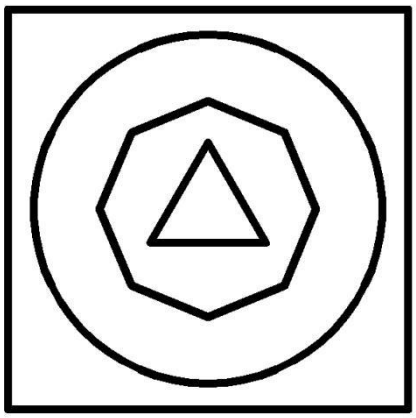
printf 不定长参数原理



本文件由皇天惊虞制作，免费流通于网络。

制作时间　**2023.12.29**

printf 不定长参数原理

xiongsiyu979 已于 2022-06-24 22:03:01 修改

一、printf()函数的参数不定长

printf()函数的功能是将字符串格式化，然后打印到屏幕上，而 printf()函数的参数是不确定的(>=1)

printf()函数的声明：int printf(const char *format, …);

使用 printf 函数格式化类似于做完型填空，它分为两部分 const char *format 是题目， ...是选项，做对题目只有一个要求：根据题目的顺序，选项的变量是题目指定的变量类型

```
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("%c say %s %d \n", 'I', "hello", 6);

    return 0;
}
```

二、printf()实现参数不定长原理

参数传递是依次存放在栈中传递的，不定参数存放在固定参数的后面

内存中不定参数，按照 4 或 8 字节对齐（32 位机器上按照 4 字节对齐，64 位机器上按照 8 字节对齐），对于指针变量，只存储指针变量所指向的地址

来写一个简易版的 my_printf()，实现上面标准 C 语言库函数 printf()的效果

```
#include <stdio.h>
#include <stdarg.h>

void my_printf(const char *format, ...)
{
    va_list arg;      //va_list 是一个宏，是指向不定长参数列表的指针，也就是“选项”列表
    va_start(arg, format);          //初始化 arg，将 format 后第一个地址赋值给 arg

    while (*format)//遍历"题目"
    {
        char ret = *format;             //一个 char 一个 char 地遍历
        if (ret == '%')
        {
            switch (*++format)        //先++再取值
            {
            case 'c':
```

```
            {
                char ch = va_arg(arg, int);      //va_arg(arg,int),获得 arg 指向参数的值，同时使 arg 指向下一个参数,int 用来指名当前参数型,当前参数虽然为 char 类型，但是却用 int，这是由于说明当前参数类型只是为了从内存取值，而 va_list 是按照 4 字节或 8 字节对齐的，指定为 char 会报错
                putchar(ch);
                break;
            }
            case 's':
            {
                char *pc = va_arg(arg, char *);
                while (*pc)
                {
                    putchar(*pc);
                    pc++;
                }
                break;
            }
            case 'd':
            {
                int in = va_arg(arg, int);
                putchar(in+'0');
                break;
            }
            default:
                break;
            }
        }

        else
        {
            putchar(*format);
        }
        format++;
    }
    va_end(arg);//va_end 在有些实现中可能会把 arg 改成无效值，这里，是把 arg 指针指向了 NULL,避免出现野指针
}
int main(int argc, char **argv)
{
    my_printf("%c say %s %d \n", 'I', "hello", 6);

    return 0;
}
```

printf 不定长参数原理

printf 的实现分析

printf 函数:

```
int printf(const char *fmt, ...){
    int i;
    char buf[256];
    va_list arg = (va_list)((char*)(&fmt) + 4);
    i = vsprintf(buf, fmt, arg);
    write(buf, i);
    return i;}
```

vsprintf 函数将所有的参数内容格式化之后存入 buf, 返回格式化数组的长度, vsprintf 函数如下:

```
int vsprintf(char *buf, const char *fmt, va_list args) {
    char* p;
    char tmp[256];
    va_list p_next_arg = args;
    for (p=buf;*fmt;fmt++) {
        if (*fmt != '%') {
            *p++ = *fmt;
            continue;
        }
        fmt++;
        switch (*fmt) {
        case 'x':
            itoa(tmp, *((int*)p_next_arg));
```

```
            strcpy(p, tmp);
            p_next_arg += 4;
            p += strlen(tmp);
            break;
        case 's':
            break;
        default:
            break;
        }
    }
    return (p - buf); }
```

随后 write 函数将参数放入寄存器，然后用 int 21h 调用 sys_call 。sys_call 将字符串中的字节从寄存器中通过总线复制到显卡的显存中，显存中存储的是字符的 ASCII 码。

字符显示驱动子程序通过 ASCII 码在字模库中找到点阵信息，并将点阵信息存储到 vram 中。

显示芯片会按照一定的刷新频率逐行读取 vram，并通过信号线向液晶显示器传输每一个点（RGB 分量）。

最后，hello 程序的输出：hello 就显示在了屏幕上。