**一、系统IO控制LCD液晶屏**

我们前面一直使用系统IO接口操作普通文本，但是在linux中，一切皆为文件。所以我们也可以通过系统IO接口访问硬件设备对应的文件，间接操作硬件设备。比如说我们开发板的液晶屏设备，通过系统IO接口访问液晶屏设备文件**（/dev/fb0）**可以控制液晶屏的数据显示。

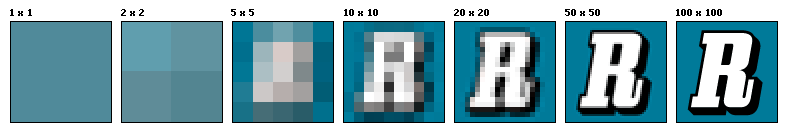
open("**/dev/fb0**");

**1、液晶屏的基本概念**

分辨率：**800\*480**分辨率

宽、高两个维度上的像素点数目。

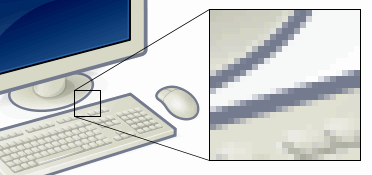
分辨率越高，所需要的显存越大。



像素：BMP（RGB）3字节 开发板像素点：（ARGB）4字节

屏幕上显示颜色的最小单位，英文叫 pixel。注意，位图（如jpg、bmp等格式的常见图片）也是由一个个的像素点构成的，跟屏幕的像素点的概念一样。**原理上讲，将一张位图显示到屏幕上，就是将图片上的像素点一个个复制到屏幕像素点上。**

**思路：将每个像素点都写入颜色代码即可显示整个开发板的颜色**



色深：。

每个像素所对应的内存字节数，一般有8位、16位、24位或32位

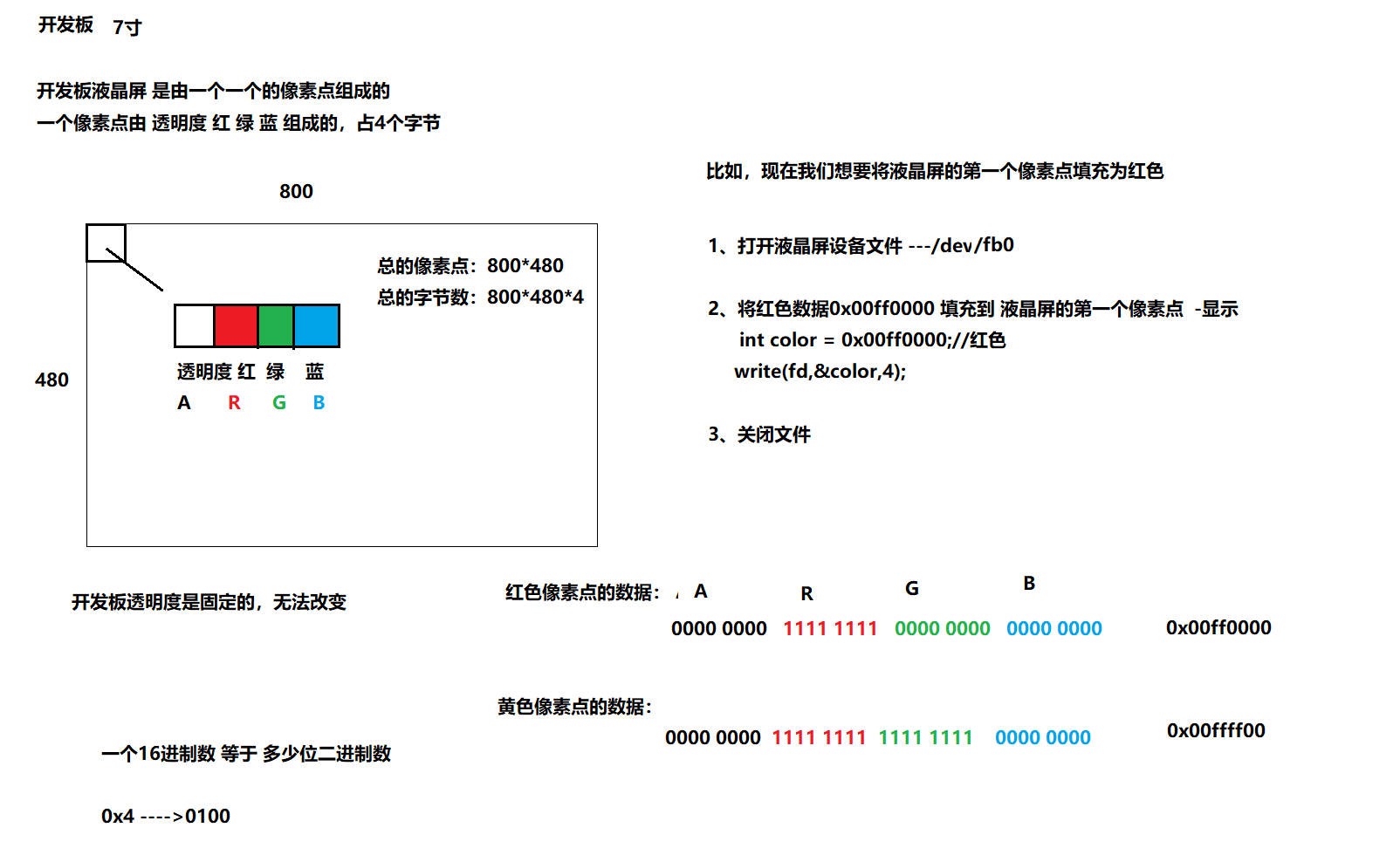
GEC6818开发板的屏幕的色深是32位的（ARGB）4字节

32位色深的屏幕一般被称为真彩屏，或1600万色屏。

色深决定了一个像素点所能表达的颜色的丰富程度，色深越大，色彩表现力越强。

**2、液晶屏参数**

已知lcd液晶设备的名字，就可以使用open()去访问该设备，如果我们要写一些数据（颜色）进去，那么首先必须要先知道屏幕的一些参数。



1）屏幕尺寸：7寸

2）在我们开发板的LCD屏上，一共由800\*480个像素点组成，每个像素点是4个字节，像素点是由透明度+红色值+绿色值+蓝色值组成的

比如：

显示红色：0x00ff0000（这些数值可以通过画图工具和计算器计算得出）

显示绿色：0x0000ff00

显示蓝色：0x000000ff

**3、案例**

屏幕的第一个像素点显示为红色。

//任务：将一个红色（写入）填充到液晶屏的第一个像素

int main()

{

//红色

int color = 0x00ff0000;

//1、打开液晶屏文件/dev/fb0，

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

if(lcdfd == -1)

{

printf("open lcd error\n");

return -1;

}

//2、将颜色0x00ff0000 写入 液晶屏文件中

write(lcdfd,&color,4);

//3、关闭文件

close(lcdfd);

}

思考： 如何显示满屏绿色。

#include <stdlib.h>

#include <stdbool.h>

#define GREEN 0x0000ff00//绿色

int main()

{

//1、打开液晶屏文件/dev/fb0，

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

if(lcdfd == -1)

{

printf("open lcd error\n");

return -1;

}

//2、先准备800\*480这么大的颜色数据

int colorbuf[800\*480]={0};

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

{

colorbuf[i] = GREEN;

}

//3、一次性将准备好的满屏绿色数据 写入 液晶屏文件中

write(lcdfd,colorbuf,800\*480\*4);

//4、关闭文件

close(lcdfd);

}

练习1：在开发板上显示一张德国国旗或者3种以上颜色

练习2：在开发板上显示一张法国国旗



答案：

练习1：

#include<stdio.h>

#include<stdbool.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

int main()

{

    //1、打开液晶屏文件  必须是以可读可写 O\_RDWR

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

    if(fd == -1){

        perror("open lcd error");

        return -1;

    }

int color1 = 0x000000;

    int color2 = 0xff0000;

    int color3 = 0xffff00;

    for(int i=0;i<800\*160;i++)

        write(fd,&color1,4);//一次只显示一个像素点

    for(int i=0;i<800\*160;i++)

        write(fd,&color2,4);//一次只显示一个像素点

    for(int i=0;i<800\*160;i++)

        write(fd,&color3,4);//一次只显示一个像素点

    //4、关闭

    close(fd);

    return 0;

}

练习2：

//法国国旗

int color1 = 0x0000ff;

    int color2 = 0xffffff;

    int color3 = 0xff0000;

    for(int y=0;y<480;y++)

    {

        for(int i=0;i<267;i++)

            write(fd,&color1,4);//一次只显示一个像素点

        for(int i=0;i<266;i++)

            write(fd,&color2,4);//一次只显示一个像素点

        for(int i=0;i<267;i++)

            write(fd,&color3,4);//一次只显示一个像素点

    }

**4、bmp图片编程**

任务：在开发板上显示一张800\*480的24位bmp图片(大小：1152054)

1、思路

1）、打开液晶屏文件open("/dev/fb0")

2）、打开bmp图片文件open("bmp");

3）、读取bmp图片的数据

4）、将读取出来的图片数据写入到液晶屏上

5）、关闭液晶屏文件、bmp图片文件

问题一：图片颜色和大小对不上

原因：像素点少了一个字节

解决：像素点添加一个字节

问题二：图片上下颠倒

原因：图片和液晶屏的扫瞄方式是相反的，所以图片的下发像素点在液晶屏上方显示

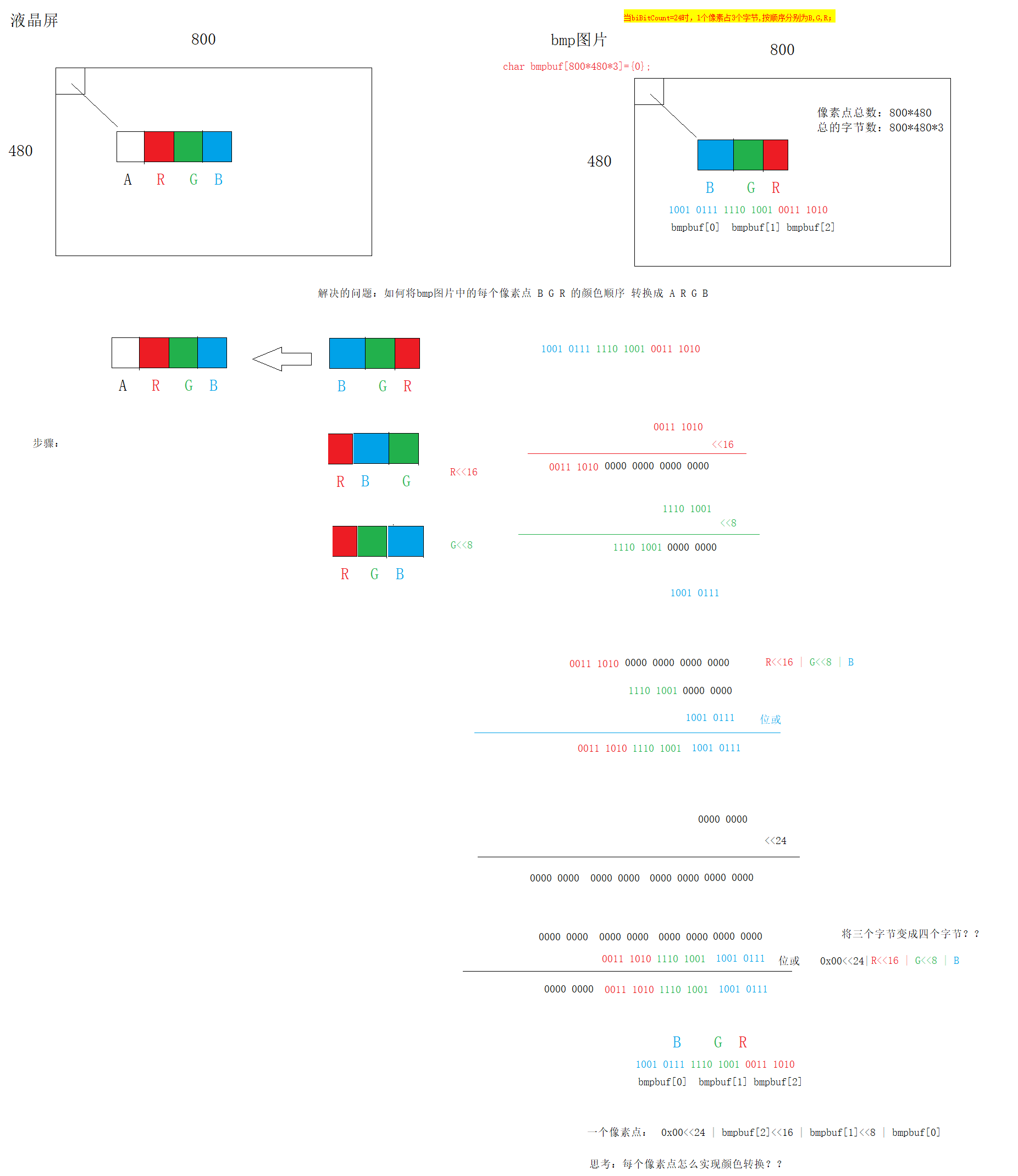
解决：提前将图片垂直翻转

问题三：图片左边突出

原因：因为多出了54个字节（保存图片的信息）

解决：跳过这54个字节

2、分析bmp图片的格式



int bmp\_show(const char\*fileName)

{

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

    if(lcd\_fd < 0)

    {

        printf("打开LCD失败\n");

        return -1;

    }

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

    if(bmp\_fd < 0)

    {

        printf("打开BMP失败\n");

        return -1;

    }

    //存放图片数据

    char buf[800\*480\*3]={0};

    char bmp[800\*480\*4]={0};

    char bmp1[800\*480\*4]={0};

    //将图片所有数据读取出来存放到buf内

    read(bmp\_fd,buf,800\*480\*3);

    //i+=3:表示一个像素点3个字节；j+=4：表示一个像素点4个字节

    for(int i=0,j=0;i<800\*480\*3;i+=3,j+=4)//每个像素点补充一个字节

    {

        bmp[j]=buf[i];

        bmp[j+1]=buf[i+1];

        bmp[j+2]=buf[i+2];

        bmp[j+3]=0;

    }

    for(int y=0;y<480;y++)//图片上下数据交换

    {

        for(int x=0;x<800\*4;x++)

        {

            //(479-y):表示最后一行，\*800\*4：表示每行的字节，+x：表示该行的哪个字节

            bmp1[y\*800\*4+x]=bmp[(479-y)\*800\*4+x];

        }

    }

    //将图片所有数据写入到液晶屏上

    write(lcd\_fd,bmp1,800\*480\*4);

    close(lcd\_fd);

    close(bmp\_fd);

    return 0;

}