

在 `strcpy.c` 和 `fixed-strcpy.c` 這兩段程式中，最大的不同就在於計算字串長度的 `while` 迴圈。

兩者的程式碼如下。

而為了比較兩者的差異，都在 `while` 迴圈後面加上 `printf`，藉此確認兩者計算出的 `len` 值。

在原程式碼中，該字串為 `cs23!`。(字串長度為：5)

`strcpy.c`：

```
while(src[len++]);  
printf("%d\n", len);
```

`len = 6`

`fixed-strcpy.c`：

```
while(src[++len]);  
printf("%d\n", len);
```

`len = 5`

這兩者程式碼的不同就在於 `++` 是放在 `len` 的前面還是後面。

所以兩者的定義有什麼差異呢？

**`len++`：**先進行取值再自增。

**`++len`：**先自增再取值。

之前的 `while` 迴圈：(先判斷，再動作)

```
while(src[len]/*代表len!=NULL時，len才會++*/) {  
    len++;  
}
```

現在的 `while` 迴圈：(判斷和 `++` 同時)

```
while(src[len++]);  strcpy.c  
while(src[++len]);  fixed-strcpy.c
```

strcpy.c :

```
while(src[len++]);
```

len = 0 有字符(-c)，len++，此時 len = 1

len = 1 有字符(-s)，len++，此時 len = 2

len = 2 有字符(-2)，len++，此時 len = 3

len = 3 有字符(-3)，len++，此時 len = 4

len = 4 有字符(-!)，len++，此時 len = 5

len = 5 無字符('\0')，跳出迴圈，但 len 仍會++，此時 len=6，比真實答案多 1

以下藉由印出 while 迴圈中的 len 值，和最終的 len 值來驗證：

```
while(src[len++]) {  
    printf("%d\n", len);  
}  
printf("\n%d\n", len);
```

```
1  
2  
3  
4  
5  
6
```

fixed-strcpy.c :

```
while(src[++len]);
```

直接從 len =1 開始(因為++在後面，代表先+，再開始運算)

len=1，有字符(-s)，此時 len=1

len=2，有字符(-2)，此時 len=2

len=3，有字符(-3)，此時 len=3

len=4，有字符(-!)，此時 len=4

len=5，無字符('\0')，跳出迴圈，此時 len=5 為正確答案。

藉由 while 迴圈進行最後的驗證：

```
while(src[++len]) {  
    printf("%d\n", len);  
}  
printf("\n%d\n", len);
```

```
1  
2  
3  
4  
5
```