<https://blog.csdn.net/nopoppy/article/details/52589745>

这几天总遇到sprintf，下面自己来小小地总结下~

srpintf()函数的功能非常强大：效率比一些字符串操作函数要高；而且更具灵活性；可以将想要的结果输出到指定的字符串中，也可作为缓冲区，而printf只能输出到命令行上~

头文件：stdio.h

函数功能：格式化字符串，将格式化的数据写入字符串中。

函数原型：int sprintf(char \*buffer, const char \*format, [argument]...)

参数：

（1）buffer：是char类型的指针，指向写入的字符串指针；

（2）format：格式化字符串，即在程序中想要的格式；

（3）argument：可选参数，可以为任意类型的数据；

函数返回值：buffer指向的字符串的长度；

用处：

（1）格式化数字字符串：在这点上sprintf和printf的用法一样，只是打印到的位置不同而已，前者打印给buffer字符串，后者打印给标准输出，所以sprintf也可以用来将整型转化为字符串，比itoa效率高且如此地简便~比如：sprintf(buffer, "%d", 123456);执行后buffer即指向字符串“123456”~

（2）连接字符：

下面通过两个例子来说明这个问题：

（a）连接以'\0'结束的字符串：

**[cpp]** [view plain](http://blog.csdn.net/zenail501129/article/details/22991485) [copy](http://blog.csdn.net/zenail501129/article/details/22991485)

[在CODE上查看代码片](https://code.csdn.net/snippets/276786)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** buffer[10];
5. **char** \*a = "1234";
6. **char** \*b = "5678";
7. sprintf(buffer, "%s%s", a, b);
8. printf("%s\n", buffer);
9. **return** 0;
10. }

**[cpp]** [view plain](https://blog.csdn.net/nopoppy/article/details/52589745) [copy](https://blog.csdn.net/nopoppy/article/details/52589745)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** buffer[10];
5. **char** \*a = "1234";
6. **char** \*b = "5678";
7. sprintf(buffer, "%s%s", a, b);
8. printf("%s\n", buffer);
9. **return** 0;
10. }

运行结果：

https://img-blog.csdn.net/20140405183500343?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemVuYWlsNTAxMTI5/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

（b）连接结尾没有'\0'的字符数组或字符串缓冲区：

**[cpp]** [view plain](http://blog.csdn.net/zenail501129/article/details/22991485) [copy](http://blog.csdn.net/zenail501129/article/details/22991485)

[在CODE上查看代码片](https://code.csdn.net/snippets/276786)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** a[] = {'1', '2', '3', '4'};
5. **char** b[] = {'5', '6', '7', '8'};
6. **char** buffer[10];
7. sprintf(buffer, "%.4s%.4s", a, b);
8. printf("%s\n", buffer);
9. **return** 0;
10. }

**[cpp]** [view plain](https://blog.csdn.net/nopoppy/article/details/52589745) [copy](https://blog.csdn.net/nopoppy/article/details/52589745)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** a[] = {'1', '2', '3', '4'};
5. **char** b[] = {'5', '6', '7', '8'};
6. **char** buffer[10];
7. sprintf(buffer, "%.4s%.4s", a, b);
8. printf("%s\n", buffer);
9. **return** 0;
10. }

运行结果：

https://img-blog.csdn.net/20140405183500343?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemVuYWlsNTAxMTI5/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

如果第二种情况的程序中，sprintf的format部分为"%4s%4s"或"%s%s"，运行结果均为：

https://img-blog.csdn.net/20140405183920968?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemVuYWlsNTAxMTI5/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast很莫名其妙吧~

（c）如果我们想动态获取要处理的字符缓冲区长度，则将上面sprintf改为：sprintf(buffer, "%.\*s%.\*s", sizeof(a), a, sizeof(b), b);即可~

（3）利用sprintf中的返回值：在”AOV网络和拓扑（二）——实现“篇中，有这样一行程序：pos += sprintf(output+pos, "%d ", j+1); 它的作用是什么呢？

因为sprintf函数的返回值为output+pos所指向字符串的长度，所以对于pos来说，相当于执行了一次pos+=sizeof(output+pos)，如果这条语句放在一个循环里，则第二次执行sprintf时output+pos随即指向了当前缓冲区的末尾（注意不是output的末尾！否则会读取非法内存！），这样就可以生成一个具有一定规则的字符串了~写个例子：

**[cpp]** [view plain](http://blog.csdn.net/zenail501129/article/details/22991485) [copy](http://blog.csdn.net/zenail501129/article/details/22991485)

[在CODE上查看代码片](https://code.csdn.net/snippets/276786)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** buf[100];
5. **int** pos = 0;
6. **for**(**int** j = 0; j < 10; j++)
7. pos += sprintf(buf+pos, "%d-", j);
8. buf[pos-1] = '\n';//将最后一个字符'-'转换为'\n'
9. printf(buf);
10. **return** 0;
11. }

**[cpp]** [view plain](https://blog.csdn.net/nopoppy/article/details/52589745) [copy](https://blog.csdn.net/nopoppy/article/details/52589745)

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** buf[100];
5. **int** pos = 0;
6. **for**(**int** j = 0; j < 10; j++)
7. pos += sprintf(buf+pos, "%d-", j);
8. buf[pos-1] = '\n';//将最后一个字符'-'转换为'\n'
9. printf(buf);
10. **return** 0;
11. }

运行结果：

https://img-blog.csdn.net/20140405192208343pos每次增加2，因为每次buf字符串后面都要加上两个字符j和‘-’。