

scanf, gets, getchar 的区别

三者都是从标准输入流 `stdio` (标准输入设备, 一般指向键盘) 中读取内容。

1. `scanf` 输入字符串的时候不会接收 `Space` 空格, 回车 `Enter`, `Tab` 键, 则认为输入结束。

2. `gets` 能接收空格键, 回车键, `Tab` 键, 回车则认为输入结束

3. `getchar` 只能接受一个字符, 遇到回车结束输入, 可接受回车键。常用来吸收回车符。

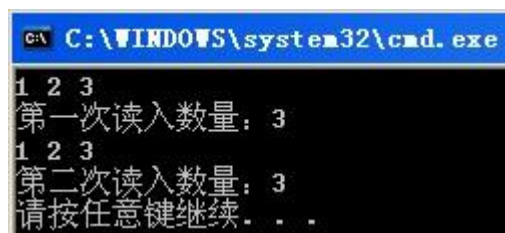
输入操作原理

程序的输入都建有一个缓冲区, 即输入缓冲区。一次输入过程是这样的, 当一次键盘输入结束时会将输入的数据存入输入缓冲区, 而 `cin` 函数, `scanf` 函数直接从输入缓冲区中取数据。正因为 `cin` 函数, `scanf` 函数是直接取数据的, 所以有时候当缓冲区中有残留数据时, `cin` 函数, `scanf` 函数会直接取得这些残留数据而不会请求键盘输入, 这就是有时自己写的代码中为什么会出现输入语句失效的原因! 因为 `scanf()` 和 `getchar()` 函数是从输入流缓冲区中读取值的, 而并非从键盘(也就是终端)缓冲区读取。而读取时遇到回车(`\n`)而结束的, 这个 `\n` 会一起读入输入流缓冲区的, 所以第一次接受输入时取走字符后会留下字符 `\n`, 这样第二次的读入函数直接从缓冲区中把 `\n` 取走了, 显然读取成功了, 所以不会从终端读取! 这就是为什么这个程序只执行了一次输入操作就结束的原因!

例子:

```
#include<stdio.h>
int main(void){
    int a=0,b=0,c=0,ret=0;
    ret=scanf("%d%d%d",&a,&b,&c);
    printf("第一次读入数量: %d\n",ret);
    ret=scanf("%d%d%d",&a,&b,&c);
    printf("第二次读入数量: %d\n",ret);
    return 0;
}
```

我们定义了 `a`, `b`, `c` 三个变量来接受输入的内容, 定义了变量 `ret` 来接收 `scanf` 函数的返回值。正确输入的话:



但是当输入内容与格式换字符串不匹配时, 结果会令人大跌眼镜 (仔细分析会对 `scanf` 函数和 `stdin` 流有更深入的了解):



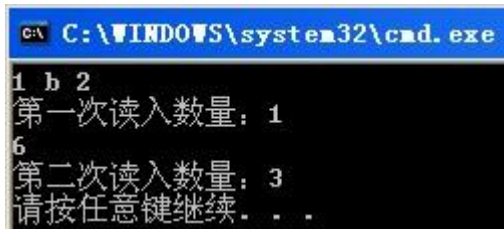
执行到第一个 `scanf` 时, 当输入字符 '`b`' 的时候与 `ret=scanf("%d%d%d",&a,&b,&c);` 中的格式化字符串不匹配, `stdin` 流被阻塞, `scanf` 函数不在读取后面的部分, 直接将 1 返回, 表示只将 `stdin` 流中的 1 读入到了变量 `a` 中。

执行到第二个 scanf 时，字符 'b' 还是与格式化字符串不匹配，stdin 流仍然被阻塞，所以没有提示输入，scanf 函数将 0 返回。

将代码作如下修改，可以有力的证明上述结论。

```
#include<stdio.h>
int main(void){
int a=0,b=0,c=0,ret=0;
ret=scanf("%d%d%d",&a,&b,&c);
printf("第一次读入数量：%d\n",ret);
ret=scanf("%c%d%d",&a,&b,&c);
printf("第二次读入数量：%d\n",ret);
return 0;
}
```

当把第二个 scanf 函数内的格式化字符串改为 "%c%d%d" 时，运行结果如下：



执行到第一个 scanf 函数时，由于输入 'b' 的原因 scanf 函数直接返回 1，stdin 流阻塞。

执行到第二个 scanf 函数时，字符 'b' 与格式化字符串 "%c%d%d" 中的 %c 匹配，stdin 流终于疏通，在输入 6，则将变量 a, b, c 分别赋值为 98('b' 的 ASCII 码)、2、6，scanf 函数返回 3。

不同的输入函数是否会舍弃最后的回车符问题

向缓冲区读取字符时

1 scanf () 用 %c, %s 空格键，Tab 键，回车键结束一次输入，不会舍弃最后的回车符或空格或 Tab (即还在缓冲区中)，可使用 getchar 来吸收 scanf() 执行之后的换行符。

2 getchar () 以回车键结束，也不会舍弃回车符 (即存入缓存区)

3 gets () 以换行符结束，但之后会舍弃换行符并以 '\0' 代替 (意思是 '\n' 不会被代入到字符数组中，也不会将换行符存入到缓存区) 也就是说：gets() 函数读取到 \n (我们输入的回车) 于是停止读取，但是它不会把 \n 包含到字符串里面去。然而，和它配合使用的 puts 函数，却在输出字符串的时候自动换行。