string.h

strcpy vs. strncpy

string.h

strcpy vs. strncpy

以為是一個安全玩具，怎麼用都安全!?

string.h

strcpy vs. strncpy

以為是一個安全玩具，怎麼用都安全!?

真相是：會的人「小心使用」才安全

string.h

strcpy vs. strncpy

以為是一個安全玩具，怎麼用都安全!?

真相是：會的人「小心使用」才安全

參數越多，會出錯的地方越多

string.h

strcpy vs. strncpy

以為是一個安全玩具，怎麼用都安全!?

真相是：會的人「小心使用」才安全

參數越多，會出錯的地方越多

一定要那麼辛苦嗎? 有不用動腦可以完成的嗎?

string.h

strcpy vs. strncpy

以為是一個安全玩具，怎麼用都安全!?

真相是：會的人「小心使用」才安全

參數越多，會出錯的地方越多

一定要那麼辛苦嗎? 有不用動腦可以完成的嗎?

往好的地方想，你會了以後競爭的人會比較少

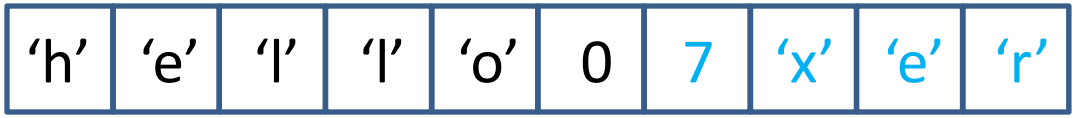
strcpy

- char ***strcpy**(char *destination, const char *source);

strcpy

- char ***strcpy**(char *destination, const char *source);

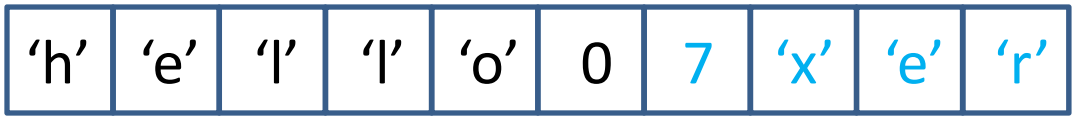
char src[10] = “hello”;



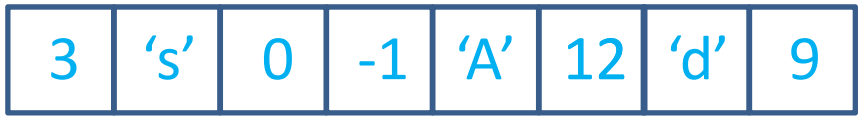
strcpy

- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



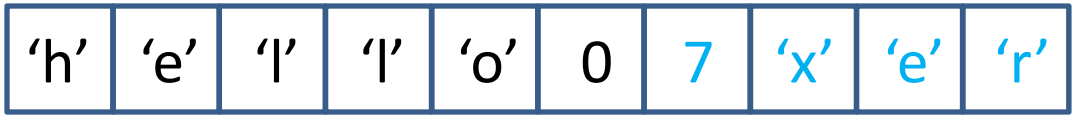
char dest[8];



strcpy

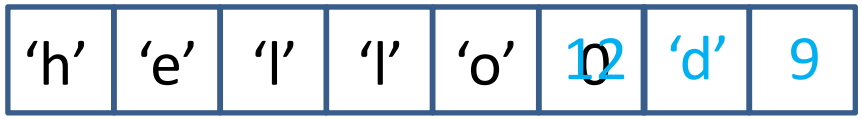
- char ***strcpy***(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[8];



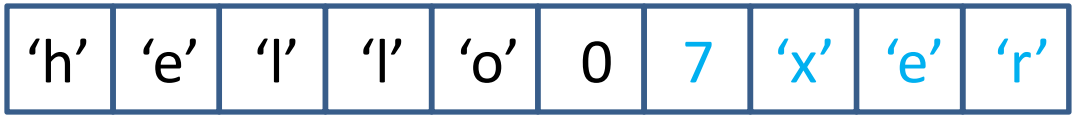
strcpy

- char ***strcpy**(char *destination, const char *source);

strcpy

- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



strcpy

- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

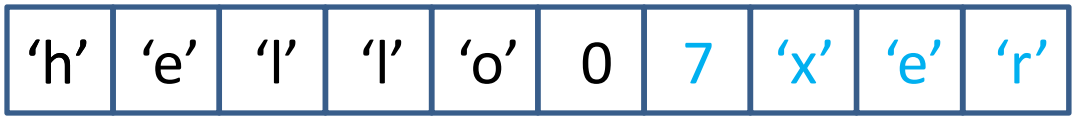
char dest[4];

3	‘s’	0	-1		

strcpy

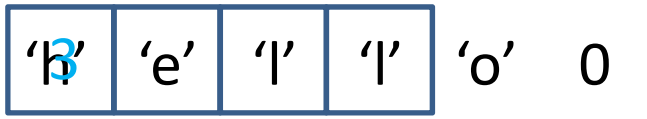
- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[4];

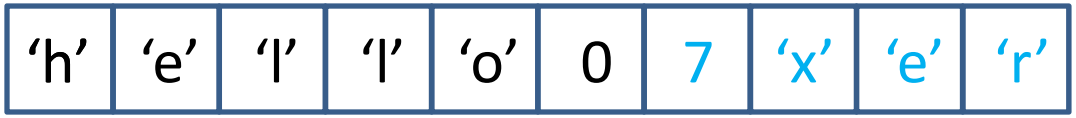


如果這裡原先有重要東西，就被覆蓋掉了

strcpy

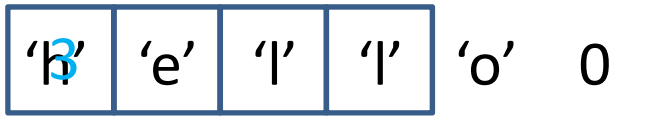
- char ***strcpy***(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[4];



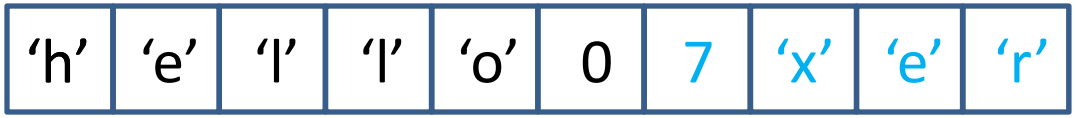
如果這裡原先有重要東西，就被覆蓋掉了

是什麼東西會被覆蓋掉?
不一定耶，不同程式、不同環境都不一樣，編譯器剛剛好安排資料存放在那個地方

strcpy

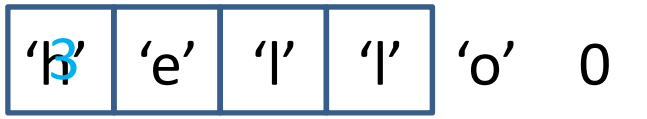
- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[4];



如果這裡原先有重要東西，就被覆蓋掉了

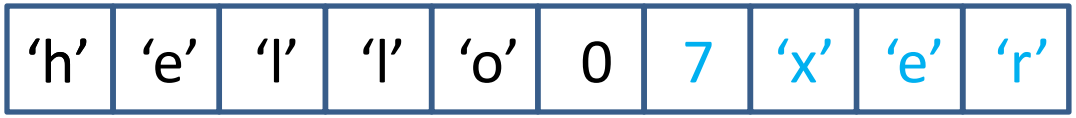
是什麼東西會被覆蓋掉?
不一定耶，不同程式、不同環境都不一樣，編譯器剛剛好安排資料存放在那個地方

程式表現無法預期

strcpy

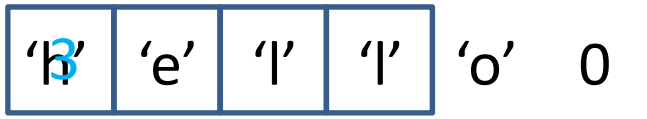
- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[4];



如果這裡原先有重要東西，就被覆蓋掉了

是什麼東西會被覆蓋掉?
不一定耶，不同程式、不同環境都不一樣，編譯器剛剛好安排資料存放在那個地方

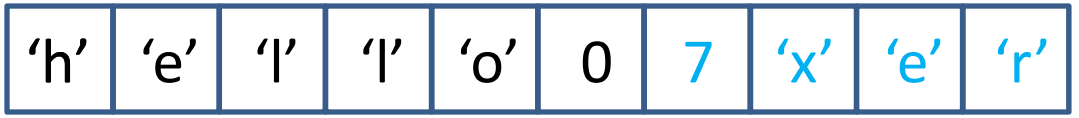
程式表現無法預期

```
i = 0;
while (source[i]!=0) {
   destination[i] = source[i];
   i = i + 1;
}
destination[i] = 0;
```

strcpy

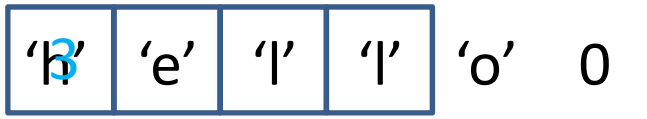
- char ***strcpy**(char *destination, const char *source);

char src[10] = “hello”;



strcpy(dest, src);

char dest[4];



如果這裡原先有重要東西，就被覆蓋掉了

是什麼東西會被覆蓋掉?
不一定耶，不同程式、不同環境都不一樣，編譯器剛剛好安排資料存放在那個地方

程式表現無法預期

```
i = 0;
while (source[i]!=0) {
   destination[i] = source[i];
   i = i + 1;
}
destination[i] = 0;
```

為什麼不檢查一下 **destination** 陣列的大小**??**

strncpy

- char ***strncpy**(char *destination, const char *source, size_t num);

strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

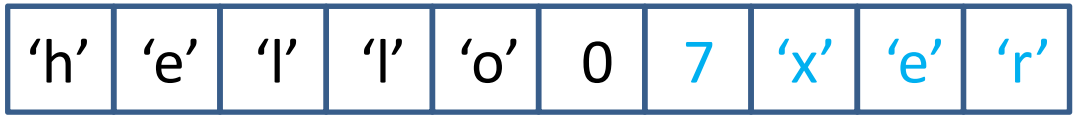
這樣子行了吧，strncpy 自己可以檢查一下目標的大小

strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

這樣子行了吧，strncpy 自己可以檢查一下目標的大小

char src[10] = “hello”;



strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

這樣子行了吧，strncpy 自己可以檢查一下目標的大小

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

char dest[8];

3	‘s’	0	-1	‘A’	12	‘d’	9

strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

這樣子行了吧，strncpy 自己可以檢查一下目標的大小

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy(dest, src, sizeof(dest));

char dest[8];

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**

strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

這樣子行了吧，strncpy 自己可以檢查一下目標的大小

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy(dest, src, sizeof(dest));

char dest[8];

sizeof(char[8])

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**

strncpy

- char ***strncpy***(char *destination, const char *source, size_t num);

這樣子行了吧，strncpy 自己可以檢查一下目標的大小

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy(dest, src, sizeof(dest));

char dest[8];

sizeof(char[8])

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**

最後這兩個 **0** 是讓它更安全嗎**?**

strncpy: error

- char ***strncpy***(char *destination, const char *source, size_t num);

strncpy: error

- char ***strncpy**(char *destination, const char *source, size_t num);

  請注意：

strncpy: error

- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy: error

- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

char dest[4];

3	‘s’	0	-1

strncpy: error

- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy(dest, src, sizeof(dest));

char dest[4];

‘h’	‘e’	‘l’	‘l’

糟糕!! 這樣子少了結束字元，不是合法的字串了

strncpy: error

- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

strncpy(dest, src, sizeof(dest));

char dest[4];

在這種情況下需要自己加上結束字元才能當作字串使用

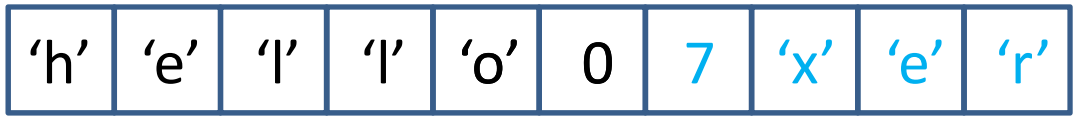
‘h’	‘e’	‘l’	‘l’

糟糕!! 這樣子少了結束字元，不是合法的字串了

strncpy: error

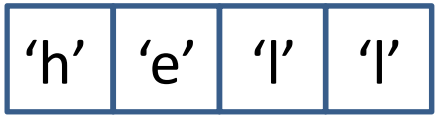
- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;



strncpy(dest, src, sizeof(dest));

char dest[4];



在這種情況下需要自己加上結束字元才能當作字串使用

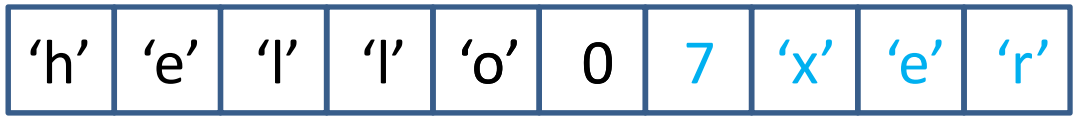
```
strncpy(dest, src, sizeof(dest));
dest[sizeof(dest)-1] = 0;
```

糟糕!! 這樣子少了結束字元，不是合法的字串了

strncpy: error

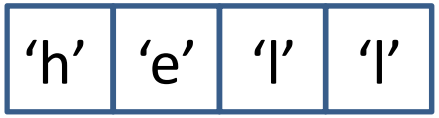
- char ***strncpy***(char *destination, const char *source, size_t num);

請注意： char src[10] = “hello”;



strncpy(dest, src, sizeof(dest));

char dest[4]; 在這種情況下需要自己加上結束字元才能當作字串使用




```
strncpy(dest, src, sizeof(dest));
dest[sizeof(dest)-1] = 0;
```

或是

```
char dest[4] = {0};
…
strncpy(dest, src, sizeof(dest)-1);
```

糟糕!! 這樣子少了結束字元，不是合法的字串了

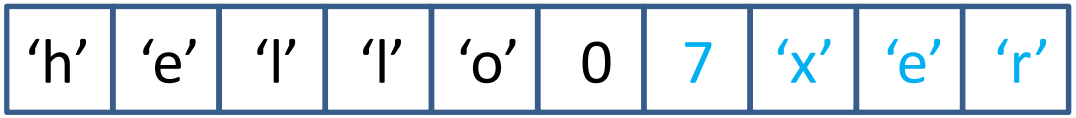
strncpy: error 2

- char ***strncpy**(char *destination, const char *source, size_t num);

strncpy: error 2

- char ***strncpy***(char *destination, const char *source, size_t num);

char src[10] = “hello”;



strncpy: error 2

- char ***strncpy***(char *destination, const char *source, size_t num);

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

char dest[8];

3	‘s’	0	-1	‘A’	12	‘d’	9

strncpy: error 2

- char ***strncpy***(char *destination, const char *source, size_t num);

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

三芝國小

strncpy(dest, src, 10);

不小心給錯了
不是來源字串
的長度, 是目標
字串的長度

char dest[8];

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**

0 **0**

最後這兩個 **0** 是根據你
給的 **10** 這個參數做的,
有雷, 程式表現無法預期!

strncpy: error 2

- char ***strncpy***(char *destination, const char *source, size_t num);

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

三芝國小

strncpy(dest, src, 10);

不小心給錯了
不是來源字串
的長度, 是目標
字串的長度

char dest[8];

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**

0 **0**

```
i = 0;
while (i<num&&source[i]!=0) {
   destination[i] = source[i];
   i = i + 1;
}
while (i<num) destination[i] = 0;
```

最後這兩個 **0** 是根據你
給的 **10** 這個參數做的,
有雷, 程式表現無法預期!

strncpy: error 2

- char ***strncpy***(char *destination, const char *source, size_t num);

char src[10] = “hello”;

‘h’	‘e’	‘l’	‘l’	‘o’	0	7	‘x’	‘e’	‘r’

三芝國小

strncpy(dest, src, 10);

不小心給錯了
不是來源字串
的長度, 是目標
字串的長度

char dest[8];

‘h’	‘e’	‘l’	‘l’	‘o’	0	**0**	**0**	**0**	**0**

```
i = 0;
while (i<num&&source[i]!=0) {
   destination[i] = source[i];
   i = i + 1;
}
while (i<num) destination[i] = 0;
```

最後這兩個 **0** 是根據你
給的 **10** 這個參數做的,
有雷, 程式表現無法預期!

如果目標字串很長, 補 **0** 很浪費時間

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間
- 請使用 **snprintf**(char *buf, size_t buflen, const char *format, …)

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間
- 請使用 **snprintf**(char *buf, size_t buflen, const char *format, …)
  - 正常情況下最多只會輸出 buflen-1 個字元, 最後補 0

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間
- 請使用 **snprintf**(char *buf, size_t buflen, const char *format, …)
  - 正常情況下最多只會輸出 buflen-1 個字元, 最後補 0
  - 不會像 strncpy() 補 0 到整個陣列結束

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間
- 請使用 **snprintf**(char *buf, size_t buflen, const char *format, …)
  - 正常情況下最多只會輸出 buflen-1 個字元, 最後補 0
  - 不會像 strncpy() 補 0 到整個陣列結束
- 如果只複製單一字串, 小心使用 **sprintf**(char *buf, const char *format, …) 也是很安全的選擇

snprintf or sprintf

- 擔心 BOF、擔心缺結束字元、又擔心補 0 花時間
- 請使用 **snprintf**(char *buf, size_t buflen, const char *format, …)
  - 正常情況下最多只會輸出 buflen-1 個字元, 最後補 0
  - 不會像 strncpy() 補 0 到整個陣列結束

- 如果只複製單一字串, 小心使用 **sprintf**(char *buf, const char *format, …) 也是很安全的選擇
  - char dest[**50**]
    sprintf(dest, "%**49**s", src);