

## 嵌入式系统仿真实验第八讲

上一次我们做了图形用户接口驱动的实验，可以在模拟的 LCD 上显示图像，但是我们的菜单还是通过串口 0 打印出来的，有没有办法在 LCD 上显示字符呢，或者想 windows、android 等操作系统一样 GUI 操作界面呢？

今天的实验，我们来看如何在 LCD 终端界面上显示字符以及模拟的串口模拟键盘输入字符。

### 1 .LCD 字符显示驱动\*

要显示字符，实际就是如何将字符在 LCD 上画出来而已，LCD 是一种点阵显示设备，上次我们已经做过实验了，我们测试了 LCD 的驱动，然后在内存中映射了 LCD 的显示 framebuffer，也就是那个 fb，然后我们对 fb 进行操作就是相当于是在 LCD 上画图一样。明白了这个原理，要显示字符就比较简单了，首先我们需要知道一个字符它的点阵信息，然后读取这个点阵信息，把它按需要显示的坐标在 fb 上画出来就是了。字符点阵可以从一些操作系统或者输入法里面去提取出来，字符文件一般分为点阵字符文件和矢量字符文件，目前我们还是 bare metal programming(中文大家习惯说成裸机编程)，所以我们这门课只给大家演示点阵字符的显示原理。

为了显示文本，我们可以找到一个 ASCII 字符的原始位图字体文件 font.bin，包含 128 个 ASCII 字符的点阵位图，其中每个字符用 8x16 的点阵位图表示，每个字符占据 16 个字节，每个字节表示字符点阵中的一行像素点。可以使用字符的 ASCII 码值（乘以 16）作为偏移量，来访问该字符在位图文件中的字节地址。然后扫描每个字节中的位。对于每一个 0 的位，表示这一点是黑点，将 BGR = 0x00000000 (black) 写入 fb 相应的像素点。对于每个 1 的位，表示这一点是黑点，将 BGR = 0x00FFFFFF (白色) 写入 fb 相应的像素点，就可以完成一个字符在 fb 上的显示。当然也可以将不同的 RGB 值（顺序按 BGR 赋值）写入到 fb 的像素点，用彩色显示每个字符。前面我们是把图像文件包含在可执行文件中，这里也可以把原始字体文件作为二进制数据部分放入，另一种方法是先将位图转换为 char 映射。例如，我们可以编写下面的 C 语言程序将字体位图转换为字符映

射。

```
/**** bitmap2charmap.c: run as a.out font.bin > font ****/
#include <stdio.h>
main(int argc, char *argv[ ])
{
    int i, n; u8 buf[16];
    FILE *fp = fopen(argv[1], "r"); // fopen file for READ
    while((n = fread(buf, 1, 16, fp)){ // read 16 bytes
        for (i=0; i<n; i++) // write each byte as 2 hex
            printf("0x%2x ", buf[i]);
    }
    printf("\n");
}
```

明白这个道理后就非常简单了,先写个在 fb 的(x,y)坐标点画点和擦除的函数:

```
int clr_pix(int x, int y)
{
    int pix = y*640 + x;
    fb[pix] = 0x00000000; //clean a pixel at point(x,y) we defined BLACK
    is default color
}

int set_pix(int x, int y)
{
    int pix = y*640 + x;
    fb[pix] = color;
}
```

我们用黑色去擦除位点,而 color 是一个整形数据,按照 BGR 的顺序三种色调各占 8 位来分配,比如 0x00FFFFFF (白色), 0x00FF0000 (蓝色), 0x0000FF00 (绿色)、0x000000FF (红色)。

LCD 屏幕可视为由 480x640 像素组成的矩形框。每个像素在屏幕上都有一个 (x, y) 坐标。相应地,帧缓冲区 u32 fb[]是一个包含 480 \* 640 的 32 位 (u32) 整数存储区域,其中每个整数的低 24 位表示像素的 BGR 值。通过 Mailman 算法 (第 2 章, Wang 2015\*) 给出 fb[]中像素在 (x, y) = (列, 行) 坐标处的线性地址或索引, 见公式 1。

$$\text{pixel\_index} = x + y * 640 \quad (1)$$

下面我们接着直接参考 WSU 华盛顿州立大学的 K.C. Wang 的教材，给出画字符和滚动光标的代码：

```
int draw_char(unsigned char c, int x, int y)
{
    int r, bit;
    unsigned char *caddress, byte;

    caddress = font + c*16;
    // printf("c=%x %c caddr=%x\n", c, c, caddress);

    for (r=0; r<16; r++){
        byte = *(caddress + r);

        for (bit=0; bit<8; bit++){
            if (byte & (1<<bit))
                set_pix(x+bit, y+r);
        }
    }
}

int clr_char(unsigned char c, int x, int y)
{
    int row, bit;
    unsigned char *caddress, byte;

    caddress = font + c*16;
    // printf("c=%x %c caddr=%x\n", c, c, caddress);

    for (row=0; row<16; row++){
        byte = *(caddress + row);

        for (bit=0; bit<8; bit++){
            if (byte & (1<<bit))
                clr_pix(x+bit, y+row);
        }
    }
}

int draw_string(char *s, int x, int y)
{
    while(*s){
        draw_char(*s, x, y);
        x+=8;
    }
}
```

```

        s++;
    }
}

int scroll()
{
    int i;
    for (i=scrow_row*16*640; i<640*480-640*16; i++){
        fb[i] = fb[i+ 640*16];
    }
}

int kprt_char(char c, int ro, int co)
{
    int x, y;
    x = co*8;
    y = ro*16;
    //printf("c=%x [%d%d] (%d%d)\n", c, ro,co,x,y);
    draw_char(c, x, y);
}

int erase_kprt_char(char c, int ro, int co)
{
    int x, y;
    x = co*8;
    y = ro*16;
    //printf("c=%x [%d%d] (%d%d)\n", c, ro,co,x,y);
    clr_char(c, x, y);
}

int erasechar()
{
    // erase char at (row,col)
    int r, bit, x, y;
    unsigned char *caddress, byte;

    x = col*8;
    y = row*16;

    //printf("ERASE: row=%d col=%d x=%d y=%d\n",row,col,x,y);

    for (r=0; r<16; r++){
        for (bit=0; bit<8; bit++){

```

```

        clr_pix(x+bit, y+r);
    }
}

int clr_cursor()
{
    erase_kprt_char(cursor, row, col);
}

int put_cursor(unsigned char c)
{
    kpert_char(c, row, col);
}

int kputc(char c)
{
    clr_cursor();
    if (c=='\r'){
        col=0;
        //printf("row=%d col=%d\n", row, col);
        put_cursor(cursor);
        return;
    }
    if (c=='\n'){
        row++;
        if (row>=25){
            row = 24;
            scroll();
        }
        //printf("row=%d col=%d\n", row, col);
        put_cursor(cursor);
        return;
    }
    if (c=='\b'){
        if (col>0){
            col--;
            erasechar();
            put_cursor(cursor);
        }
        return;
    }
    // c is ordinary char
    kpert_char(c, row, col);
}

```

```

col++;
if (col>=80){
    col = 0;
    row++;
    if (row >= 25){
        row = 24;
        scroll();
    }
}
put_cursor(cursor);
//printf("row=%d col=%d\n", row, col);
}

```

```

int kprints(char *s)
{
    while(*s){
        kputc(*s);
        s++;
    }
}

```

```

int krpx(int x)
{
    char c;
    if (x){
        c = tab[x % 16];
        krpx(x / 16);
    }
    kputc(c);
}

```

```

int kprintx(int x)
{
    kputc('0');
    kputc('x');
    if (x==0)
        kputc('0');
    else
        krpx(x);
    kputc(' ');
}

```

```

int krpu(int x)
{

```

```

    char c;
    if (x){
        c = tab[x % 10];
        krpu(x / 10);
    }
    kputc(c);
}

int kprintu(int x)
{
    if (x==0){
        kputc(' ');
        kputc('0');
    }
    else
        krpu(x);
    kputc(' ');
}

int kprinti(int x)
{
    if (x<0){
        kputc('-');
        x = -x;
    }
    kprintu(x);
}

int kprintf(char *fmt,...)
{
    int *ip;
    char *cp;
    cp = fmt;
    ip = (int *)&fmt + 1;

    while(*cp){
        if (*cp != '%'){
            kputc(*cp);
            if (*cp=='\n')
                kputc('\r');
            cp++;
            continue;
        }
        cp++;

```

```

switch(*cp){
    case 'c': kputc((char)*ip);    break;
    case 's': kprints((char *)*ip); break;
    case 'd': kprinti(*ip);        break;
    case 'u': kprintu(*ip);        break;
    case 'x': kprintx(*ip);        break;
}
cp++; ip++;
}
}

```

在上面的示例中，我们按照 K.C. Wang 的教材还实现了类似 C 语言库函数的 `printf` 函数里面的通配符%的用法。同样的原理也在上次那个串口的打印函数里面也实现了 `printf` 函数通配符%的功能。

## 2 LCD 显示字符测试实验

这次我们实现一个模拟串口输入，LCD 应答的聊天机器人程序，当然目前还是 bare metal programming 阶段，所以没有聊天机器人的算法，大家如果对这个感兴趣，可以在后面和我一起探讨，如何开发聊天机器人，包括语义理解、知识图谱、深度神经网络、对抗生成网络等，目前无特定目的的聊天机器人还处于初级阶段，但特定领域的基于案例推理的聊天机器人目前已经在路线导航、导购、语音客服、智能导医等方面有了较好的商业应用。

我们在主程序 `switch` 循环里加个菜单，用户按't'进入机器人聊天。

```

case 0x74:
    uprintf(up, "\n\r");
    while(1){
        robot_walking();
        set_color(GREEN);

        uprintf(up, "I : ");
        upgets(up, mtext);

        kprintf("You said : ");
        uprintf(up, "\n");
        set_color(RED);
        kprintf("%s\n", mtext);
        if(my_strcmp(mtext, "bye")==0){
            kprintf("Robot: bye-bye\n");

```



```

        uprintf(up, "Robot: bye-bye\n");
        break;
    }
    ret = search_answer(mtext, answer);
    if(ret){
        kprintf("Robot: %s\n", answer);
        uprintf(up, "Robot:%s\n", answer);
    }else{
        kprintf("Robot: %s\n", "haha");
        uprintf(up, "Robot:%s\n", "haha");
    }
}
break;

```

其中，search\_answer 为机器人搜索回答的算法，其实我们说人工智能的最基础就是搜索算法，这个搜索算法决定了你的机器人的智能程度。目前我们只是简单地 demo 一下：

```

int search_answer(char *que, char *ans)
{
    char *index1 = que;
    char *index2 = ans;

    if (my_strcmp(index1, "hello")==0)
    {
        my_strcpy("you are welcome!", index2);
        return 1;
    }
    if (my_strcmp(index1, "who are you")==0)
    {
        my_strcpy("I am wuster.", index2);
        return 1;
    }
    if (my_strcmp(index1, "what's up")==0)
    {
        my_strcpy("Not much just busy.", index2);
        return 1;
    }
    if (my_strcmp(index1, "who is Donald Trump")==0)
    {
        my_strcpy("BRD, a joker!", index2);
        return 1;
    }
}

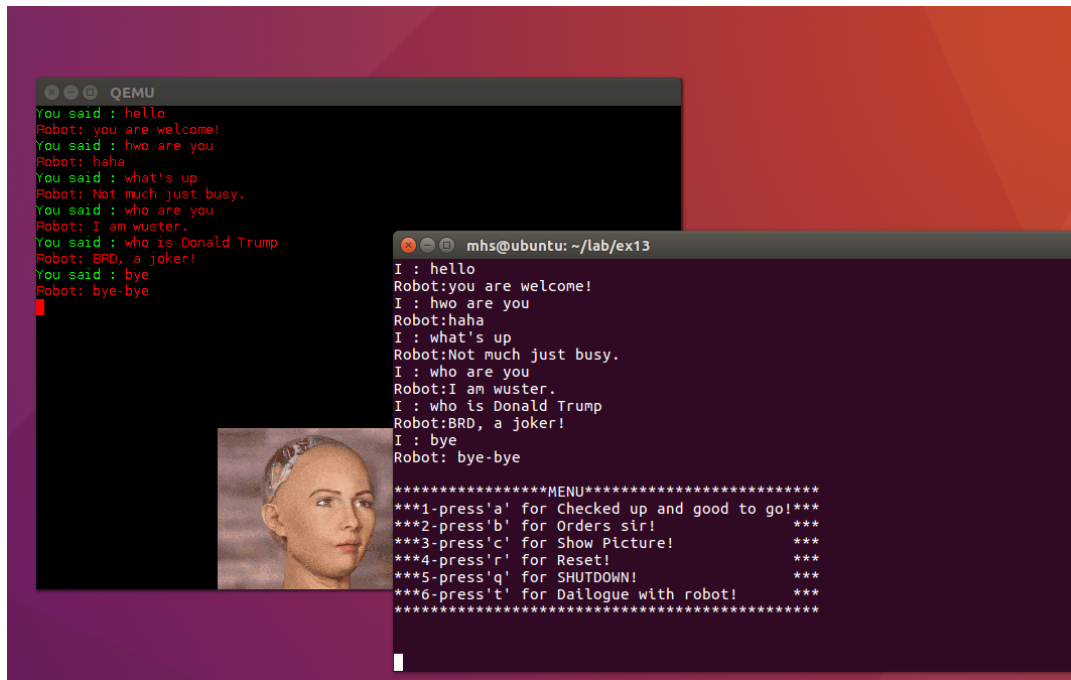
```

```
return 0;

}
```

Makefile 文件没有什么特别的，由于我们直接在 main.c 里面#include “vic.c”了，所以我们根本就没改它。

下面运行编译测试：



```
QEMU
You said : hello
Robot: you are welcome!
You said : hwo are you
Robot: haha
You said : what's up
Robot: Not much just busy.
You said : who are you
Robot: I am wuster.
You said : who is Donald Trump
Robot: BRD, a joker!
You said : bye
Robot: bye-bye
```

```
mhs@ubuntu: ~/lab/ex13
I : hello
Robot:you are welcome!
I : hwo are you
Robot:haha
I : what's up
Robot:Not much just busy.
I : who are you
Robot:I am wuster.
I : who is Donald Trump
Robot:BRD, a joker!
I : bye
Robot: bye-bye

*****MENU*****
***1-press'a' for Checked up and good to go!***
***2-press'b' for Orders sir!
***3-press'c' for Show Picture!
***4-press'r' for Reset!
***5-press'q' for SHUTDOWN!
***6-press't' for Dailogue with robot!
```

图 1 运行测试图

抱歉，今天人很不舒服，不是因为新冠，而是花粉过敏期到了，每年这个时候我都要像死过去一般的难受，今年因为新冠，所以基本没出门，刚也是带了两层口罩出门去拿物质，现在我休息一会，今天的实验讲义比较粗糙了，请大家原谅，直接读代码吧，相信如果你一步步和我学过来的，今天的实验应该比较好理解，今天本来还要讲中断和键盘驱动的，留到下次吧。

注：本实验教程为武汉科技大学机器人与智能系统研究院闵华松老师的网络课程教学文档，可以复制，不做商业用途。

\* K.C. Wang, [kwang@eecs.wsu.edu](mailto:kwang@eecs.wsu.edu), 本次代码参考了 WSU 的 K.C. Wang 的教材：

[1] Wang K C. Embedded real-time operating systems[M]. Embedded and Real-Time Operating Systems. Springer, Cham, 2017: 401-475.