C语言scanf的高级用法，原来scanf还有这么多新技能

< 上一节 下一节 >

在前面几节中，我们演示了如何使用 scanf() 来读取各种各样的数据，汇总了 scanf() 可以使用的格式控制符，然后还讲解了缓冲区，从根本上消除了 scanf() 的那些奇怪行为，至此，很多初学者就认为自己已经完全掌握了 scanf()。

其实，这只是 scanf() 的基本用法，每个C语言程序员都应该掌握，如果你想让自己的输入更加炫酷、更加个性化、更加安全，那么还需要学习 scanf() 的高级用法，这才是大神和菜鸟的分水岭。

好了，言归正传，我们分三个方面讲解 scanf() 的高级用法。

1) 指定读取长度

还记得在 printf() 中可以指定最小输出宽度吗？就是在格式控制符的中间加上一个数字，例如，`%10d` 表示输出的整数至少占用 10 个字符的位置：

- 如果整数的宽度不足 10，那么在左边以空格补齐；
- 如果整数的宽度超过了 10，那么以整数本身的宽度来输出，10 不再起作用。

其实，scanf() 也有类似的用法，也可以在格式控制符的中间加一个数字，用来表示读取数据的最大长度，例如：

`%2d` 表示最多读取两位整数；

`%10s` 表示读取的字符串的最大长度为 10，或者说，最多读取 10 个字符。

请看下面的例子：

```
01.  #include <stdio.h>
02.
03.  int main(){
04.      int n;
05.      float f;
06.      char str[23];
```

```
07.
08.     scanf("%2d", &n);
09.     scanf("%*[^\n]"); scanf("%*c");  //清空缓冲区
10.     scanf("%5f", &f);
11.     scanf("%*[^\n]"); scanf("%*c");  //清空缓冲区
12.     scanf("%22s", str);
13.     printf("n=%d, f=%g, str=%s\n", n, f, str);
14.
15.     return 0;
16. }
```

输入示例 ①：

```
20↙
100.5↙
http://c.biancheng.net↙
n=20, f=100.5, str=http://c.biancheng.net
```

输入示例 ②：

```
8920↙
10.2579↙
http://data.biancheng.net↙
n=89, f=10.25, str=http://data.biancheng.
```

这段代码使用了多个 scanf() 函数连续读取数据，为了避免受到缓冲区中遗留数据的影响，每次读取结束我们都使用 `scanf("%*[^\n]"); scanf("%*c");` 来清空缓冲区。

限制读取数据的长度在实际开发中非常有用，最典型的一个例子就是读取字符串：我们为字符串分配的内存是有限的，用户输入的字符串过长就存放不了了，就会冲刷掉其它的数据，从而导致程序出错甚至崩溃；如果被黑客发现了这个漏洞，就可以构造栈溢出攻击，改变程序的执行流程，甚至执行自己的恶意代码，这对服务器来说简直是灭顶之灾。

在用 gets() 函数读取字符串的时候，有一些编译器会提示不安全，建议替换为 gets_s() 函数，就是因为 gets() 不能控制读取到的字符串的长度，风险极高。

就目前学到的知识而言，虽然 scanf() 可以控制字符串的长度，但是字符串中却不能包含空白符，这是硬伤，所以 scanf() 暂时还无法替代 gets()。不过大家也不要着急，稍后我还会补充 scanf() 的高级用法，届时 scanf() 就可以完全替代 gets()，并且比 gets() 更加智能。

2) 匹配特定的字符

%s 控制符会匹配除空白符以外的所有字符，它有两个缺点：

- %s 不能读取特定的字符，比如只想读取小写字母，或者十进制数字等，%s 就无能为力；
- %s 读取到的字符串中不能包含空白符，有些情况会比较尴尬，例如，无法将多个单词存放到一个字符串中，因为单词之间就是以空格为分隔的，%s 遇到空格就读取结束了。

要想解决以上问题，可以使用 scanf() 的另外一种字符匹配方式，就是 `%[xxx]`，`[ ]` 包围起来的是需要读取的字符集合。例如，`%[abcd]` 表示只读取字符 `abcd`，遇到其它的字符就读取结束；注意，这里并不强调字符的顺序，只要字符在 abcd 范围内都可以匹配成功，所以你可以输入 abcd、dcba、ccdc、bdcca 等。

请看下面的代码：

```
01.  #include <stdio.h>
02.  int main(){
03.      char str[30];
04.      scanf("%[abcd]", str);
05.      printf("%s\n", str);
06.
07.      return 0;
08.  }
```

输入示例 ①：

```
abcdefgh↙
abcd
```

输入示例 ②：

```
baccbaxyz↙
baccba
```

使用连接符

为了简化字符集合的写法，scanf() 支持使用连字符 `-` 来表示一个范围内的字符，例如 %[a-z]、%[0-9] 等。

连字符左边的字符对应一个 ASCII 码，连字符右边的字符也对应一个 ASCII 码，位于这两个 ASCII 码范围以内的字符就是要读取的字符。注意，连字符左边的 ASCII 码要小于右边的，如果反过来，那么它的行为是未定义的。

常用的连字符举例：

- `%[a-z]` 表示读取 abc...xyz 范围内的字符，也即小写字母；
- `%[A-Z]` 表示读取 ABC...XYZ 范围内的字符，也即大写字母；
- `%[0-9]` 表示读取 012...789 范围内的字符，也即十进制数字。

你也可以将它们合并起来，例如：

- %[a-zA-Z] 表示读取大写字母和小写字母，也即所有英文字母；
- %[a-z-A-Z0-9] 表示读取所有的英文字母和十进制数字；
- %[0-9a-f] 表示读取十六进制数字。

请看下面的演示：

```
#include <stdio.h>
int main(){
    char str[30];
    scanf("%[a-zA-Z]", str);  //只读取字母
    printf("%s\n", str);
    return 0;
}
```

输入示例：

```
abcXYZ123↙
abcXYZ
```

不匹配某些字符

假如现在有一种需求，就是读取换行符以外的所有字符，或者读取 0~9 以外的所有字符，该怎么实现呢？总不能把剩下的字符都罗列出来吧，一是麻烦，二是不现实。

C语言的开发者们早就考虑到这个问题了，scanf() 允许我们在 %[ ] 中直接指定某些不能匹配的字符，具体方法就是在不匹配的字符前面加上 ^，例如：

- %[^\n] 表示匹配除换行符以外的所有字符，遇到换行符就停止读取；
- %[^0-9] 表示匹配除十进制数字以外的所有字符，遇到十进制数字就停止读取。

请看下面的例子：

```
#include <stdio.h>
int main(){
    char str1[30], str2[30];
    scanf("%[^0-9]", str1);
    scanf("%*[^\n]"); scanf("%*c");  //清空缓冲区
    scanf("%[^\n]", str2);
    printf("str1=%s \nstr2=%s\n", str1, str2);

    return 0;
}
```

输入示例：

```
abcXYZ@#87edf↙
c c++ java python go javascript↙
str1=abcXYZ@#
str2=c c++ java python go javascript
```

请注意第 6 行代码，它的作用是读取一行字符串，和 gets() 的功能一模一样。你看，scanf() 也能读取带空格的字符串呀，谁说 scanf() 不能完全取代 gets()，这明显是错误的说法。

另外，scanf() 还可以指定字符串的最大长度，指定字符串中不能包含哪些字符，这是 gets() 不具备的功能。

例如，读取一行不能包含十进制数字的字符串，并且长度不能超过 30：

```
01.  #include <stdio.h>
02.  int main(){
03.      char str[31];
04.      scanf("%30[^0-9\n]", str);
05.      printf("str=%s\n", str);
06.
07.      return 0;
08.  }
```

输入示例 ①：

```
http://c.biancheng.net http://biancheng.net↙
str=http://c.biancheng.net http://
```

输入示例 ②：

```
I have been programming for 8 years.↙
str=I have been programming for
```

总之，scanf() 不仅可以完全替代 gets()，并且比 gets() 的功能更加强大。

3) 丢弃读取到的字符

在前面的代码中，每个格式控制符都要对应一个变量，把读取到的数据放入对应的变量中。其实你也可以不这样做，scanf() 允许把读取到的数据直接丢弃，不往变量中存放，具体方法就是在 % 后面加一个 `*` ，例如：

- `%*d` 表示读取一个整数并丢弃；
- `%*[a-z]` 表示读取小写字母并丢弃；

- %*[^\n] 表示将换行符以外的字符全部丢弃。

请看下面的代码演示：

```
#include <stdio.h>
int main(){
    int n;
    char str[30];
    scanf("%*d %d", &n);
    scanf("%*[a-z]");
    scanf("%[^\n]", str);
    printf("n=%d, str=%s\n", n, str);

    return 0;
}
```

输入示例：

```
100 999abcxyzABCXYZ↙
n=999, str=ABCXYZ
```

对结果的分析：整数 100 被第一个 scanf() 中的 %*d 读取后丢弃了，整数 999 被第 %d 读取到，并赋值给 n。此时缓冲区中剩下 abcxyzABCXYZ，第二个 scanf() 将 abcxyz 读取并丢弃，剩下的 ABCXYZ 被最后一个 scanf() 读取到并赋值给 str。

大家有没有意识到，将读取到的字符直接丢弃，这就是在清空输入缓冲区呀，虽然有点蹩脚，但是行之有效。在《清空（刷新）缓冲区，从根本上消除那些奇怪的行为》一节中，我们已经给出了使用 scanf() 清空缓冲区的方案，就是：

```
scanf("%*[^\n]"); scanf("%*c");
```

下面我们就来解释一下。

首先需要明白的是，等到需要清空缓冲区的时候，缓冲区中的最后一个字符一定是换行符 \n，因为输入缓冲区是行缓冲模式，用户按下回车键会产生换行符，结束本次输入，然后输入函数开始读取。

scanf("%*[^\n]"); 将换行符前面的所有字符清空，scanf("%*c"); 将最后剩下的换行符清空。

有些网友将这两条语句合并起来，写作：

```
scanf("%*[^\n]%*c");
```

这是错误的。合并以后的语句不能清空单个换行符，因为该语句要求换行符前边至少要有一个其它的字符，单个换行符会导致匹配失败。

总结

scanf() 控制字符串的完整写法为：

```
%{*} {width} type
```

其中，{ } 表示可有可无。各个部分的具体含义是：

- type 表示读取什么类型的数据，例如 %d、%s、%[a-z]、%[^\n] 等；type 必须有。
- width 表示最大读取宽度，可有可无。
- * 表示丢弃读取到的数据，可有可无。

< 上一节 下一节 >