手把手教你实现printf函数（C语言方式）

blog.csdn.net/qq_44078824/article/details/118440458

引言

在日常的嵌入式开发过程中，常常会用到格式化输出的功能。比如在LCD屏幕上，显示需要的字符，如果没有格式化输出，用起来将会是十分麻烦。本文运用变参函数的知识，提供一种实现printf的格式化输出的实现方法供大家参考。

实现思路

通过一个个读取需要打印的字符，如果遇到格式化输出的字符，则根据格式化规则，用变参函数的方式取读取到参数，然后将参数拆解输出出来。

参考工程

使用VS17编译的工程：代码打包下载

主要难点为变参函数，下面介绍变参函数。

变参函数学习

1. 定义

即：函数数目可变的函数。

变参函数原型

```
type VarArgFunc(type FixedArg1, type FixedArg2, …);
```

参数分两部分：固定参数+数目可变参数。

固定参数：至少一个。
可变参数： 大于等于0个。使用“...”表示。

2. 变参函数举例

printf（及其家族），原型：

```
int printf(const char* format, ...);
```

实际调用形式：

```
printf("string");

printf("%d", i);

printf("%s", s);

printf("number is %d, string is:%s", i, s);
```

3.实现原理

使用到的宏： 需要#include <stdarg.h>

C调用约定下可使用va_list系列变参宏实现变参函数，此处va意为variable-argument(可变参数)。

val_list:

原型：

```
type va_arg(va_list argptr, type);
```

是在C语言中解决变参问题的一组宏，用于获取不确定个数的参数。

va_start:

原型：

```
void va_start(va_list argptr,last_parm);
```

读取可变参数的过程其实就是在栈区中，使用指针,遍历栈区中的参数列表,从低地址到高地址一个一个地把参数内容读出来的过程。实现功能类似变参的初始化。

```
// vc 6.0定义
#define va_start(ap,v) ( ap = (va_list)&v + _INTSIZEOF(v) ) //ap 为val_list变量。v为最后一个固定参数地址
```

va_arg:

宏定义：type va_arg (va_list ap, type)
该宏用于变参数函数调用过程中，type是当前参数类型，调用该宏后，ap指向变参数列表中的下一个参数，返回ap指向的参数值，是一个类型为type的表达式。

va_end:

原型：

```
void va_end(va_list argptr);
```

指针va_list置为无效，结束变参的获取

典型用法如下：

```
#include <stdarg.h>

int VarArgFunc(int dwFixedArg, ...){ //以固定参数的地址为起点依次确定各变参的内存起始地址

    va_list pArgs = NULL;  //定义va_list类型的指针pArgs，用于存储参数地址

    va_start(pArgs, dwFixedArg); //初始化pArgs指针，使其指向第一个可变参数。该宏第二个参数是变参列表的前一个参数，即最后一个固定参数

    int dwVarArg = va_arg(pArgs, int); //该宏返回变参列表中的当前变参值并使pArgs指向列表中的下个变参。该宏第二个参数是要返回的当前变参类型

    //若函数有多个可变参数，则依次调用va_arg宏获取各个变参

    va_end(pArgs);  //将指针pArgs置为无效，结束变参的获取

    /* Code Block using variable arguments */

}

//可在头文件中声明函数为extern int VarArgFunc(int dwFixedArg, ...);，调用时用VarArgFunc(FixedArg, VarArg);
```

代码实现

本工程包括main.c , myPrintf.c ,myPrintf.h , send.c , send.h这几个代码文件。

头文件写了函数的声明，源文件写了函数的实现过程。

1.send.c就是底层实现打一个字符的函数，演示暂时使用printf %c 来模拟。可以将这个打印字符的函数替换成自己需要的底层函数。

```
#include <stdio.h>
#include "./../ins/send.h"

// 底层打印字符函数
void my_send_char(char chr)
{
    // 可以替换成自己的函数，比如LCD显示字符
    // LCD_Show_Char(chr);   // 注意移动光标位置
    printf("%c", chr);
}
```

send.h文件

```
#ifndef _MYPRINTF_H_
#define _MYPRINTF_H_

// 底层打印字符函数
void my_send_char(char chr);

#endif
```

2.myPrintf.c 文件简单写了一个计算n的m次幂函数，用于数据的拆分显示。其次则是Print函数。

Print函数首先对一些参数进行了定义和对va_start进行初始化，然后循环变量字符串。在循环里面，嵌套swich来来进行格式化输出。第一层switch用于匹配一些转义字符%，\r，\t，\n等转义字符。当遇到%时，则进入第二层switch用于匹配格式化输出的规则。比如：当遇到%d时，则调用va_arg读出一个整型参数，然后对该参数进行拆分打印处理。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "./../ins/myPrintf.h"
#include "./../ins/send.h"
extern void my_send_char(char chr);

// 计算m^n
unsigned long m_pow_n(unsigned long m, unsigned long n)
{
    unsigned long i = 0, ret = 1;
    if (n < 0) return 0;
    for (i = 0; i < n; i++)
    {
        ret *= m;
    }
    return ret;
}

// 返回值为打印字符的个数
// 支持%d，%o，%x，%s，%c，%f（只打印6位数字）
int Print(const char* str, ...)
{
    if (str == NULL) return -1;

        unsigned int ret_num = 0;// 返回打印字符的个数
    char* pStr = (char*)str;// 指向str
    int ArgIntVal = 0;  // 接收整型
    unsigned long ArgHexVal = 0;// 接十六进制
    char* ArgStrVal = NULL;  // 接收字符型
    double ArgFloVal = 0.0; // 接受浮点型
    unsigned long val_seg = 0;   // 数据切分
    unsigned long val_temp = 0;  // 临时保存数据
    int cnt = 0;       // 数据长度计数
    int i = 0;

    va_list pArgs; // 定义va_list类型指针，用于存储参数的地址
    va_start(pArgs, str); // 初始化pArgs
    while (*pStr != '\0')
    {
        switch (*pStr)
        {
        case ' ':
            my_send_char(*pStr); ret_num++; break;
        case '\t':
            my_send_char(*pStr); ret_num += 4; break;
        case '\r':
            my_send_char(*pStr); ret_num++; break;
        case '\n':
            my_send_char(*pStr); ret_num++; break;
        case '%':
            pStr++;
            // % 格式解析
            switch (*pStr)
            {
            case '%':
                my_send_char('%');// %%，输出%
                ret_num++;
                pStr++;
```

```
                    continue;
case 'c':
    ArgIntVal = va_arg(pArgs, int);// %c，输出char
    my_send_char((char)ArgIntVal);
    ret_num++;
    pStr++;
                    continue;
case 'd':
    // 接收整型
    ArgIntVal = va_arg(pArgs, int);
    if (ArgIntVal < 0)// 如果为负数打印，负号
    {
        ArgIntVal = -ArgIntVal;// 取相反数

        my_send_char('-');
        ret_num++;
    }
    val_seg = ArgIntVal;// 赋值给 val_seg处理数据
    // 计算ArgIntVal长度
    if (ArgIntVal)
    {
        while (val_seg) {
            cnt++;
            val_seg /= 10;
        }
    }
    else cnt = 1;// 数字0的长度为1

    ret_num += cnt;// 字符个数加上整数的长度

    // 将整数转为单个字符打印
    while (cnt)
    {
        val_seg = ArgIntVal / m_pow_n(10, cnt - 1);
        ArgIntVal %= m_pow_n(10, cnt - 1);
        my_send_char((char)val_seg + '0');
        cnt--;
    }
    pStr++;
    continue;
case 'o':
    // 接收整型
    ArgIntVal = va_arg(pArgs, int);
    if (ArgIntVal < 0)// 如果为负数打印，负号
    {
        ArgIntVal = -ArgIntVal;// 取相反数

        my_send_char('-');
        ret_num++;
    }
    val_seg = ArgIntVal;// 赋值给 val_seg处理数据
    // 计算ArgIntVal长度
    if (ArgIntVal)
    {
        while (val_seg) {
            cnt++;
            val_seg /= 8;
        }
    }
    else cnt = 1;// 数字0的长度为1
```

```
                ret_num += cnt;// 字符个数加上整数的长度

                // 将整数转为单个字符打印
                while (cnt)
                {
                    val_seg = ArgIntVal / m_pow_n(8, cnt - 1);
                    ArgIntVal %= m_pow_n(8, cnt - 1);
                    my_send_char((char)val_seg + '0');
                    cnt--;
                }
                pStr++;
                                continue;
            case 'x':
                // 接收16进制
                ArgHexVal = va_arg(pArgs, unsigned long);
                val_seg = ArgHexVal;
                // 计算ArgIntVal长度
                if (ArgHexVal)
                {
                    while (val_seg) {
                        cnt++;
                        val_seg /= 16;
                    }
                }
                else cnt = 1;// 数字0的长度为1

                ret_num += cnt;// 字符个数加上整数的长度
                // 将整数转为单个字符打印
                while (cnt)
                {
                    val_seg = ArgHexVal / m_pow_n(16, cnt - 1);
                    ArgHexVal %= m_pow_n(16, cnt - 1);
                    if (val_seg <= 9)
                        my_send_char((char)val_seg + '0');
                    else
                    {
                                                //my_send_char((char)val_seg - 10 + 'a');
//小写字母
                        my_send_char((char)val_seg - 10 + 'A');
                    }
                    cnt--;
                }
                pStr++;
                                continue;
            case 'b':
                // 接收整型
                ArgIntVal = va_arg(pArgs, int);
                val_seg = ArgIntVal;
                // 计算ArgIntVal长度
                if (ArgIntVal)
                {
                    while (val_seg) {
                        cnt++;
                        val_seg /= 2;
                    }
                }
                else cnt = 1;// 数字0的长度为1

                ret_num += cnt;// 字符个数加上整数的长度
```

```
    // 将整数转为单个字符打印
    while (cnt)
    {
        val_seg = ArgIntVal / m_pow_n(2, cnt - 1);
        ArgIntVal %= m_pow_n(2, cnt - 1);
        my_send_char((char)val_seg + '0');
        cnt--;
    }
    pStr++;
                    continue;
case 's':
    // 接收字符
    ArgStrVal = va_arg(pArgs, char*);
    ret_num += (unsigned int)strlen(ArgStrVal);
    while (*ArgStrVal)
    {
        my_send_char(*ArgStrVal);
        ArgStrVal++;
    }

    pStr++;
                    continue;

case 'f':
    // 接收浮点型 保留6为小数，不采取四舍五入
    ArgFloVal = va_arg(pArgs, double);
    val_seg = (unsigned long)ArgFloVal;// 取整数部分
    val_temp = val_seg;      // 临时保存整数部分数据
    ArgFloVal = ArgFloVal - val_seg;// 得出余下的小数部分
    // 计算整数部分长度
    if (val_seg)
    {
        while (val_seg) {
            cnt++;
            val_seg /= 10;
        }
    }
    else cnt = 1;// 数字0的长度为1
    ret_num += cnt;// 字符个数加上整数的长度
    // 将整数转为单个字符打印
    while (cnt)
    {
        val_seg = val_temp / m_pow_n(10, cnt - 1);
        val_temp %= m_pow_n(10, cnt - 1);
        my_send_char((char)val_seg + '0');
        cnt--;
    }
    // 打印小数点
    my_send_char('.');
    ret_num++;
    // 开始输出小数部分
    ArgFloVal *= 1000000;
    // printf("\r\n %f\r\n", ArgFloVal);
    cnt = 6;
    val_temp = (int)ArgFloVal;// 取整数部分
    while (cnt)
    {
        val_seg = val_temp / m_pow_n(10, cnt - 1);
        val_temp %= m_pow_n(10, cnt - 1);
        my_send_char((char)val_seg + '0');
```

```
                    cnt--;
                }
                ret_num += 6;
                pStr++;
                                continue;
            default:// % 匹配错误，暂输出空格
                                my_send_char(' '); ret_num++;
                                continue;
            }


        default:
            my_send_char(*pStr); ret_num++;
            break;
        }
        pStr++;
    }
    va_end(pArgs);// 结束取参数

    return ret_num;
}
```

myPrintf.h则写了Print的声明。

```
#ifndef _MYPRINTF_H_
#define _MYPRINTF_H_


// 返回值为打印字符的个数
// 支持%d，%x，%s，%c，%f（只打印6位数字）
int Print(const char* str, ...);

#endif
```

3.main写了一些简要测的测试，测试全部正常。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "./ins/myPrintf.h"

int main()
{

        Print(NULL, 123);
        Print("% \r\n");
        Print("%%d\r\n", 123);
        Print("%d(int0)\r\n", 0);
        Print("%d(int100)\r\n", 100);
        Print("int(-123) =%d\r\n", -123);
        Print("%d(int 123)\r\n", 123);
        Print("%d(int 0x1234=4660)\r\n", 0x1234);
        Print("%d(int 0x12345678=305419896)\r\n", 0x12345678);

        Print("oct %o(int 0x1234=011064)\r\n", 0x1234);
        Print("oct %o(int 0x12345678=02215053170)\r\n", 0x12345678);
        Print("hex(0x0)=%x\r\n", 0x0);
        Print("hex(0x100)=%x\r\n", 0x100);
        Print("hex(0x1234)=%x\r\n", 0x1234);
        Print("hex(0x7fffffff)=0x%x\r\n", 0x7fffffff);
        Print("hex(0xffffffff)=0x%x\r\n", 0xffffffff);
        Print("bin(0xff)=%b\r\n", 0xff);
        Print("%b(bin 0xff)\r\n", 0xff);
        Print("str=%s\r\n", "hello");
        Print("%s(str)\r\n", "hello");
        Print("ch =%c\r\n", 'a');
        Print("%c(ch)\r\n", 'a');
        Print("float(3.141592) = %f\r\n", 3.141592);
        printf("=====pinrf:float= %f\r\n\r\n", 3.141592);

        Print("float(123456789.123456789) = %f\r\n", 123456789.123456789);
        printf("==========pinrf:float = %f\r\n\r\n", 123456789.123456789);

        Print("float(123456.123456789) = %f\r\n", 123456.123456789);
        printf("==========pinrf:float = %f\r\n\r\n", 123456.123456789);

        Print("%d%s\r\n\r\n", 123, "abc");
        return 0;
}
```

程序运行结果

注：`%f`保留6位置小数，不采取四舍五入

```
%d
0(int0)
100(int100)
int(-123) =-123
123(int 123)
4660(int 0x1234=4660)
305419896(int 0x12345678=305419896)
oct 11064(int 0x1234=011064)
oct 2215053170(int 0x12345678=02215053170)
hex(0x0)=0
hex(0x100)=100
hex(0x1234)=1234
hex(0x7fffffff)=0x7FFFFFFF
hex(0xffffffff)=0xFFFFFFFF
bin(0xff)=11111111
11111111(bin 0xff)
str=hello
hello(str)
ch =a
a(ch)
float(3.141592) = 3.141592
=====pinrf:float= 3.141592

float(123456789.123456789) = 123456789.123456
==========pinrf:float = 123456789.123457

float(123456.123456789) = 123456.123456
==========pinrf:float = 123456.123457

123abc
```

总结:

暂未做数据安全性检查，规范使用可以正常输出。
使用VS2017进行编译测试，结果符合预期。

本文到此结束，感谢大家的阅读。

欢迎大家评论交流。