


# scanf

的 秘密

丁培毅

- 既然是「秘密」 釋迦牟尼佛曰
- 不能說 不可說 說不得
- 可是  蠻神秘的喔。。。不知道可惜
- 算了, 資訊系同學比較威 不用說  
有一天註定要知道的話自然就會發現了  
你比較好命的話, 也許 30 年都不需要知道
- 實際上是。。。廢話有點多  
其實我真的懶得做投影片  
太複雜了不好說  
說了滿屋子瞌睡蟲就來了

- 實習裡有類似下面這樣格式的資料檔案

```
99 number\n
6 first\n
7 second\n
...
```

```
99
6
7
...
```

```
99 number of items\n
6 first item\n
7 second item\n
...
```

scanf("%\*[\n]");

如果用 C 的 stdio 來處理

```
int i, ndata, data[100];
scanf("%d", &ndata);
for (i=0; i<ndata; i++) {
scanf("%d %s", &data[i]);
}
for (i=0; i<ndata; i++) {
scanf("%d", &data[i]);
scanf("%s", garbage);
}
```

```
int i, ndata, data[100];
scanf("%d", &ndata);
scanf("%*s");
for (i=0; i<ndata; i++) {
scanf("%d", &data[i]);
scanf("%*s");
}
```

煩不煩啊!! 還要用一個 garbage 陣列

scanf 不爽地說。。。

又沒人強迫你這樣寫  
我也不喜歡作白功啊  
幫你讀出來, 你卻當垃圾丟掉

所以 %\*5s, %\*d, %\*c, %\*f, %\*x。。。不會太難接受吧?!

- 作業一裡面需要處理如下的資料

```
12345678901+987654321012345678901231\n
```

```
用 scanf 來處理 int n1, n2; char op; scanf("%d%c%d", &n1, &op, &n2);
如果, 不是變態那麼多位數的話, 根本完美
scanf("%d%c%d", &n1, &op, &n2);
還好我也不是肉腳啊!!!
scanf("%s%c%s", n1, &op, n2);
printf("n1=[%s]\nop=[%c]\nn2=[%s]\n", n1, op, n2);
檢查, 不過是個作業而已, 滾!
```

```
什麼! 怎麼這樣? n1=[12345678901+]
op=[ ]
n2=[987654321012345678901231]
```

scanf 就不能聽話一點嗎?

資料之間就不能都有空格嗎?

```
scanf("%d+c%d", &n1, &op, &n2);
12345678901+987654321012345678901231\n
```

你這個爛仔亂終棄的

哪裡來的  這麼點渣滓就放棄我囉!!! 洗洗睡囉!!

扣扣! 助教! 該上場了!!

# %s 格式命令的延伸

- **scanf** 在執行 %s 命令時, 是由輸入串流中
  1. 跳過所有 white space (「空格」、「\t」、「\n」) 字元,
  2. 讀取一連串不是 white space 的字元,
  3. 直到又遇見 white space 字元為止,如果串流中的資料是 「 1234abcd!@#\$ 」, **scanf("%s", buf)** 會把 1234abcd!@#\$ 讀到 buf 字元陣列裡面

## 驚奇之旅 %[xyz] 或是 %[xyz] 格式命令

- 如何擴充 %s 的功能呢? 可以指定 接受哪些字元 嗎?  
例如: 輸入串流中有一串字元「 abcbadab », 希望讀到 abcba 就好了, 不要接受其它任何字元

`scanf("%[abc]", buf)` 把可接受讀入 buf 的字元表列出來 字元陣列

5

- 如果想要接受的字元很多怎麼辦? 鍵盤可以打出來的 ASCII 字元有 96 個耶! 好問題
  - 可以用 0-9 表示連續的 ASCII 字元 0123456789, A-Z 表示 ABCD...Z, A-E 表示 ABCDE, ...
  - 也可以負面表列 – 把不接受的列出來 –
    - `%[^a-z]` 表示所有不是小寫英文字母的字元
    - `%[^\t\n]` 表示所有不是空格、tab、換列的字元噢, 這不是 %s 嗎?? 不要懷疑, 加上「跳過空格」就是了
- 在 scanf 的格式命令裡, 「空格」是一個命令 `scanf("%d %d", ...)` 跳過輸入串流中所有的 white space 字元 (`\t, \n` 沒有這個效果) 也可以寫成 `scanf("%d\t%d", ...)` 比較複雜一些, 沒有讀到 white space 的話 scanf 會中斷, 要自己用一個 scanf(), 還是「空格」好 :)
- 在 scanf 的格式命令裡, 「字元 x」是一個命令 `scanf("%d,%d", ...)` 要求 scanf 檢查輸入串流的字元是 x, 正確才繼續處理 也可以寫成 `scanf("%d,%x", ...)` 這裡 x 不包括 空格, \t, \n

6

# scanf 回報的訊息

- 你命令 **scanf** 由輸入串流讀取資料, 還運用前面奇奇怪怪的格式命令加上各種要求與限制, 使用者不聽話亂輸入怎麼辦??!

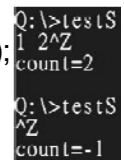
- **scanf** 當掉給你看嗎??  
好喔, M\$ 你慢慢找



- 不會啦, 除非你叫它讀取資料, 卻不給它足夠的存放空間, 否則 **scanf** 會很聽話, 不管你給它的 格式命令 和 使用者的輸入 多麼不一致, **scanf** 會根據指示進行比對與過濾!!
- 如果 **scanf** 發現輸入串流中的資料無法滿足格式命令, 它會立即中斷並返回呼叫端, 而且會藉由回傳值跟你回報處理的狀況  
要檢查! 要檢查! 要檢查!

7

- **scanf** 函式的 回傳值 代表 有幾個資料成功地讀進來  
例如: `count = scanf("%d%lf%[a-z]", &num1, &num2, str);`  
執行完以後有五種可能的回傳值: -1, 0, 1, 2, 3
- EOF
- ✓ -1 代表沒讀到資料之前, 串流已經結束(輸入Ctrl-Z)
  - ✓ 0 代表處理第一個格式命令 %d 時根本沒有在輸入串流看到十進位的資料, 所以沒有讀到任何資料 (後面兩個命令 %lf 與 %[a-z] 還沒有機會執行到), 例如輸入 abc 12 test
  - ✓ 1 代表第一個 %d 命令正確轉換資料到整數變數 num1 中, 但是%lf 命令並沒有在輸入串流看到十進位的浮點數, 所以 num2 和 str 都沒有讀到資料, 例如輸入 123 a hello
  - ✓ 2 代表 %d%lf 的執行是正確的, num1 和 num2 變數裡有正確的資料, 但是執行 %[a-z] 命令時沒有看到格式正確的資料, 例如輸入 123 45.67 89 根據回傳值, 你的程式需
  - ✓ 3 代表三個格式命令都正確執行完畢 要決定該如何執行下去
  - ✓ 讀到部份資料卻沒有完全讀完, 如何知道串流已經結束? 要使用 feof() 或是 ferror() 或是再執行一次 scanf() 囉



8

➤ 有了前面的基礎, 該研究一下怎麼處理作業的資料了

```
12345678901+ 987654321012341\n
```

- 因為數值範圍太大了, 超過 int 或是 long long 的範圍所以不能用 %d 或是 %ld, 需要用字元陣列來儲存 ⇒ **%[0-9]**
- 但是 %s 會把任意的字元都讀進字元陣列裡, 例如: !@#\$ 或是 12345678901+, 假設沒有 !@#\$ 無良的輸入, 多讀進來一個 + 號還是很討厭啊!! ⇒ **%[0-9]**, 這樣子 + 號就乖乖地留在串流裡面了
- 可是 %s 會跳過引導的空格, %[0-9] 不會 ⇒ 在 % 前面加 **空格**
- 接下來是空格和 +-\* ⇒ **空格%[-+\*/]**
- 測試以後發現另一個問題, - 號沒有辦法輸入, 稍微想一下就會想到 **%[0-9]** 這個命令的減號, 如果它代表一個連續範圍, 那就不代表減號 ⇒ **空格%[-+\*/]** 或是 **空格%[-+\*/]**
- 還有一個問題是 +-\* 符號的個數 ⇒ **空格%1[-+\*/]**

# 你遜喔!! scanf 不安全啦!!

**error C4996: 'scanf': This function or variable may be unsafe. Consider using scanf\_s instead. To disable deprecation, use \_CRT\_SECURE\_NO\_WARNINGS**

```
#pragma warning(disable:4996) 有沒有一種愉悅的感覺
或是
#define _CRT_SECURE_NO_WARNINGS 無視它
或是
#define _CRT_SECURE_NO_DEPRECATED
```



- 不要這樣啦!! 是哪裡不安全呢???????
- 聽人家說不安全就不安全了喔?

```
#include <stdio.h>
int main() {
    char str[5];
    int count = scanf("%s", str);
    printf("count=%d str=[%s]\n", count, str);
    return 0;
}
```

正常情況下

```
1234
count=1 str=[1234]
```

記憶體錯誤



使用者不聽話

```
12345678
count=1 str=[12345678]
```

程式沒有語法和語意的錯誤, 但是使用者不依照規範輸入, 麻煩就大了

那怎麼辦?

```
就說不要啊!! scanf("%4s", str);
```

程式保護自己一下, 使用者不依照規範輸入都沒有辦法讓程式當掉

可是 scanf 真的很強耶, 好可惜喲, 吃飯噲噲, 就別吐了吧! 5678 還留在串流裡

```
count=1 str=[1234]
```

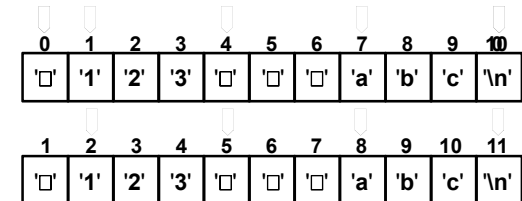
時一定要記得喔!!

## %n - 可以知道 scanf 讀到串流第幾個字元的密技

```
#include <stdio.h>
int main() {
    int num, pos1, pos2;
    char buf[10];
    scanf("%d\n%s\n", &num, &pos1, buf, &pos2);
    scanf("%d\n%s\n", &num, &pos1, buf, &pos2);
    printf("pos1=%d pos2=%d\n", pos1, pos2);
    scanf("%d\n%s\n", &num, &pos1, buf, &pos2);
    printf("pos1=%d pos2=%d\n", pos1, pos2);
    return 0;
}
```

```
Q: \>testScanfPos
123 abc
pos1=4 pos2=10
123 abc
pos1=8 pos2=11
```

```
pos1 8
pos2 10
```



## %n 平常可以用在哪裡?

- %n 用在 **sscanf** 可以很方便地移動字串緩衝區的指標
- 假設你有一個字元陣列 `char buf[] = "abc 123";`
- `char str[10];`  
`sscanf(buf, "%s", str);` // 由 buf 中讀 abc 到 str
- 接下來怎麼用 **sscanf** 讀剩下的 123 呢?
- 需要知道前面讀 abc 的時候到底讀了幾個字元
- `int pos, x;`  
`sscanf(buf, "%s%n", &str, &pos);`  
`sscanf(&buf[pos], "%d", &x);` // x 就會是 123 了
- 試試看, 你可以用 **sscanf** 取代 **strtok** 來處理字串

13

## 其它

```
q:\>testScanfGets
123 56
count=1 buf=[123 56
] len=6 pos=7

q:\>testScanfGets
01234 6789012
count=1 buf=[01234 678] len=9 pos=-1
```

- `char buf[10];`  
`fgets(buf,10,stream);`  
`int count, len, pos=-1;`  
`count=fscanf(stream,"%9[^\n]%n%*1[\n]%n",buf,&len,&pos);`  
`if (pos>=0) buf[len] = '\n', buf[len+1] = '\0';`
- `if (getchar()=='\n')` 無論如何都由串流中讀出一個字元  
...  
`if (1==scanf("%*1[\n]"))` 是換列字元才讀否則留在串流裡  
....
- 請注意 `int a, b; char c;`  
`scanf("%d%d%c", &a, &b, &c);` 和  
`scanf("%d", &a); scanf("%d", &b); scanf("%c", &c);`
  - 在輸入串流內資料正確時 (例如 1 2 a) 是一樣的
  - 在輸入串流內資料不正確時 (例如 1 a 2) 表現是不一樣的

14