链滴

【基于粤嵌 GEC210 开发板的 -- 滑动相册功能】

作者：13097917715

原文链接：https://ld246.com/article/1575267133638

来源网站：链滴

******# 1.准备工作**

我们选用的是.jpg格式的图片，因为每一款图片都有自己的编码格式，而我们的开发板是不认识.jpg式的图片的，因此我们要去JPEG的官网下载相应的资源库jpegsr9c.zip。

点此下载jpegsr9c.zip

2.jpeg库移植

第一步

将下载好的jpegsr9c.zip文件解压，会得到一个.tar.gz的文件，再将这个文件拖到linux系统下，用tar令解压将这个压缩包解压出来。

切换到解压的目录；执行

./configure--prefix=/usr/local/lib CC=arm-linux-gcc --host=arm-linux--enable-shared --enabl-static

第二步

执行

make

可能需要几分钟。

执行

makeinstall

第四步

库的使用

前面几步如果有什么问题请自行google或者阅读解压目录下的**install.txt**文件。

把下面几个文件拷贝到你的程序目录下

jmorecfg.h、jpeglib.h、jerror.h、jconfig.h

这几个文件的位置就在/usr/local/lib下面

到时候在你的源代码中加入

#include "jpeglib.h"

第六步

把/usr/local/lib文件夹下面的libjp.so.9这个动态库文件拷贝到arm板上lib库里。

然后在arm板上执行程序。

3.代码实现

1.在屏幕上显示的过程

display.c

```
#include <stdio.h>
#include <sys/types.h>
```

```c
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>

/*
    函数功能：
        打开lcd设备文件及获取到lcd显存的地址
    参数：
        lcd_fd：以后我们打开文件的文件描述符将会放到这个内存地址中
    返回值：
        成功返回显存的地址，失败返回NULL
*/
unsigned int *open_lcd_device_map_memory(int *lcd_fd)
{
    unsigned int *lcd_fb_addr;

    *lcd_fd = open( "/dev/fb0", O_RDWR);
    if(*lcd_fd == -1)
    {
        perror("open lcd freambuffer file error");
        goto opt_file_err;
    }

    lcd_fb_addr = mmap( NULL, 800*480*4, PROT_READ|PROT_WRITE, MAP_SHARED, *lcd_fd, 0
);
    if(lcd_fb_addr == MAP_FAILED)
    {
        perror("map freambuffer memory error");
        goto map_opt_err;
    }

    return lcd_fb_addr;

map_opt_err:
    close(*lcd_fd);

opt_file_err:
    return NULL;
}

/*
    函数功能：
        关闭lcd设备文件及取消映射lcd显存的地址
    参数：
        lcd_fd：lcd设备文件的文件描述符
        lcd_fb_addr：显存的内存地址
    返回值：
        成功返回0，失败返回-1
*/
int close_lcd_device_unmap_memory(int lcd_fd, unsigned int *lcd_fb_addr)
{
    int retval;
```

```
    retval = munmap( lcd_fb_addr, 800*480*4);
    if(retval == -1)
    {
        perror("unmap freambuffer memory error");
        goto unmap_opt_err;
    }

    retval = close(lcd_fd);
    if(retval == -1)
    {
        perror("close lcd freambuffer file error");
        goto opt_file_err;
    }

    return 0;

unmap_opt_err:
    close(lcd_fd);

opt_file_err:
    return -1;
}

/*
    函数功能：
        在指定的位置将对应的点置上对应的颜色
    参数：
        x：指定的X坐标
        y：指定的Y坐标
        color：显示的颜色
        lcd_fb_addr：显存的内存地址
    返回值：
        无返回值
*/
void lcd_display_point(unsigned int x, unsigned int y, unsigned int color, unsigned int *lcd_fb_
ddr)
{
    *(lcd_fb_addr+800*y+x) = color;
}


/*
    函数功能：
        在指定的位置上显示一个圆
    参数：
        x_cc_cnt：指定圆心的X坐标
        y_cc_cnt：指定圆心的Y坐标
        cc_r：圆的半径
        color：圆的颜色
        lcd_fb_addr：显存的内存地址
    返回值：
        无返回值
*/
```

```
void lcd_display_circle(unsigned int x_cc_cnt, unsigned int y_cc_cnt, unsigned int cc_r, unsigne
 int color, unsigned int *lcd_fb_addr)
{
    unsigned int x, y;

    for(y=0; y<480; y++)
    {
        for(x=0; x<800; x++)
        {
            if((x-x_cc_cnt)*(x-x_cc_cnt)+(y-y_cc_cnt)*(y-y_cc_cnt) <= cc_r*cc_r)
            {
                lcd_display_point(x, y, color, lcd_fb_addr);
            }
        }
    }

}
```

2.改变照片

display_jpeg.c

```
#include <stdio.h>
#include <setjmp.h>
#include <stdlib.h>
#include <jpeglib.h>
#include <jerror.h>

extern JSAMPLE * image_buffer;    /* Points to large array of R,G,B-order data */
extern int image_height;    /* Number of rows in image */
extern int image_width;        /* Number of columns in image */

struct my_error_mgr {
  struct jpeg_error_mgr pub;    /* "public" fields */

  jmp_buf setjmp_buffer;        /* for return to caller */
};

typedef struct my_error_mgr * my_error_ptr;



METHODDEF(void)
my_error_exit (j_common_ptr cinfo)
{
  /* cinfo->err really points to a my_error_mgr struct, so coerce pointer */
  my_error_ptr myerr = (my_error_ptr) cinfo->err;

  /* Always display the message. */
  /* We could postpone this until after returning, if we chose. */
  (*cinfo->err->output_message) (cinfo);

  /* Return control to the setjmp point */
  longjmp(myerr->setjmp_buffer, 1);
```

```
}


/*
 * Sample routine for JPEG decompression.  We assume that the source file name
 * is passed in.  We want to return 1 on success, 0 on error.
 */

/*
    函数功能：
            显示图片
    参数：
            filename：文件名称
            x_s：起始显示x轴
            y_s：起始显示y轴
            lcd_buf_ptr：显存的内存地址
    返回值：
            0：显示失败
 */
int display_format_jpeg(const char *filename, unsigned int x_s, unsigned int y_s, unsigned int *
cd_buf_ptr)
{

    struct jpeg_decompress_struct cinfo;

    struct my_error_mgr jerr;
    FILE * infile;        /* source file */
    int row_stride;       /* physical row width in output buffer */

    if((infile = fopen(filename, "rb")) == NULL)
    {
        fprintf(stderr, "can't open %s\n", filename);
        return 0;
    }

    cinfo.err = jpeg_std_error(&jerr.pub);
    jerr.pub.error_exit = my_error_exit;

    if(setjmp(jerr.setjmp_buffer))
    {
        jpeg_destroy_decompress(&cinfo);
        fclose(infile);
        return 0;
    }

    jpeg_create_decompress(&cinfo);

    jpeg_stdio_src(&cinfo, infile);

    jpeg_read_header(&cinfo, TRUE);

    jpeg_start_decompress(&cinfo);

    row_stride = cinfo.output_width * cinfo.output_components;
```

```
    char *buffer;       /* Output row buffer */

    buffer =(char *)(*cinfo.mem->alloc_sarray)
        ((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);

    unsigned int x, y = y_s;
    unsigned int color;
    char *buf_save = buffer;

    while(cinfo.output_scanline < cinfo.output_height)
    {
        buffer = buf_save;

        jpeg_read_scanlines(&cinfo, (JSAMPARRAY)&buffer, 1);

        for(x=x_s; x<x_s+cinfo.output_width; x++)
        {
            color = buffer[0]<<16 | buffer[1]<<8 |buffer[2];

            lcd_display_point(x, y, color, lcd_buf_ptr);

            buffer+=3;
        }

        y++;
    }



    jpeg_finish_decompress(&cinfo);

    jpeg_destroy_decompress(&cinfo);


    fclose(infile);

    return 1;
}


/*
    函数功能：
            改变显示图片
    参数：
            panduan：判断值，1、4显示上一张，2、3显示下一张
            lcd_buf_ptr：显存的内存地址
            i：照片序号
            jpg_name：照片名称数组
    返回值：
            i：当前显示图片下标
 */
int changge_display_gpeg(int panduan,unsigned int *lcd_fb_addr,int i,char *jpg_name[])
{
```

```
    if(panduan==1||panduan==4)
    {
        i=i-1;
        if(i<0)
            i=4;
        display_format_jpeg( jpg_name[i], 50, 0, lcd_fb_addr);


    }
    else if(panduan==2||panduan==3)
    {
        i=i+1;
        if(i==5)
            i=0;
        display_format_jpeg( jpg_name[i], 50, 0, lcd_fb_addr);

    }
    else
        return i;
    return i;
}
```

3.触控屏的判断

ts.c

```
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/input.h>//需要用到输入子系统的头文件
#include <stdbool.h>//需要用到bool类型，需要导入的头文件
/*
    函数功能：
        打开触摸屏文件
    参数：
        无参数
    返回值：
        成功返回一个触摸屏的文件描述符，失败返回-1
*/
int open_touchscreen_device(void)
{
    int ts_fd;//存放触摸屏的文件描述符

    /*以只读的形式打开触摸屏设备文件*/
    ts_fd = open("/dev/event0", O_RDONLY);
    if(ts_fd == -1)
    {
        perror("open touchscreen device error");
        goto opt_file_err;
```

```
    }

    return ts_fd;

opt_file_err:
    return -1;
}

/*
    函数功能：
        关闭触摸屏文件
    参数：
        ts_fd：触摸屏的文件描述符
    返回值：
        成功返回0，失败返回-1
*/
int close_touchscreen_device(int ts_fd)
{
    int retval;//返回值变量

    retval = close(ts_fd);
    if(retval == -1)
    {
        perror("close touchscreen device error");
        goto opt_file_err;
    }

    return 0;

opt_file_err:
    return -1;
}
/*
    函数功能：
        时刻返回触摸屏的状态值（按下或者松开，对应的坐标）
        一经调用就会在这里监控触摸屏，如果触摸屏有数据，则获取到数据之后直接返回
    参数：
        ts_fd：触摸屏的文件描述符
        x：如果是获取到一对坐标，该函数会将X坐标的值放入x所指向的内存中
        y：如果是获取到一对坐标，该函数会将Y坐标的值放入y所指向的内存中
    返回值：
        返回值有四种可能：
            返回值为0，代表触摸屏被按下了
            返回值为1，代表触摸屏松开触摸了
            返回值为2，代表成功获取到一对坐标，这个时候你就可以开始分析x跟y的坐标是多少了
            返回值为-1，代表出错了
*/
int get_touchscreen_status( int ts_fd, unsigned int *x, unsigned int *y)
{
    bool read_success_flag = false;//判断是否读到一对坐标数据
    ssize_t rd_ret;//接受read函数的返回值
    struct input_event ts_data;//存放读取到的触摸屏数据

    /*一直监控触摸屏*/
```

```
    while(1)
    {
        /*读取触摸屏数据，如果触摸屏没有被触摸，这个函数将会阻塞在这里*/
        rd_ret = read( ts_fd, &ts_data, sizeof(ts_data));
        if(rd_ret == -1)
        {
            perror("get touchscreen data value err");
            break;
        }

        /*判断读到的事件类型是不是坐标类型*/
        if(ts_data.type == EV_ABS)
        {
            /*判断数据值是X坐标数据值吗*/
            if(ts_data.code == ABS_X)
            {
                *x = ts_data.value;//存放x坐标的数据
            }
            /*判断数据值是Y坐标数据值吗*/
            if(ts_data.code == ABS_Y)
            {
                *y = ts_data.value;//存放Y坐标的数据
                read_success_flag = true;//将判断是否读到一对坐标数据置为真，代表读到一对数据
            }

            /*判断是否读到一对数据*/
            if(read_success_flag)
            {
                read_success_flag = false;//数据打印完后，将判断是否读到一对坐标数据置为假

                goto get_abs_status;
            }
        }/*判断读到的事件类型是不是按键类型*/
        else if(ts_data.type == EV_KEY)
        {
            /*如果是触摸屏被按下，value的值会等于1*/
            if(ts_data.value == 1)
            {
                goto touch_status;
            }
            else/*等于0的话就是被松开来*/
            {
                goto release_status;
            }
        }

    }

    return -1;

touch_status:
    return 0;

release_status:
```

```
    return 1;

get_abs_status:
    return 2;

}
/*
    函数功能：
        判断手势：轻触向左、轻触向右、左滑、右滑
    参数：
        retval：状态值1,2,3,4
        x：如果是获取到一对坐标，该函数会将X坐标的值放入x所指向的内存中
        y：如果是获取到一对坐标，该函数会将Y坐标的值放入y所指向的内存中
        old_x：上一次按下x轴的位置
    返回值：
        返回值有四种可能：
            返回值为1，代表触摸屏向左的按钮按下
            返回值为2，代表触摸屏向右的按钮按下
            返回值为3，代表触摸屏被向左滑动
            返回值为4，代表触摸屏被向右滑动
*/
int panduan_ts_lr(int retval, int x, int y, int old_x)
{

    if (retval == 0 && x<50 && x>0 && y>190  &&  y<290)
        {
        perror("111111111111");
        return 1;}
    if (retval == 0 && x>750 && x<800 && y>190  &&  y<290)
        return 2;
    if (retval == 1 && x<old_x-5)
        {
        perror("222222222222");
        return 3;}
    if (retval == 1 && x>old_x+5)
        return 4;
    return 0;

}
```

4.主函数

mian.c

```
#include <stdio.h>
#include <pthread.h>
#include <display.h>
#include <ts.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h> /* superset of previous */
#include <arpa/inet.h>
#include <unistd.h>
```

```
#define    start_display    1        //判断当前屏幕状态
#define    stop_display    0        //判断当前屏幕状态
int i=0;       //控制图片第几张
int lcd_fd;
int ts_fd;
int lcd_fd;
int retval;
int panduan;
unsigned int x, y, old_x;
unsigned int juxing_color;
unsigned int * lcd_fb_addr;
int retval;
unsigned int x, y;
unsigned int * lcd_fb_addr;
char *jpg_name[] = {"1.jpg","2.jpg","3.jpg","4.jpg","5.jpg"};  //照片名称
unsigned int display_flag=start_display;

int main(void)
{
  juxing_color = 0xEEE5DE;
  old_x=0;
  lcd_fb_addr = open_lcd_device_map_memory( &lcd_fd);
  if(lcd_fb_addr == NULL)
     goto opt_lcd_device_err;
  lcd_display_juxing(juxing_color,lcd_fb_addr);//显示矩形和背景色
  display_format_jpeg( jpg_name[i], 50, 0, lcd_fb_addr);    //首先显示第一张图片
  ts_fd = open_touchscreen_device();
  if(ts_fd == -1)
     goto opt_ts_device_err;
  while(1)
  {
     retval = get_touchscreen_status( ts_fd, &x, &y);

     //获取触摸屏状态，有4个状态，0，1，2，-1
     retval = get_touchscreen_status(ts_fd, &x, &y);//获取触摸数据
     panduan=panduan_ts_lr(retval, x, y, old_x);//获取判断值1,2,3,4
     i=changge_display_gpeg(panduan,lcd_fb_addr,i,jpg_name);
     switch(retval)
     {
        //当retval（触摸屏状态）为0时：代表触摸屏被按下了
        case 0:
           old_x=x;
           printf("start touch\n");
           display_flag = 0;
           break;
        //当retval（触摸屏状态）为1时：代表触摸屏松开触摸了
        case 1:
           printf("release touch\n");
           display_flag = 1;
           break;
        //当retval（触摸屏状态）为2时：     代表成功获取到一对坐标，这个时候你就可以开始分析x
y的坐标是多少了
        case 2:
```

```
                display_flag = 0;
                break;
            //当retval（触摸屏状态）为-1时：代表出错了
            default:
                goto get_ts_status_err;
        }

    }

    retval = close_lcd_device_unmap_memory( lcd_fd, lcd_fb_addr);
    if(retval == -1)
        goto opt_lcd_device_err;

    return 0;

get_ts_status_err:
    close_touchscreen_device(ts_fd);
opt_ts_device_err:
    close_lcd_device_unmap_memory( lcd_fd, lcd_fb_addr);
opt_lcd_device_err:
    return -1;
}
```

4.编译代码

另外编译的时候请一定使用下面的方法

执行

arm-linux-gcc -o 你的输出 你的程序 -L/usr/local/lib-l:libjpeg.so.9

上面-L后面的路径可以根据自己目录更改。

“编译时千万别以为加入了-ljpeg参数就可以了，老问题，编译器会给你链接libjpeg6的库，要指定库l：libjpeg.so.8，如果找不到，前面再指定库目录-L/你libjpeg的安装目录/lib，这样就一切完美了”