

下圖螢光筆繪製處為相異點，我們將探討 `len++` 與 `++len` 在此處形成之差異

```
int main()
{
    char src[] = "cs23!";
    char dst[]="Hello hello";
    char *curdst;
    int len=0;

    printf("src address %p and first char %c \n", (void *)&src, src[0]);
    printf("dst address %p and first char %c \n", (void *)&dst, dst[0]);

    // compute where NULL character is '\0' ASCII 0
    // while(src[len++]); THE BUG. What was the problem?
    while(src[++len]); // THE FIX: How does this fix it? **001**

    // print out the char arrays and various addresses.
    printf("src array %s and last element %d\n", src, atoi(&src[len]));
    printf("dst array %s and last element %c\n", dst, dst[len]);

    // do the copy
    curdst= my_strcpy(dst, src);

    // check to see if the NULL char is copied too.
    printf("dst array %s and last element %d\n", dst, atoi(&dst[len]));

    return 0;
}

char *my_strcpy(char *s1, const char *s2) {
    register char *d = s1;

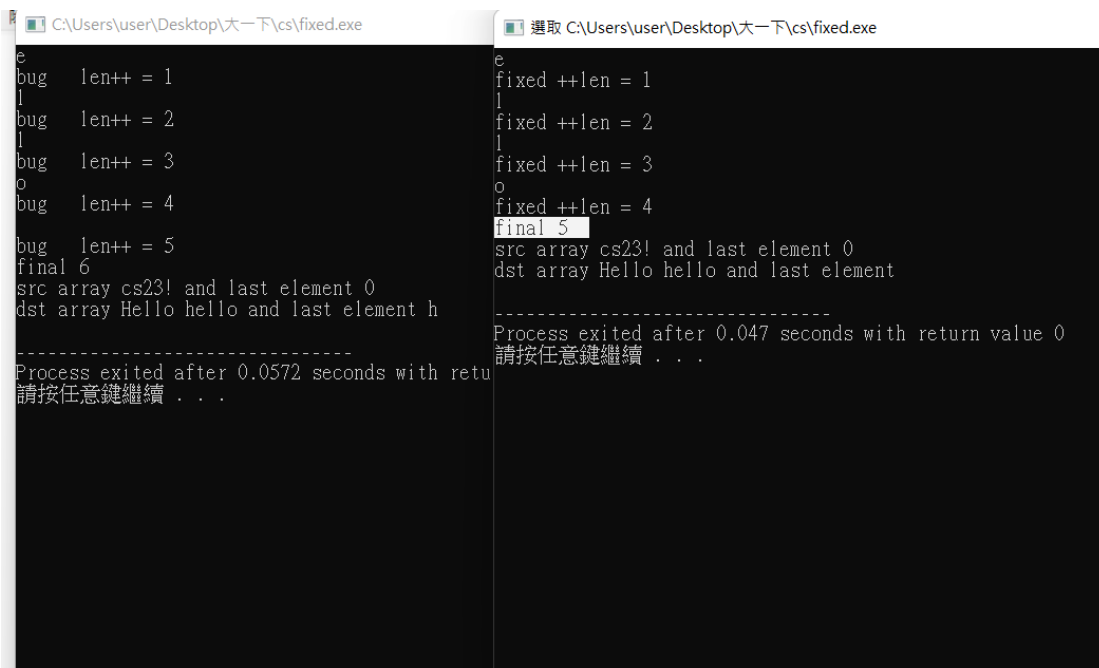
    // print the pointer variables address and their contents, and first char
    printf("s2 address %p, its contents is a pointer %p to first char %c \n", (void *)&s2, (void *)s2, *s2);
    printf("s1 address %p, its contents is a pointer %p to first char %c \n", (void *)&s1, (void *)s1, *s1);

    while ((*d++ = *s2++));
    return(s1);
}
```

為了更明顯看出差異，我們將程式碼改寫為：

```
27 // compute where NULL character is '\0' ASCII 0
28 int flag = 0; // 0為bug, 1為true
29
30 if(flag==0){
31     while(src[len++]) {
32         printf("%c\n",dst[len]);
33
34         // THE BUG. What was the problem?5
35         printf("bug   len++ = %d\n",len);
36     }
37 }
38 if(flag==1) {
39     while(src[++len]) {
40         printf("%c\n",dst[len]);
41         // THE FIX: How does this fix it? **001**
42         printf("fixed ++len = %d\n",len);
43     }
44 }
45 printf("final %d\n",len);
46 printf("src array %s and last element %d\n", src, atoi(&src[len]));
47 printf("dst array %s and last element %c\n", dst, dst[len]);
48
```

以下為 bug 與 fixed 所形成之差異



The image shows two side-by-side terminal windows comparing the execution of a C program. The left window, titled 'C:\Users\user\Desktop\大一下\cs\fixed.exe', shows the 'bug' version. It has a loop that increments 'len' from 1 to 5, then prints 'final 6'. The right window, titled 'C:\Users\user\Desktop\大一下\cs\fixed.exe', shows the 'fixed' version. It has a loop that increments 'len' from 1 to 5, then prints 'final 5'. Both windows show the same source and destination arrays and exit after 0.0572 and 0.047 seconds respectively.

```
e
bug  len++ = 1
1
bug  len++ = 2
1
bug  len++ = 3
0
bug  len++ = 4
0
bug  len++ = 5
final 6
src array cs23! and last element 0
dst array Hello hello and last element h
-----
Process exited after 0.0572 seconds with return value 0
請按任意鍵繼續 . . .

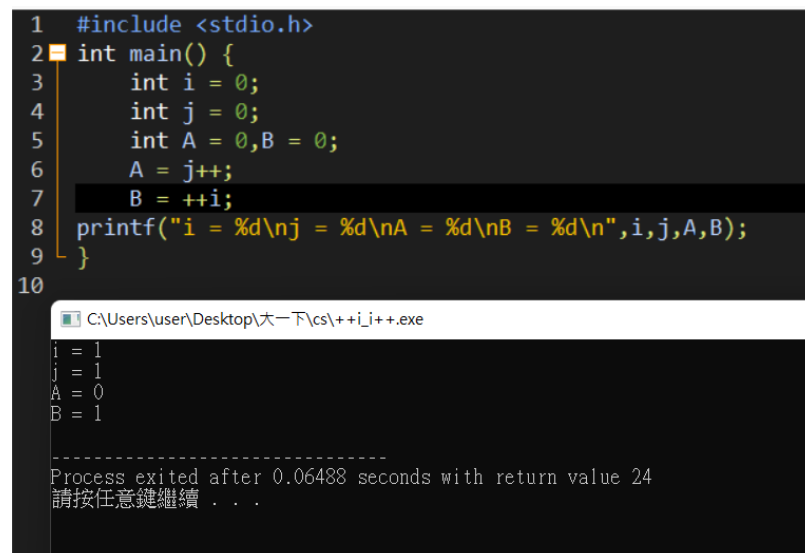
e
fixed ++len = 1
1
fixed ++len = 2
1
fixed ++len = 3
1
fixed ++len = 4
0
fixed ++len = 5
final 5
src array cs23! and last element 0
dst array Hello hello and last element
-----
Process exited after 0.047 seconds with return value 0
請按任意鍵繼續 . . .
```

Bug 版本算出來的 len 為 5；fixed 版本算出來的 len 為 6

Fixed : ++len 加到 5 了才發現第五個是 0，跳出迴圈，所以最後的 len 落在 5

Bug： len++數了五個以後，因為第六個是 0，跳出迴圈，再做++(先做再+)，所以最後的 len 落在 6

以下為 j++與++i 的差異



The image shows a C code snippet and its output. The code defines variables i, j, A, and B, and prints their values. The output shows i = 1, j = 1, A = 0, and B = 1.

```
1 #include <stdio.h>
2 int main() {
3     int i = 0;
4     int j = 0;
5     int A = 0, B = 0;
6     A = j++;
7     B = ++i;
8     printf("i = %d\nj = %d\nA = %d\nB = %d\n", i, j, A, B);
9 }
10
```

```
i = 1
j = 1
A = 0
B = 1
-----
Process exited after 0.06488 seconds with return value 24
請按任意鍵繼續 . . .
```

第 6 行為把 j 丟給 A，再對 j 做++；

第 7 行為先對 i 做++，再丟給 B。