其實 strcpy.c 與 fixed-strcpy.c 只差在一行 code， 而且這行 code 還不是在 function my_strcpy 裡面，是 main 裡面的，雖然從跑完程式的 output 來看，似乎無法發現差異，見下圖

```
src address 0x7fff4419dc26 and first char c
dst address 0x7fff4419dc2c and first char H
src array cs23! and last element 0
dst array Hello hello and last element h
s2 address 0x7fff4419dbe0, its contents is a pointer 0x7fff4419dc26 to first char c
s1 address 0x7fff4419dbe8, its contents is a pointer 0x7fff4419dc2c to first char H
dst array cs23! and last element 0
```

strcpy.c

```
src address 0x7ffe7f8b7576 and first char c
dst address 0x7ffe7f8b757c and first char H
src array cs23! and last element 0
dst array Hello hello and last element
s2 address 0x7ffe7f8b7530, its contents is a pointer 0x7ffe7f8b7576 to first char c
s1 address 0x7ffe7f8b7538, its contents is a pointer 0x7ffe7f8b757c to first char H
dst array cs23! and last element 0
```

fixed_strcpy.c

只有畫線部分的差異，但其實這代表著原本的 len 計算中，舊的會多計算一次，使得複製時會多複製一個 char，但是結果無法分辨，因為還是會複製到結尾的 NULL(\0)，所以不容易看出來，因字串會在此終結。

而來看程式碼

```
// compute where NULL character is '\0' ASCII 0

while(src[len++]);

// print out the char arrays and various addresses.
```

strcpy.c

```
// 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.
```

fixed-strcpy.c

唯一的差別就是在 len++ 變 ++len

這使得在 count 字串長度時，原本的 len++會在讀取完時使 len 加一，而當讀取到 NULL，雖然 while 終止，但 len 仍加一，所以改成先加一，即++len 即可改善此情況。

這樣阻止了 len 多算一位，複製時也會少複製到多的那一位，output 時就會發現在 dst 中取到正確的空白字元而非 h 字元，所以便改善了此 bug。