

嵌入式C语言之- printf函数

讲师：叶大鹏

助力你成为优秀的电子工程师！



整体介绍

- printf是在stdio.h中声明的一个函数，因此使用前必须加入#include <stdio.h>，使用它可以向标准输出设备（比如屏幕、串口）输出数据；
- 在嵌入式开发中，用于打印调试信息。

用法

1. printf(字符串)

```
printf("Hello, World!");
```

2. printf(字符串, 格式符参数, 格式符参数...)

```
printf("My age is %d.\n", 30); //使用常量作为参数
```

```
uint8_t age = 35;
```

```
printf("My age is %d.\n", age); //使用变量作为参数
```

- 格式符%d表示以有符号的十进制形式输出一个整型，格式符参数中的30和age会代替%d的位置；
- 第1行代码中的\n是个转义字符，表示换行，所以输出了第一句"My age is 30"后会先换行，再输出"My age is 35"。

用法

```
My age is 30.  
My age is 35.
```

- 如果去掉第1行中的\n，将会是这样的效果：

```
My age is 30.My age is 35.
```

- 左边字符串中格式符的个数 必须跟 右边格式符参数的个数一样。

常用格式符及其含义

格式符	描述
d	以带符号的十进制形式输出整数（正数不输出+）
u	以不带符号的十进制形式输出整数
x / X	以十六进制形式输出整数 ， x 对应的是 abcdef, X 对应的是 ABCDEF（没有前导 0x 或者 0X）
f / lf	以小数形式输出单、双精度数，默认输出为6位小数（lf 在 C99 开始加入标准，意思和 f 相同）
c	输出一个ASCII码字符
s	输出字符串
p	以16进制形式输出地址

格式符还可以添加一些精细的格式控制

- 输出宽度

1. 我们先看看默认的整型输出

```
printf("The price is %d.", 14);
```

输出结果(注意, 后面是有个点的):

```
The price is 14.
```

2. 如果把%d换成%4d:

```
printf("The price is %4d.", 14);
```

输出结果:

```
The price is 14.
```

我们会发现"is"跟"14"的距离被拉开了;

格式符还可以添加一些精细的格式控制

%4d的意思是输出宽度为4，而"14"的宽度为2，因此多出2个宽度，多出的宽度就会在左边用空格填补，因此你会看到"14"左边多了2个空格；

3. 如果实际数值宽度比较大，比如用%4d输出宽度为6的"142434"，那就会按照实际数值宽度6来输出：

```
printf("The price is %4d.",142434);
```

输出结果为：

```
The price is 142434.
```

格式符还可以添加一些精细的格式控制

- 格式修饰符 0

如果把%4d换成%04d:

```
printf("The price is %04d.", 14);
```

输出结果为:

```
The price is 0014.
```

%04d的意思是输出宽度为4，而"14"的宽度为2，因此多出2个宽度，多出的宽度就会在左边用0填补，因此你会看到"14"左边多了2个0;

格式符还可以添加一些精细的格式控制

- 浮点数的小数位

1. 我们先看下默认的浮点数输出

```
printf("My height is %f.\n",179.95f);
```

输出结果:

```
My height is 179.949997.
```

默认是输出6位小数。

2. 如果只想输出2位小数，把%f换成%.2f即可

```
printf("My height is %.2f\n",179.95f);
```

输出结果:

```
My height is 179.95
```

格式符还可以添加一些精细的格式控制

3. 当然，可以同时设置输出宽度和小数位数

```
printf("My height is %8.1f",179.95f);
```

输出结果：

```
| My height is      179.9
```

输出宽度为8，保留1位小数。

格式符还可以添加一些精细的格式控制

3. 当然，可以同时设置输出宽度和小数位数

```
printf("My height is %8.1f",179.95f);
```

输出结果：

```
| My height is      179.9
```

输出宽度为8，保留1位小数。

格式符还可以添加一些精细的格式控制

- 格式修饰符 #

1. 我们先看看默认的整型十六进制输出

```
printf("The hex is %X.", 140);
```

输出结果:

```
The hex is 8C.
```

2. 如果把%X换成 %#X:

```
printf("The hex is %#X.", 140);
```

输出结果:

```
The hex is 0X8C.
```

%#X的意思是，当使用十六进制转换说明符x或X时，在输出数据前面加上前导符0x或0X。

转义序列

- 大家有没有想过这样一个问题：怎样将%、\和""这三个符号通过printf输出呢？

要输出%只需在前面再加上一个%，要输出\只需在前面再加上一个\，要输出双引号也只需在前面加上一个\即可：

```
printf("%%d\n");
```

```
printf("\\ \n");
```

```
printf("\" \"\n");
```

输出结果：

```
%d
```

```
\
```

```
" "
```

转义序列

- \ 在这里有着特殊含义，叫做 转义字符，当编译器遇到字符串中的转义字符时，会将转义字符及其下一个字符组成一个转义序列

转义序列	描述
\'	单引号
\"	双引号
\?	问号
\\	反斜杠
\b	退格
\n	换行
\r	回车
\t	水平制表符
\v	垂直制表符

转义序列 \n

- \n用于将光标移到下一行的起始位置。

```
printf("This is pn xuetang\n");  
printf("We learn C together\n");
```

输出结果：

```
This is pn xuetang  
We learn C together
```

转义序列 \t

- \t水平制表符，相当于按了Tab键，有些终端环境对Tab宽度默认为4，有些则为8。

```
printf("\t%d\n", 111);  
printf("1234567\t%d\n", 111);  
printf("12345678\t%d\n", 111);
```

输出结果：

111 → 111前面空了8格，表示一个\t,如果是2个\t,则要空出16格

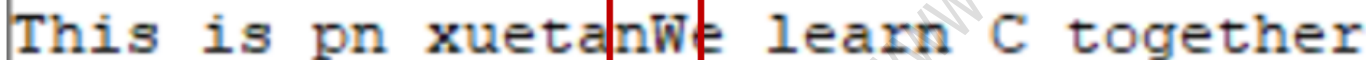
1234567 111 → 111前面空了1格，因为前面已近有了7位，1+7=8表示一个\t

12345678 111 → 111前面空了8格，表示一个\t,因为前面已经有了8位，
\t重新开始

转义序列 \b

```
printf("This is pn xuetang\b");  
printf("We learn C together");
```

输出结果:



This is pn xuetanWe learn C together

\b退1格，输出位置回退到xuetan的后面，所以再输出字符时，g被替换掉了。

THANK YOU!