int collide(object \*sprite1, object \*sprite2)

{

    if ((sprite1->y >= (sprite2->y + sprite2->image->h)) ||

        (sprite1->x >= (sprite2->x + sprite2->image->w)) ||

        (sprite2->y >= (sprite1->y + sprite1->image->h)) ||

        (sprite2->x >= (sprite1->x + sprite1->image->w)))

    {

        return 0;

    }

    return 1;

}

void WaitFrame(void)

{

    static Uint32 next\_tick = 0;

    Uint32 this\_tick;

    this\_tick = SDL\_GetTicks();

    if (this\_tick < next\_tick)

    {

        SDL\_Delay(next\_tick - this\_tick); //延时

    }

    next\_tick = this\_tick + (1000.0 / FRAME\_RATES); //下一帧

}

void runGame(void)

{

    int i, j;

    SDL\_Event event;

    Uint8 \*keys;

    numUpdates = 0;

    SDL\_Rect dst; //开始把背景画上去

    dst.x = 0;

    dst.y = 0;

    dst.w = background->w;

    dst.h = background->h;

    SDL\_BlitSurface(background, NULL, screen, &dst);

    SDL\_UpdateRect(screen, 0, 0, 0, 0);

    //初始化大炮参数，存活，位置和运动方向

    cannon.alive = 1;

    cannon.x = (screen->w - cannon.image->w) / 2;

    cannon.y = (screen->h - cannon.image->h) - 1;

    cannon.facing = 0;

    drawObject(&cannon);

    //初始化炮弹的存活参数

    for (i = 0; i < MAX\_SHOTS; ++i)

    {

        shots[i].alive = 0;

    }

    //初始化飞机的存活参数

    for (i = 0; i < MAX\_AIRS; ++i)

    {

        airs[i].alive = 0;

    }

    createAir();          //产生飞机

    drawObject(&airs[0]); //画飞机

    UpdateScreen();

    while (cannon.alive == 1)

    {

        WaitFrame();

        while (SDL\_PollEvent(&event))

        {

            if (event.type == SDL\_QUIT)

                return;

        }

        keys = SDL\_GetKeyState(NULL);

        for (i = 0; i < MAX\_SHOTS; ++i)

        { //清除炮弹

            if (shots[i].alive)

            {

                eraseObject(&shots[i]);

            }

        }

        for (i = 0; i < MAX\_AIRS; ++i)

        { //清除飞机

            if (airs[i].alive)

            {

                eraseObject(&airs[i]);

            }

        }

        eraseObject(&cannon);

        for (i = 0; i < MAX\_AIRS + 1; ++i)

        { //清除爆炸

            if (explosions[i].alive)

            {

                eraseObject(&explosions[i]);

            }

        }

        for (i = 0; i < MAX\_AIRS + 1; ++i)

        { //爆炸次数记录，并减少

            if (explosions[i].alive)

            {

                --explosions[i].alive;

            }

        }

        //产生新的飞机

        if ((rand() % AIR\_ODDS) == 0)

        {

            createAir();

        }

        //产生新的炮弹

        if (!reloading)

        {

            if (keys[SDLK\_SPACE] == SDL\_PRESSED)

            {

                for (i = 0; i < MAX\_SHOTS; ++i)

                {

                    if (!shots[i].alive)

                    {

                        break;

                    }

                }

                //炮弹移动轨迹

                if (i != MAX\_SHOTS)

                {

                    shots[i].x = cannon.x + (cannon.image->w - shots[i].image->w) / 2;

                    shots[i].y = cannon.y - shots[i].image->h;

                    shots[i].alive = 1;

                }

            }

        }

        reloading = (keys[SDLK\_SPACE] == SDL\_PRESSED);

        cannon.facing = 0;

        //移动大炮

        if (keys[SDLK\_RIGHT])

        {

            ++cannon.facing;

        }

        if (keys[SDLK\_LEFT])

        {

            --cannon.facing;

        }

        cannon.x += cannon.facing \* CANNON\_SPEED;

        //大炮碰撞屏幕边缘处理

        if (cannon.x < 0)

        {

            cannon.x = 0;

        }

        else if (cannon.x >= (screen->w - cannon.image->w))

        {

            cannon.x = (screen->w - cannon.image->w) - 1;

        }

        //移动飞机

        for (i = 0; i < MAX\_AIRS; ++i)

        {

            if (airs[i].alive)

            {

                //计算飞机位移

                airs[i].x += airs[i].facing \* AIR\_SPEED;

                //飞机触发边缘碰撞处理

                if (airs[i].x < 0)

                {

                    airs[i].x = 0;

                    airs[i].y += airs[i].image->h;

                    airs[i].facing = 1;

                }

                else if (airs[i].x + airs[i].image->w >= screen->w)

                {

                    airs[i].x = screen->w - airs[i].image->w - 1;

                    airs[i].y += airs[i].image->h;

                    airs[i].facing = -1;

                }

            }

        }

        //炮弹的移动

        for (i = 0; i < MAX\_SHOTS; ++i)

        {

            if (shots[i].alive)

            {

                shots[i].y -= SHOT\_SPEED; //计算炮弹位移

                if (shots[i].y < 0)

                {

                    shots[i].alive = 0;

                }

            }

        }

        //炮弹击中飞机，计算爆炸参数

        for (j = 0; j < MAX\_SHOTS; ++j)

        {

            for (i = 0; i < MAX\_AIRS; ++i)

            {

                if (shots[j].alive && airs[i].alive && collide(&shots[j], &airs[i]))

                {

                    score += SCORE\_PER\_AIR;

                    airs[i].alive = 0;

                    explosions[i].x = airs[i].x;

                    explosions[i].y = airs[i].y;

                    explosions[i].alive = EXPLODE\_TIME;

                    shots[j].alive = 0;

                    break;

                }

            }

            //得分上限

            if (score >= 1000)

                return;

        }

        //飞机和大炮碰撞的处理

        for (i = 0; i < MAX\_AIRS; ++i)

        {

            if (airs[i].alive && collide(&cannon, &airs[i]))

            {

                airs[i].alive = 0;

                explosions[i].x = airs[i].x;

                explosions[i].y = airs[i].y;

                explosions[i].alive = EXPLODE\_TIME;

                cannon.alive = 0;

                explosions[MAX\_AIRS].x = cannon.x;

                explosions[MAX\_AIRS].y = cannon.y;

                explosions[MAX\_AIRS].alive = EXPLODE\_TIME;

            }

        }

        //画飞机

        for (i = 0; i < MAX\_AIRS; ++i)

        {

            if (airs[i].alive)

            {

                drawObject(&airs[i]);

            }

        }

        //画炮弹

        for (i = 0; i < MAX\_SHOTS; ++i)

        {

            if (shots[i].alive)

            {

                drawObject(&shots[i]);

            }

        }

        //画大炮

        if (cannon.alive)

        {

            drawObject(&cannon);

        }

        //画爆炸

        for (i = 0; i < MAX\_AIRS + 1; ++i)

        {

            if (explosions[i].alive)

            {

                drawObject(&explosions[i]);

            }

        }

        UpdateScreen();

        //按下Esc退出

        if (keys[SDLK\_ESCAPE] == SDL\_PRESSED)

        {

            cannon.alive = 0;

        }

    }

}

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

{

    if (SDL\_Init(SDL\_INIT\_VIDEO) < 0)

    {

        fprintf(stderr, "Couldn't initialize SDL: %s\n", SDL\_GetError());

        exit(2);

    }

    atexit(SDL\_Quit);

    screen = SDL\_SetVideoMode(600, 480, 0, SDL\_SWSURFACE);

    if (screen == NULL)

    {

        fprintf(stderr, "Couldn't set 600x480 video mode: %s\n",

                SDL\_GetError());

        exit(2);

    }

    srand(time(NULL));

    if (loadData() == 1)

    {

        runGame();

        freeObject();

        printScore();

        SDL\_FreeSurface(background);

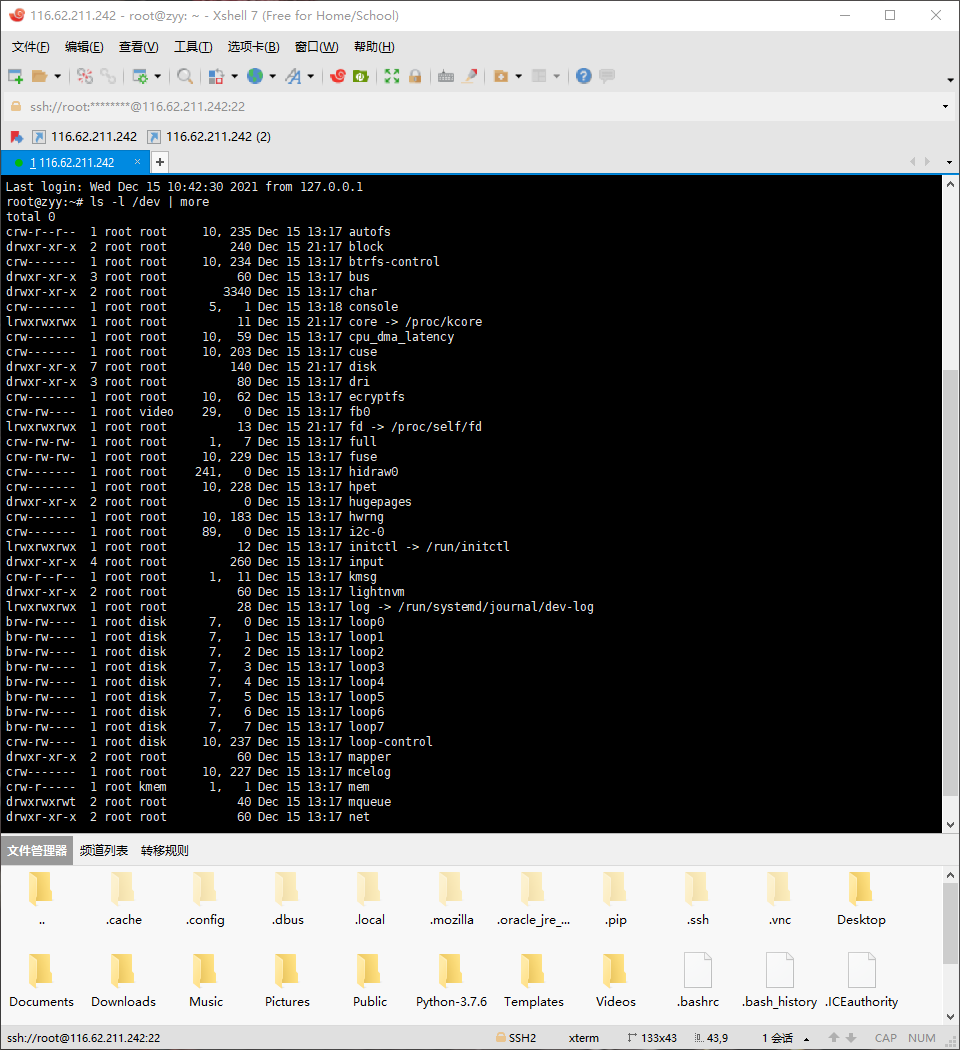
    }

    return 0;

}

任务三：实践P：280 任务1，学习字符设备驱动程序框架、驱动程序的编译、加载与卸载，提示：根据上课讲解的编译方法进行编译。

1. 分为字符设备、块设备、网络设备



3. 代码如下

#include <linux/module.h>

MODULE\_LICENSE("Dua; BSD/GPL");

#include <linux/init.h>

staic int \_\_init hello\_init(void)

{

    printfk("Hello World\n");

    return 0;

}

static void \_\_exit hello\_exit(void)

{

    printf("Goodbye\n");

}

module\_init(hello\_init);

module\_exit(helle\_exit);

6. 使用命令lsmod查看

