[两种将字符串转换成浮点数的方法](http://blog.csdn.net/loomman/article/details/3373298)

**两种将字符串转换成浮点数的方法**

**方法一：**

1. char  szString[] = "-2876.99812376443";
2. double db1;
3. db1 = atof(szString);
4. printf("atof result:/n");
5. printf("%f  %.12f  %.2f  %e  %E/n", db1, db1, db1, db1, db1);
6. printf("%.1e  %.1E  %.18e  %.18E/n", db1, db1, db1, db1);

**方法二：**

1. char  szString2[] = "-2876.99812376443";
2. double db2;
3. sscanf(szString2, "%lf", &db2);
4. printf("/nsscanf result:/n");
5. printf("%f  %.12f  %.2f  %e  %E/n", db2, db2, db2, db2, db2);
6. printf("%.1e  %.1E  %.18e  %.18E/n", db2, db2, db2, db2);

这两种方法可以得到一模一样的输出！输出如下：

atof result:  
-2876.998124  -2876.998123764430  -2877.00  -2.876998e+003  -2.876998E+003  
-2.9e+003  -2.9E+003  -2.876998123764430100e+003  -2.876998123764430100E+003

sscanf result:  
-2876.998124  -2876.998123764430  -2877.00  -2.876998e+003  -2.876998E+003  
-2.9e+003  -2.9E+003  -2.876998123764430100e+003  -2.876998123764430100E+003

很多人对sscanf家族的函数不太了解，我想把sscanf家族的函数具体用法写出来，希望大家可以共同进步，有什么不对的地方欢迎提出来。先瞄下该家族的一些函数原型：

// 从键盘输入数据到变量

1. int scanf(char \*format，...);

// 从字符串输入数据到变量，如下相同

1. int sscanf(**const** char \*buffer, **const** char \*format, ... );
2. int \_sscanf\_l(**const** char \*buffer, **const** char \*format, locale\_t locale, ...  );
3. int swscanf(**const** wchar\_t \*buffer,**const** wchar\_t \*format, ...  );
4. int \_swscanf\_l(**const** wchar\_t \*buffer,**const** wchar\_t \*format,locale\_t locale, ...  );

下面我们只看看标准形式的sscanf函数：

1. int sscanf(**const** char \*buffer, **const** char \*format, ... );

先说说关于它的返回值的问题，库函数几乎都是有返回值的，有些人可能很奇怪，怎么很少人用过sscanf的返回值呢？sscanf会返回成功接收到的变量数量的值。比如sscanf（"3.14159","%f",&pi);返回值是1。

测试如下程序：

1. #include <stdio.h>
2. int main ()
3. {
4. int a;
5. printf ("%d",scanf("%d/n",&a));
6. **return** 0;
7. }

如果你开始就输入回车，程序会继续等待你输入，因为在输入数字的时候，scanf会跳过空白字符。the c programming language上说，scanf实际上是用getchar()接受由数字组成的字符串，再转换成数字。如果我输入ctrl-z(unix上是ctrl-d)则会返回-1(随编译器而定）.这实际上就是常量EOF的值，也就是所谓的返回eof。如果我键入的不是数字返回值就是0。但是如果我输入浮点数，又会怎么样呢？我举的例子中同样会返回1，但是缓冲区会留下垃圾，如果是scanf("%d%d",&a，&b);则会出错。这是可以使用一个库函数fflush(stdin)来清除缓冲。不过貌似雨中飞燕大姐说这个用法是非标准的。K&R，只是说行为没有定义，但我们可以使用while((c=getchar())!='/n'&&c!=EOF);同样可以清除后面的垃圾scanf的格式匹配还是比较简单，一定要记住的就是普通变量一定要加上&，否则编译器无法检测错误，但运行肯定会段错误。

1. ┏━━━━┯━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
2. ┃ 代  码 │             意                          义             ┃
3. ┠────┼────────────────────────────┨
4. ┃   %a   │读浮点值(仅适用于 C99)                 ┃
5. ┃   %A   │读浮点值(仅适用于 C99)                ┃
6. ┃   %c   │读单字符                                              ┃
7. ┃   %d   │读十进制整数                                     ┃
8. ┃   %i   │读十进制、八进制、十六进制整数 ┃
9. ┃   %e   │读浮点数                                                ┃
10. ┃   %E   │读浮点数                                                ┃
11. ┃   %f   │读浮点数                                                ┃
12. ┃   %F   │读浮点数(仅适用于 C99)                    ┃
13. ┃   %g   │读浮点数                                                ┃
14. ┃   %G   │读浮点数                                                ┃
15. ┃   %o   │读八进制数                                              ┃
16. ┃   %s   │读字符串                                                ┃
17. ┃   %x   │读十六进制数                                            ┃
18. ┃   %X   │读十六进制数                                            ┃
19. ┃   %p   │读指针值                                                ┃
20. ┃   %n   │至此已读入值的等价字符数               ┃
21. ┃   %u   │读无符号十进制整数                                      ┃
22. ┃  %[ ]  │扫描字符集合                                            ┃
23. ┃   %%   │读 % 符号(百分号)                                       ┃
24. ┗━━━━┷━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛

前面都很简单，%p,%n很少用到，跳过。要输入%必须要在前面再加一个%，重点来谈谈%s和%[]。%s是读入一个数组，他与gets的区别就在于%s会以任何的空字符结束，而gets是回车结束。同样%s前可以加数字，表示只读多少个。ANSI C 标准向 scanf() 增加了一种新特性，称为扫描集(scanset)。 扫描集定义一个字符集合，可由 scanf() 读入其中允许的字符并赋给对应字符数组。 扫描集合由一对方括号中的一串字符定义，左方括号前必须缀以百分号。 例如，以下的扫描集使 scanf() 读入字符 A、B 和 C：

1. %[ABC]

使用扫描集时，scanf() 连续吃进集合中的字符并放入对应的字符数组，直到发现不在集合中的字符为止(即扫描集仅读匹配的字符)。返回时，数组中放置以 null 结尾、由读入字符组成的字符串。对于许多实现来说，用连字符可以说明一个范围。 例如，以下扫描集使 scanf() 接受字母 A 到 Z：

1. %[A-Z]

重要的是要注意扫描集是区分大小写的。因此，希望扫描大、小写字符时，应该分别说明大、小写字母。对于%[]还可以用^+任意字符（包括eof）来结束字符串的输入。比如%[^EOF]就是直到有EOF输入，字符串才中止。但一定要记住就是c语言是缓冲输入，即使你%[^a]，再你输入回车之前输入多少的a都是不可能结束的。%s的输入会跳过空白字符，但是%c则不会。这也就是

1. scanf("%d", &h);
2. scanf("%c", &c);

如果这写的话，变量c放的一定是回车。如果想实现这种输入，可以在两个语句之间加入一个getchar(),他可以吃掉这个回车，也可用scanf("%d %c",&h,&c);来做，再输入数字后加一个空格。就可以了但千万别用scanf("%d/n", &h); K&R说的十分清楚，任何非格式化的字符都需要完全匹配。意味着，只有输入数字后面再加/n才是合法的。还有就是\*加在任何项的前面表示该项不符值。

再看一例关于如何用sscanf来分析用逗号作为分解符的字符串，这在数据库查询结果操作中很常见，代码如下：

1. /\* The following sample illustrates the use of brackets and the
2. caret (^) with sscanf().
3. Compile options needed: none
4. \*/
5. #include <math.h>
6. #include <stdio.h>
7. #include <stdlib.h>
8. char \*tokenstring = "first,25.5,second,15";
9. int result, i;
10. double fp;
11. char o[10], f[10], s[10], t[10];
12. **void** main()
13. {
14. result = sscanf(tokenstring, "%[^','],%[^','],%[^','],%s", o, s, t, f);
15. fp = atof(s);
16. i  = atoi(f);
17. printf("%s/n %lf/n %s/n %d/n", o, fp, t, i);
18. }