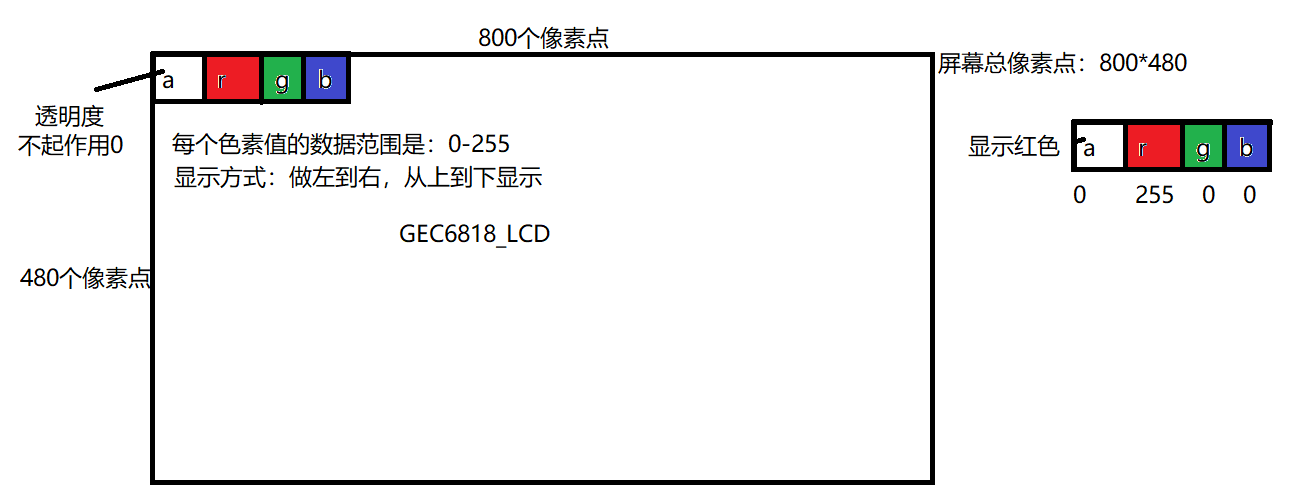
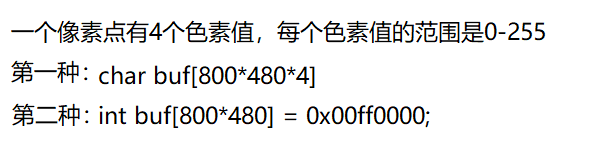
**一、显示颜色**



显示操作步骤：

1、打开文件（open）：显示屏的文件路径/dev/fb0

2、准备颜色数据



3、写入数据：write

4、关闭文件close

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

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

{

    //第一步：打开显示屏

    int lcd\_fd = open("/dev/fb0", O\_RDWR);

    if (lcd\_fd < 0)

    {

        printf("显示屏打开失败\n");

        return -1;

    }

    //第二步：准备颜色数据, 并且对其进行赋值

    int color[800\*480];

    for (int i = 0; i < 800\*480; i++)

        color[i] = 0x00ff0000;

    // 第三步：将颜色数据写入到屏幕来

    write(lcd\_fd, color, sizeof(color));

    // 第四步：关闭设备

    close(lcd\_fd);

    return 0;

}

练习1：先显示一种颜色

练习2：每隔一秒钟显示一种颜色（光标偏移）

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <sys/mman.h>

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

{

    //第一步：打开显示屏

    int lcd\_fd = open("/dev/fb0", O\_RDWR);

    if (lcd\_fd < 0)

    {

        printf("显示屏打开失败\n");

        return -1;

    }

    // 进行内存映射

    int \*p = mmap(NULL, 800\*480\*4, PROT\_READ|PROT\_WRITE, MAP\_SHARED,lcd\_fd, 0);

    if (p == NULL)

    {

        printf("内存映射失败\n");

        return -1;

    }

    while (1)

    {

        // 将光标偏移到文件的开头

        lseek(lcd\_fd, 0, SEEK\_SET);

        // 获取随机数

        srand(time(NULL));

        int random = rand();

        // 写入颜色数据

        //write(lcd\_fd, color, sizeof(color));

        for (int i = 0; i < 800\*480; i++)

            \*(p+i) =  random;

        sleep(1);

    }

    return 0;

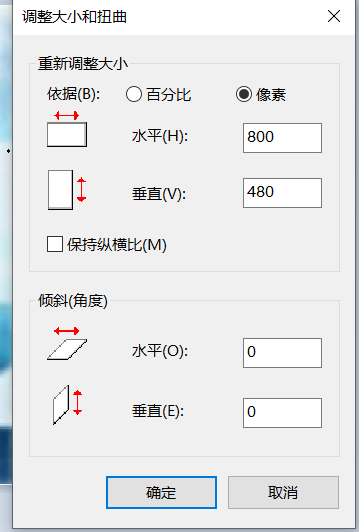
}

作业：显示一个彩虹色



**二、显示图片**

1、修改图片的大小为800\*480，用电脑自带的画图软件去修改，点击重新调整大小，以像素的方式进行设置，取消勾选保持横纵比，水平设置800，垂直设置成480

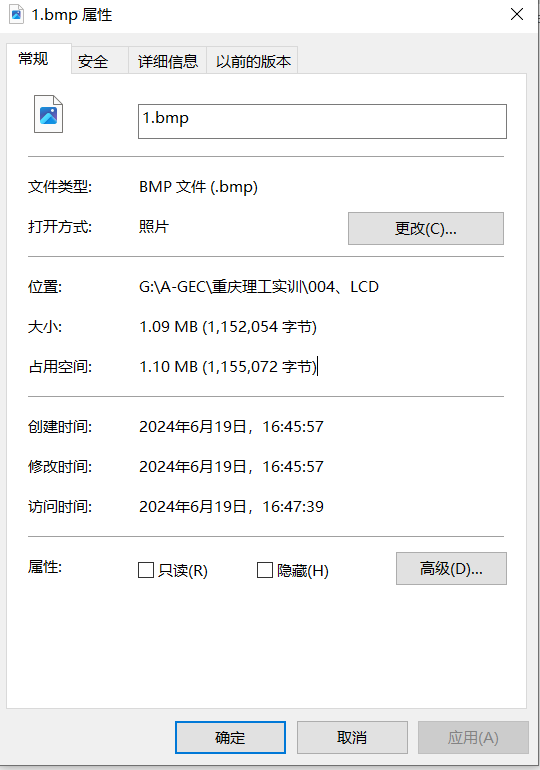


2、将图像另存为BMP格式（注意必须通过画本另存为保存，不能直接更改文件的后缀名为BMP，否则会导致图片的格式不正确）



3、BMP图片解析

一张800\*480大小的图片，一个像素点占3个字节（rgb），图片的大小800\*480\*3=1152000字节，



由图可知，图片实际大小比计算出的1152000要多54字节，多出来的54字节是图片的信息头，里面包含了该图片的一些基本信息（高、宽）

实际大小 = 颜色数据+图片信息头 = 1152000+54 = 1152054字节

显示图片的步骤：

1、打开显示屏LCD

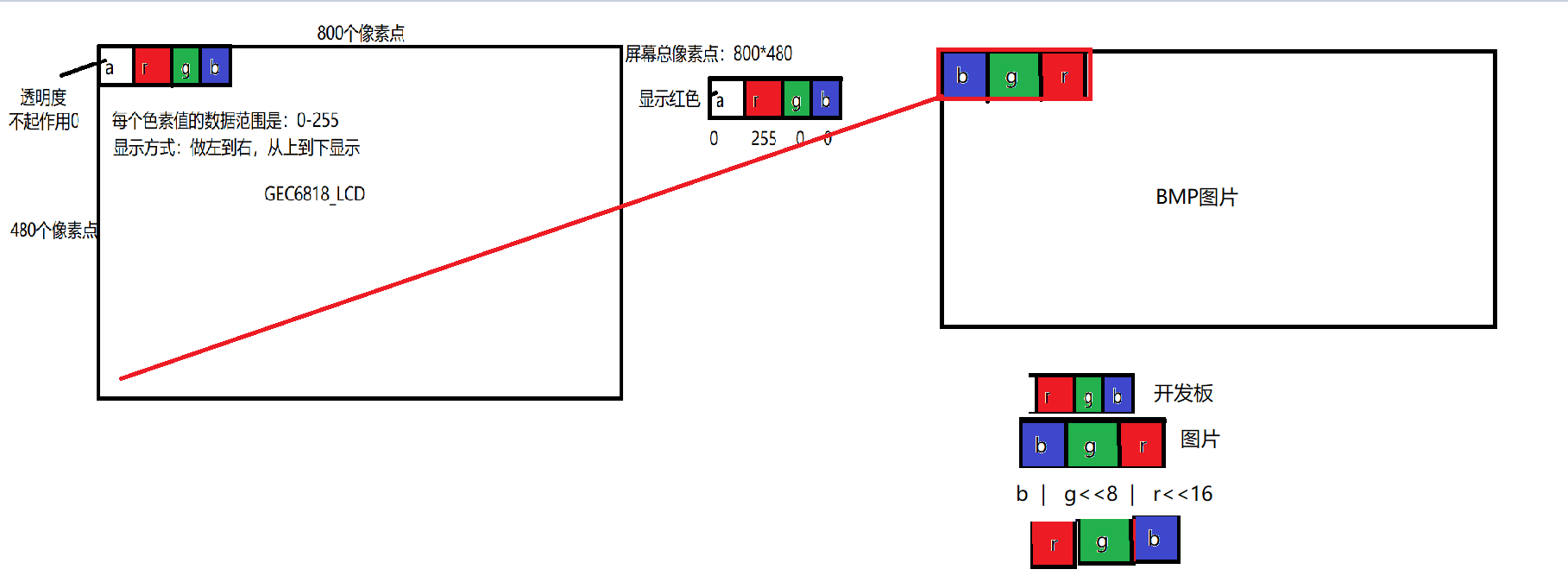
2、打开图片

3、偏移图片前54字节 lseek

4、读取颜色数据

5、将颜色数据写入到开发板

6、关闭文件



#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <sys/mman.h>

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

{

    //第一步：打开显示屏

    int lcd\_fd = open("/dev/fb0", O\_RDWR);

    if (lcd\_fd < 0)

    {

        printf("显示屏打开失败\n");

        return -1;

    }

    // 进行内存映射

    int \*p = mmap(NULL, 800\*480\*4, PROT\_READ|PROT\_WRITE, MAP\_SHARED,lcd\_fd, 0);

    if (p == NULL)

    {

        printf("内存映射失败\n");

        return -1;

    }

    // 打开图片

    int bmp\_fd = open("./1.bmp", O\_RDWR);

    if (bmp\_fd < 0)

    {

        printf("图片打开失败\n");

        return -1;

    }

    // 偏移图片前54字节

    lseek(bmp\_fd, 54, SEEK\_SET);

    // 读取颜色数据

    char color[800\*480\*3];

    read(bmp\_fd, color, sizeof(color));

    // 显示图片

    int i = 0, value = 0;

    for (int y = 0; y < 480; y++) // 外层循环来控制行数

    {

        for (int x = 0; x < 800; x++) // 内层循环来控制列数

        {

            char b = color[i++];

            char g = color[i++];

            char r = color[i++];

            value = b | g<<8 | r<<16;

            // \*(p+(800\*y+x)) = value;

            \*(p+(800\*(479-y)+x)) = value;

        }

    }

    // 关闭文件

    close(lcd\_fd);

    close(bmp\_fd);

    return 0;

}

作业：自己去显示一张图片，上传到群文件夹003-6.19作业

**三、显示任意大小的图片**

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <sys/mman.h>

int show\_any\_bmp(char \*filename, int \*p, int x\_offset, int y\_offset)

{

    // 打开图片

    int bmp\_fd = open(filename, O\_RDWR);

    if (bmp\_fd < 0)

    {

        printf("图片打开失败\n");

        return -1;

    }

    //读取图片前54字节的信息头

    char head[54];

    read(bmp\_fd, head, 54);

    // 计算图片的宽度与高度,18-21字节表示宽度，22-25字节表示图片的高度

    // 18字节是最低为，21字节是最高位

    //printf("wide = %d high = %d\n", wide, high);

    if (x\_offset+wide > 800 || y\_offset + high > 480)

    {

        printf("图片越界\n");

        return -1;

    }

    // 读取剩下的颜色数据

    char buf[wide\*high\*3];

    int num = (wide\*3)%4;   // 判断图片的宽度是不是4的整数倍

    if (num == 0)   // 是4的倍数, 全部读取

    {

        read(bmp\_fd, buf,sizeof(buf));

    }

    else    // 不是4的倍数，一行一行的去读

    {

        for(int i = 0; i < high; i++)

        {

            read(bmp\_fd, buf+i\*wide\*3, wide\*3); // 读取一行

            lseek(bmp\_fd, 4-num,SEEK\_CUR);  // 便宜掉系统自动补齐的数据

        }

    }

    // 显示图片

    int i = 0, value = 0;

    for (int y = 0; y < high; y++) // 外层循环来控制行数

    {

        for (int x = 0; x < wide; x++) // 内层循环来控制列数

        {

            char b = buf[i++];

            char g = buf[i++];

            char r = buf[i++];

            value = b | g<<8 | r<<16;

            // \*(p+(800\*y+x)) = value;

            //图片的偏移量，纵轴偏移量加上图片本身的高度不能大于480

            //横轴偏移量加上图片本身的宽度不能大于800，否则有可能会出现段错误

            \*(p+(800\*(high-1-y+y\_offset)+x+x\_offset)) = value;

        }

    }

    // 关闭设备

    close(bmp\_fd);

}

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

{

    //第一步：打开显示屏

    int lcd\_fd = open("/dev/fb0", O\_RDWR);

    if (lcd\_fd < 0)

    {

        printf("显示屏打开失败\n");

        return -1;

    }

    // 进行内存映射

    int \*p = mmap(NULL, 800\*480\*4, PROT\_READ|PROT\_WRITE, MAP\_SHARED,lcd\_fd, 0);

    if (p == NULL)

    {

        printf("内存映射失败\n");

        return -1;

    }

    show\_any\_bmp("./1.bmp", p, 100, 100);

    munmap(p, 800\*480\*4);

    close(lcd\_fd);

    return 0;

}

练习：显示一个gif动图