**字库使用**

**1.初始化字库**

font \*fontLoad(char \*fontPath);

参数一：DroidSansFallback.ttf 字库的路径

返回值：操作字库的句柄

// 打开字体

**font \*f = fontLoad("simkai.ttf");**

**2.设置字体的大小**

void fontSetSize(font \*f, s32 pixels)

参数一：操作的字库

参数二：字体的大小

**fontSetSize(f, 32);//32:长和宽32像素点**

**3.设置字体输出框的大小**

bitmap \*createBitmapWithInit(u32 width, u32 height, u32 byteperpixel，int color);

参数一：输出框的宽 0~800

参数二：输出框的高 0~480

参数三：当前屏幕的色素 例如：32位-》4

参数四：输出框的颜色

**bitmap \*bm = createBitmapWithInit(200, 100, 4,getColor(0,255,255,255));**  //白色

**4.把字体输出到输出框中**

void fontPrint(font \*f, bitmap \*screen, s32 x, s32 y, char \*text, color c, s32 maxWidth)

参数一：操作的字库

参数二：输出框

参数三：字体的位置 X轴

参数四：字体的位置 Y轴

参数五：字体文本（输出内容）

参数六：字体颜色

参数七：一行显示字体的最大宽度 （例如：字体大小为32，假设设置为64则一行显示两个字体，就换行显示）

默认为： 0

**fontPrint(f,bm,0,0,buf,getColor(0,0,0,0),0);**

**5.把输出框的所有信息显示到LCD屏幕中**

void show\_font\_to\_lcd(unsigned int \*p,int px,int py,bitmap \*bm)

参数一：LCD映射后的首地址(内存映射)

参数二：文本框的位置 X轴 输出框在液晶屏上的位置

参数三：文本框的位置 Y轴

参数四：设置好的字体输出框

//把字体框输出到LCD屏幕上

**show\_font\_to\_lcd(p,200,200,bm);**

**6.销毁所有初始化的东西**

// 关闭字体

void fontUnload(font \*f);

**fontUnload(f);**

// 关闭输出框

void destroyBitmap(bitmap \*bm)

**destroyBitmap(bm);**

注意：颜色的设置要用 getColor(A,B,G,R)

**7、编译**

arm-linux-gcc zi.c -o zi -L./ -lfont -lm

**字符串处理函数**

**函数 strstr（严格匹配大小写）**

**注意：**

在使用忽略大小写操作的函数时需要在头文件前面添加一个宏的定义

#define \_GNU\_SOURCE



 示例：

char \*s = "abcd.txt";

char \*p = strstr(s, ".wps");

if(p == NULL)

    printf("文件[%s]不是WPS文件\n", s);

else

    printf("文件[%s]是WPS文件\n", s);

#define \_GNU\_SOURCE

#include <stdio.h>

#include <string.h>

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

{

    char \*p1 = "Hello Even GZ2315 Even";

    char \*p2 = "even";

    printf("p1:%p\n" , p1);

    printf("p2:%p\n" , p2);

    // char \*ptr = strstr( p1 , p2 );  // 严格匹配大小写

    char \*ptr = strcasestr( p1 , p2 );  // 匹配的过程中忽略大小写 注意在头文件前加上#define \_GNU\_SOURCE

    if (NULL == ptr)

    {

        printf("找不到目标子串..\n");

    }

    else{

        printf("找到子串，他的入口地址是：%p:%s\n" , ptr , ptr );

    }

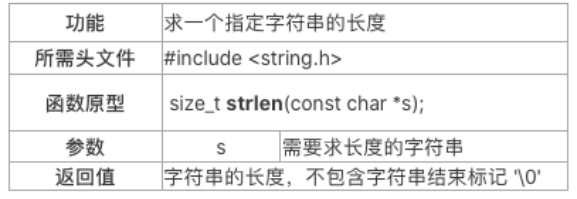
    return 0;

}

**函数strlen**

注意：

该函数是一个strxxxx类的函数，因此他的结束条件是 '\0' ， 所以在使用该函数的时候他计算的长度只到第一个出现的'\0'为止，而且这个结束符'\0'不在计算的范围内；

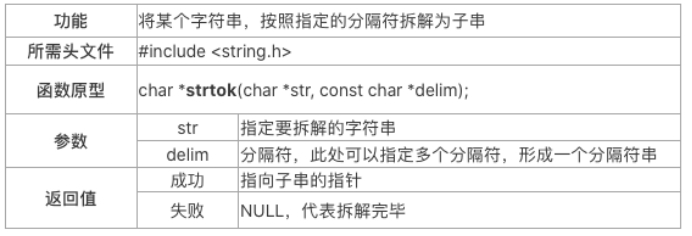


 示例：

char \*s = "www.\0yueqian.com.cn";

printf("粤嵌官网地址的长度是：%d\n", strlen(s));

**函数strtok**



 注意：

 该函数会将改变原始字符串 str（因此str必须是指向一个可读可写的入口地址），使其所包含的所有分隔符变成结束标记 ‘\0’ 。

 由于该函数需要更改字符串 str，因此 str 指向的内存必须是可写的。

 首次调用时 str 指向原始字符串，此后每次调用 str 用 NULL 代替。

 分隔符组合 delim 可以由多个字符组成，在分割的时候这多个字符会被单独匹配，匹配成功则把str中对应的字符替换为 '\0'

char buf[10]="123,456";

char buf1[3]={0};

char seqs[] = ",";  //分割的字符

    char \*tmp = strtok(buf,seqs);//123

    strcpy(buf1,tmp);

    tmp = strtok(NULL,seqs);//456 继续往下分割

    strcpy(buf1,tmp);

**函数strcpy与strncpy**



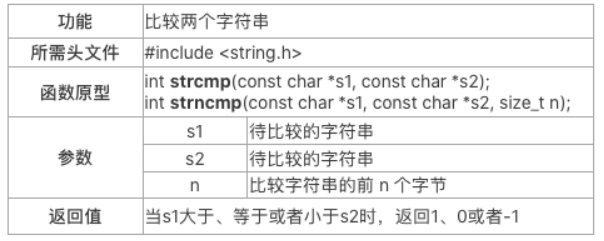
 注意：

 这两个函数的功能，都是将 src 中的字符串，复制到 dest 中。

 strcpy() 没有边界控制，因此可能会由于 src 的过长而导致内存溢出。

 strncpy() 有边界控制，最多复制 n+1 个字符（其中最后一个是 ‘\0’ ）到 dest 中。

**函数strcmp与strncmp**



 注意：

 比较字符串大小，实际上比较的是字符的 ASCII码值的大小。

 从左到右逐个比较两个字符串的每一个字符，当能“决出胜负”时立刻停止比较。

 从左到右逐个进行字符之间的减法运算，不等与0则表示不相等

 因此相等返回 0  否则返回差值

 所有的以str开头的函数都是在遇到第一个'\0'结束符的时候停止运行返回

  如果函数中有 n 则有两种结束的情况 '\0' + 到达用户指定 N字节

 示例：

printf("%d\n", strcmp("abc", "abc")); // 输出0，两个字符串相等

printf("%d\n", strcmp("abc", "aBc")); // 输出32 差值，"abc" 大于 "aBc"

printf("%d\n", strcmp("aBc", "abc")); // 输出-32差值

printf("%d\n" , strncmp(p1 , p2 , 4 )); // 比较两个字符串的前 4字节