由片語學習C程式設計

台灣大學資訊工程系劉邦鋒著

台灣大學劉邦鋒老師講授

August 19, 2016

第十一單元

字串

片語 1: 字串的宣告

```
1 char s[80];
```

- 字元是專門用來處理文字資訊。 而文字資訊一般都是成批出現。單獨使用字元來處理成批的文字資訊很麻煩，所以我們可使用**字串**來處理，
- C 以字元陣列代表字串。
- 字串的宣告就必須指定足夠的陣列長度，也就是字串的長度，來儲存字串中的字元。

範例程式 2: (string-size.c) 字串的所占的數

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5   char s[80];
6   printf("%d\n", sizeof(s));
7   return 0;
8 }
```

- 宣告一長度為 80 字元的字串，並印出其數。

輸出

```
1 80
```

範例程式 3: (string-init.c) 使用陣列的方式初始化一個字串

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    char s[80] = {'m', 'a', 'i', 'n', '(', ')', '\n',
6                  '{', '\n', '}', '\n'} ;
7    int i;
8    for (i = 0; i < 11; i++)
9      printf("%c", s[i]);
10   return 0;
11 }
```

- 既然字串是字元陣列，我們即可用使用陣列的初始化方式。

輸出

```
1 main()
2 {
3 }
```

字串處理慣例

- 陣列 s 雖然有80個字元，我們其實只對前面 11 個有興趣而已。
- C 字串處理的慣例是字串範圍到 '\0' 這個特殊字元為止。這個特殊字元中所有的皆為 0。
- 遵循這樣的慣例就能正確使用 C 有關處理字串的程式庫，例如 printf。

片語 4: 以 printf 印出字串

```
1 printf("%s", string);
```

範例程式 5: (string-printf.c) 以 printf 將字串一次印出

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    char s[80] = {'m', 'a', 'i', 'n', '(', ')', '\n',
6                  '{', '\n', '}', '\n', '\0'} ;
7    int i;
8    printf("%s", s);
9    return 0;
10 }
```

- 注意我們雖然在陣列 s 的最後加上 '\0' 但因未初始的元素是補 0，而這剛好就是 '\0'的值，所以程式裡的 '\0' 是可以不加的。

輸出

```
1 main()
2 {
3 }
```

學習要點

C 有關處理字串的程式庫的一個慣例是以 '\0' 作為字串的結尾。

字串常數

- 使用陣列的方式初始化一個字串非常麻煩。所以就有 **字串常數** 的出現。
- 字串常數和字元常數很類似，只不過字串常數是用雙引號 "。而且因為是陣列，所以可以有許多個字元在裡面。
- `printf` 輸出一個整數的參數 `"%d"` 就是字串常數。
- `"%d"` 包含兩個字元 – '%' 及 'd'。

特殊字元

字串常數用兩個雙引號 " 將字元括在一起。

範例程式 6: (string-init-double-quote.c) 使用字串常數的方式初始化

```
1 #include <stdio.h>
2 int main(void)
3 {
4   char s[80] = "main()\n{\n}\n";
5   printf("%s", s);
6   return 0;
7 }
```

- 字串常數中\n 代表換行字元 '\n'。
- 字串的最後面會加一個 '\0' 字元。

輸出

```
1 main()
2 {
3 }
```

空字串

- 有一個特別的字串常數叫做空字串，寫成 ""。
- 空字串的第一個字元就是 '\0'，所以它只佔 1 個。

範例程式 7: (empty-string.c) 空字串

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    char empty_string[] = "";
6    printf("sizeof(empty_string) = %d\n",
7           sizeof(empty_string));
8    printf("empty_string looks like %s\n",
9           empty_string);
10   return 0;
11 }
```

輸出

```
1 sizeof(empty_string) = 1
2 empty_string looks like
```

範例程式 8: (string-io.c) 字串的輸入

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    int i;
6    char string[10];
7
8    while (scanf("%s", string) != EOF) {
9      printf("%s\n", string);
10     for (i = 0; i < 10 && string[i] != '\0'; i++)
11       printf("%c ", string[i]);
12     printf("\n");
13   }
14 }
```

scanf 讀字串

- 用 scanf 的回傳值判定是否還有輸入字串需處理。
- 用 scanf 讀字串遇到空格就會斷開，而非讀整行。

輸入

```
1 This is a test.
```

輸出

```
1 This
2 T h i s
3 is
4 i s
5 a
6 a
7 test.
8 t e s t .
```

字元指標

片語 9: 依附一個字元陣列的字元指標也能當字串

```
1 char string[80];
2 char *ptr = string;
```

- 字串也常常以字元指標的類別出現，但字元陣列才有存放字元，所以字元指標必須依附一個字元陣列，才能夠進行關於字串的操作。

字元指標

- 以字元指標的類別操作字串是為了方便。
- 字元指標可以使用陣列的語法存取陣列元素。所以當字元指標指向字元陣列後，就可以很方便的存取陣列中的字元。

範例程式 10: (char-pointer.c) 依附一個字元陣列的字元指標

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char string[80];
7    char *ptr = string;;
8    int i;
9    scanf("%s", ptr);
10   printf("%s\n", ptr);
11   for (i = 0; i < strlen(ptr); i++)
12     printf("%c ", ptr[i]);
13   return 0;
14 }
```

輸入

```
1 programming
```

輸出

```
1 programming
2 p r o g r a m m i n g
```

片語 11: 字元指標類別的字串也可以有初始值

```
1 char *string = "programming";
```

- C 的特殊語法讓字元指標類別的字串也能有初始值。
- 編譯器會在唯讀中放一個字元陣列， 初始化成 "programming"，再將 string 指向這個字元陣列。

範例程式 12: (char-pointer-init.c) 三種字串

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char str1[80] = "programming";
7    char str2[] = "programming";
8    char *str3 = "programming";
9    printf("sizeof(str1) = %d\n", sizeof(str1));
10   printf("sizeof(str2) = %d\n", sizeof(str2));
11   printf("sizeof(str3) = %d\n", sizeof(str3));
12   return 0;
13 }
```

三種字串

1. `char string1[80]` 宣告了 80 個 `char`，所以佔 80 個。
2. `char string2[]` 沒說長度，所以編譯器自己算，結果是 12個 `char`，因為最後面需要再存一個 '\0'。
3. `char *string3` 本身是指標，只佔 4 個。

輸出

```
1 sizeof(str1) = 80
2 sizeof(str2) = 12
3 sizeof(str3) = 8
```

片語 13: 引入 string.h 標頭檔

```
1 #include <string.h>
```

- <string.h> 標頭檔中定義許多好用的函式。
- 使用時必須引入 <string.h>。

函式原型 14: (strlen)

```
1 int strlen(char *string);
```

- strlen (string length) 計算一個字串的長度。
- strlen 接受一個字元指標參數，這個指標指向要算長度的字串，算出字串長度並回傳呼叫者。

- 讀入一字串，印出其長度，並倒著印出這個字串。
- 需要先使用 `strlen` 計算出字串 `string` 的長度，這樣等一下決定第 `i` 的字元要跟那一個字元調換會比較方便。
- 注意我們使用類似調換整數的方法調換兩個字元。

範例程式 15: (string-length.c) 印出字串長度並倒著印出

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    int i, length;
7    char string[80], temp;
8    scanf("%s", string);
9    printf("%s\n", string);
10   length = strlen(string);
11   printf("%d\n", length);
12   for (i = 0; i < length / 2; i++) {
13     temp = string[i];
14     string[i] = string[length -i - 1];
15     string[length - i - 1] = temp;
16   }
17   printf("%s\n", string);
18   return 0;
19 }
```

輸入

```
1 programming
```

輸出

```
1 programming
2 11
3 gnimmargorp
```

範例程式 16: (my-strlen.c) 自己做 strlen

```
1  #include <stdio.h>
2  int my_strlen(char *string)
3  {
4    int i = 0;
5    while (i < 80 &&  string[i] != '\0')
6      i++;
7    return i;
8  }
9  int main(void)
10 {
11   int length;
12   char string[80];
13   scanf("%s", string);
14   printf("%s\n", string);
15   length = my_strlen(string);
16   printf("%d\n", length);
17   return 0;
18 }
```

輸入

```
1 programming
```

輸出

```
1 programming
2 11
```

範例程式 17: (char-pointer-strlen.c) 三種字串的 strlen

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char str1[80] = "programming";
7    char str2[] = "programming";
8    char *str3 = "programming";
9    printf("strlen(str1) = %d\n", strlen(str1));
10   printf("strlen(str2) = %d\n", strlen(str2));
11   printf("strlen(str3) = %d\n", strlen(str3));
12   return 0;
13 }
```

輸出

```
1 strlen(str1) = 11
2 strlen(str2) = 11
3 strlen(str3) = 11
```

函式原型 18: (strcpy-strcat)

```
1 char *strcpy(char *destination, char *source);
2 char *strcat(char *destination, char *source);
```

- strcpy (string copy) 將 source 參數字串複製到 destination 參數字串，並回傳 destination 的位址。
- source 字串的結束字元 '\0' 也會被複製到 destination 字串。
- strcat 與 strcpy 類似，但 strcat 會將第二個字串參數複製並接到第一個字串參數的後面。

範例程式 19: (string-copy.c) strcpy (string copy)

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6     char source[100];
7     char destination[100];
8     scanf("%s", source);
9     scanf("%s", destination);
10    printf("%s\n", destination);
11    strcpy(destination, source);
12    printf("%s\n", destination);
13    return 0;
14 }
```

輸入

```
1 source
2 destination
```

輸出

```
1 destination
2 source
```

範例程式 20: (string-concat.c) strcat (string concatenation)

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6     char source[100];
7     char destination[100];
8     scanf("%s", source);
9     scanf("%s", destination);
10    printf("%s\n", destination);
11    strcat(destination, source);
12    printf("%s\n", destination);
13    return 0;
14 }
```

輸入

```
1 source
2 destination
```

輸出

```
1 destination
2 destinationsource
```

緩衝區覆蓋

- 在使用 `strcpy` 及 `strcat` 時我們有可能嘗試將太長的字串複製進目的字串，以致超過目的字串的長度。
- 如果目的字串的後面還有重要的資料，這些資料就會被破壞。這個現象稱為緩衝區覆蓋 (buffer overrun)。

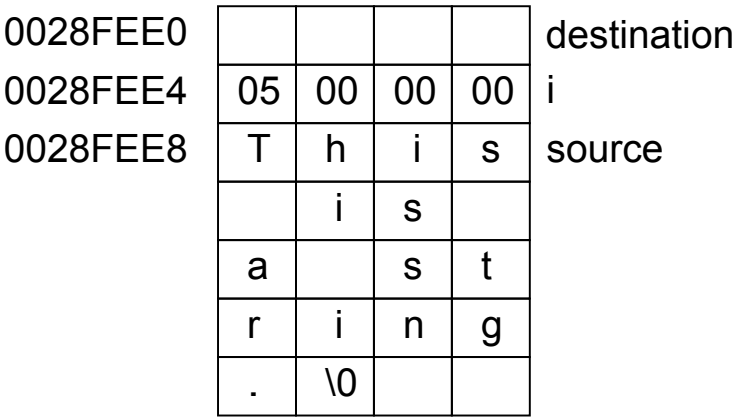
學習要點

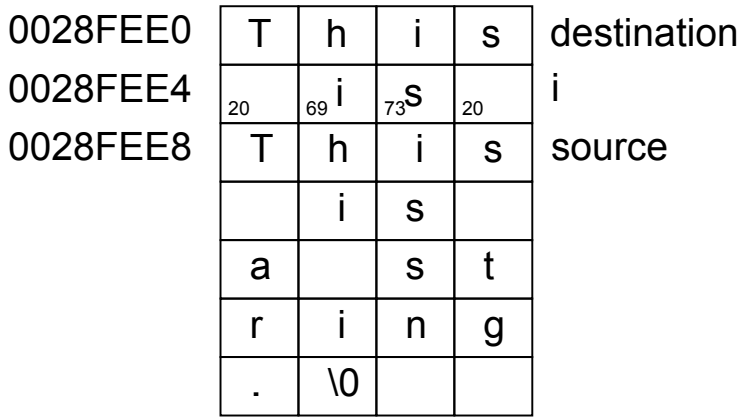
C 並不會檢查使用陣列時註標變數是否已超過範圍，所以在呼叫 strcpy 及 strcat 等複製字元進字串的函式時必須注意，以免發生緩衝區覆蓋。

範例程式 21: (string-overrun.c) strcpy 超過目的字串的長度

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void)
4  {
5    char source[40] = "This is a string.";
6    int i = 5;
7    char destination[4];
8    printf("The address of source is %p.\n",source);
9    printf("The size of source is %d\n",
10           sizeof(source));
11   printf("The address of i is %p\n", &i);
12   printf("The size of i is %d\n", sizeof(i));
```

```
14    printf("The address of destination is %p\n",
15           destination);
16    printf("The size of destination is %d\n",
17           sizeof(destination));
18    printf("strlen(source) is %d\n",strlen(source));
19    printf("i is %d\n", i);
20    strcpy(destination, source);
21    printf("i is %d\n", i);
22    printf("source is now <%s>\n", source);
23    printf("destination is now <%s>\n",destination);
24    return 0;
25  }
```





1. 首先 source 的前 4 個"This" 會被複製進 destination。
2. 再來因為 source 的長度是 17，source 接下來的4個" is " 會被複製進 i，將 i 完全覆蓋，

0028FEE0	T	h	i	s	destination
0028FEE4	20	69 i	73 s	20	i
0028FEE8	a		s	t	source
	r	i	n	g	
	.	\0	s	t	
	r	i	n	g	
	.	\0			

1. source 剩下的 9 個字元覆蓋 source 前面 9 個字元。
2. 原來 source 的 前 9 個字元 (source[0] 到 source[8]) 變成原來 source 的後 9 個字元 "a string."，而 source[9] 變成 '\0'

輸出

```
1  The address of source is 0x7fff58dc2700.
2  The size of source is 40
3  The address of i is 0x7fff58dc26ec
4  The size of i is 4
5  The address of destination is 0x7fff58dc26f0
6  The size of destination is 4
7  strlen(source) is 17
8  i is 5
9  i is 5
10 source is now <.>
11 destination is now <This is a string.>
```

函式原型 22: (strncpy-strncat)

```
1 char *strncpy(char *dest, char *source, int i);
2 char *strncat(char *dest, char *source, int i);
```

- strncpy 可由第三個參數 i 控制 "至多" 要複製幾個字元，避免緩衝區覆蓋。
- strncpy 不會幫你補結束字元 '\0'，而是要自己加。

範例程式 23: (string-ncopy.c) strncpy 可避免超過目的字串的長度

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void)
4  {
5    char source[40] = "This is a string.";
6    int i = 5;
7    char destination[4];
8    printf("The address of source is %p.\n",source);
9    printf("The size of source is %lu\n",
10           sizeof(source));
11   printf("The address of i is %p\n", &i);
12   printf("The size of i is %lu\n", sizeof(i));
```

```
14    printf("The address of destination is %p\n",
15           destination);
16    printf("The size of destination is %lu\n",
17           sizeof(destination));
18    printf("strlen(source) is %ul\n", strlen(source));
19    printf("i is %d\n", i);
20    strncpy(destination, source, 3);
21    destination[3] = '\0';
22    printf("i is %d\n", i);
23    printf("source is now <%s>\n", source);
24    printf("destination is now <%s>\n",destination);
25    return 0;
26 }
```

輸出

```
1  The address of source is 0x7fffdee8b350.
2  The size of source is 40
3  The address of i is 0x7fffdee8b33c
4  The size of i is 4
5  The address of destination is 0x7fffdee8b340
6  The size of destination is 4
7  strlen(source) is 171
8  i is 5
9  i is 5
10 source is now <This is a string.>
11 destination is now <Thi>
```

範例程式 24: (char-pointer-strcpy.c) 三種字串的 strcpy

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char string1[80] = "programming";
7    char string2[] = "programming";
8    char *string3 = "programming";
9    strcpy(string1, "more programming");
10   strcpy(string2, "more programming");
11   strcpy(string3, "more programming");
12   return 0;
13 }
```

1. `char string1[80]` 本身宣告了 80 個，所以沒問題。
2. `char string2[]` 長度只有 12，結果發生緩衝區覆蓋。
3. `char *string3 "programming"` 在唯讀， 一執行 `strcpy` 系統就宣告執行錯誤。

學習要點

以字元指標加字串常數建造出的字串放在唯讀中，所以不要嘗試寫資料進入這個字串。

函式原型 25: (strcmp-strncmp)

```
1 int strcmp(char *string1, *string2);
2 int strncmp(char *string1, *string2, int n);
```

- strcmp 比較兩個字串 string1 及 string 的大小。strncmp 只比較到 n 個字元。
  - 如果第一個字串 string1 比較小則回傳一負數，
  - 如果第一個字串 string1 比較大則回傳一正數，
  - 如果一樣大則回傳 0，
- 兩個字串的大小比法是由第一個字元開始按 ASCII 碼的大小開始比，如果相同則比第二個字元，直到比出大小或是其中一個字串結束為止。

字串排序

- 十二生肖的字串放在一個二維字串陣列中，第一維代表十二生肖，第二維代表字串。
- 使用兩層 `for` 迴圈實作泡沫排序法。
  - 第一層迴圈決定兩兩交換的範圍。
  - 第二層迴圈實際作兩兩交換。
- 使用 `strcmp` 比較兩個生肖字串。

範例程式 26: (string-sort.c) 將字串排序。

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char zodiac[12][40];
7    int i, j;
8    char temp[40];
9
10   for (i = 0; i < 12; i++)
11     scanf("%s", zodiac[i]);
```

```
13    for (i = 10; i >= 1; i--)
14      for (j = 0; j <= i; j++)
15        if (strcmp(zodiac[j], zodiac[j + 1]) > 0) {
16          strcpy(temp, zodiac[j]);
17          strcpy(zodiac[j], zodiac[j + 1]);
18          strcpy(zodiac[j + 1], temp);
19        }
20
21    for (i = 0; i < 12; i++)
22      printf("%s\n", zodiac[i]);
23    return 0;
24  }
```

輸入

```
 1 rat
 2 ox
 3 tiger
 4 hare
 5 dragon
 6 snake
 7 horse
 8 sheep
 9 monkey
10 rooster
11 dog
12 pig
```

輸出

```
 1 dog
 2 dragon
 3 hare
 4 horse
 5 monkey
 6 ox
 7 pig
 8 rat
 9 rooster
10 sheep
11 snake
12 tiger
```

將字串排序

zodiac

rat
ox
tiger
hare
dragon
snake
horse
sheep
monkey
rooster
dog
pig

zodiac

dog
dragon
hare
horse
monkey
ox
pig
rat
rooster
sheep
snake
tiger

範例程式 27: (string-pointer-sort.c) 使用指標陣列將字串排序

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char zodiac[12][40];
7    char *zptr[12];
8    int i;
9    int j;
10   char *temp;
11   for (i = 0; i < 12; i++) {
12     scanf("%s", zodiac[i]);
13     zptr[i] = zodiac[i];
14   }
```

```
16    for (i = 10; i >= 1; i--)
17      for (j = 0; j <= i; j++)
18        if (strcmp(zptr[j], zptr[j + 1]) > 0) {
19          temp = zptr[j];
20          zptr[j] = zptr[j + 1];
21          zptr[j + 1] = temp;
22        }
23    for (i = 0; i < 12; i++)
24      printf("%s\n", zptr[i]);
25    return 0;
26  }
```

字串排序

- 使用一個二維字元陣列存生肖字串，
- 使用另一個字元指標陣列 `zptr` 存生肖字串的起始。
- 用 `zptr[i]`當參數呼叫 `strcmp` 來比較字串。
- 泡沫排序法交換資料時，只要交換 `zptr[i]` 的值即可。省去 `strcpy` 的時間。

輸入

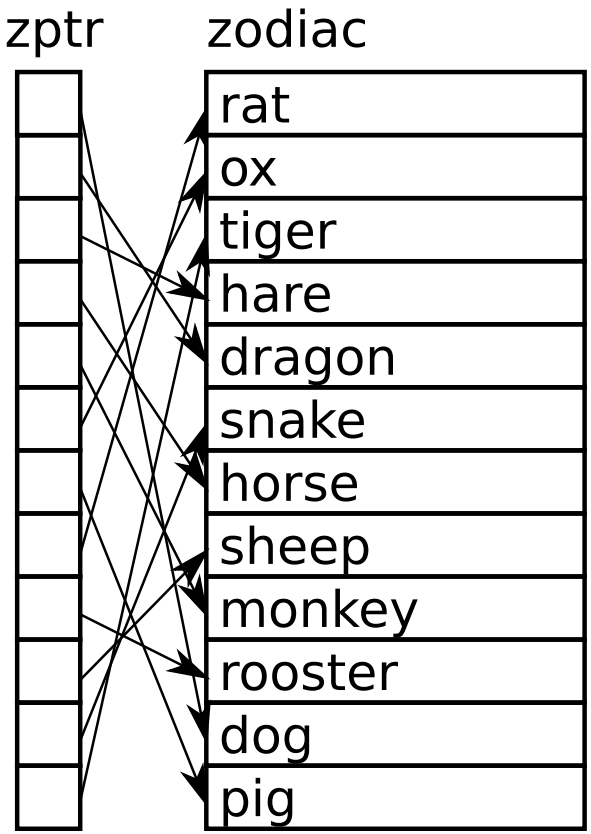
```
 1 rat
 2 ox
 3 tiger
 4 hare
 5 dragon
 6 snake
 7 horse
 8 sheep
 9 monkey
10 rooster
11 dog
12 pig
```

輸出

```
 1 dog
 2 dragon
 3 hare
 4 horse
 5 monkey
 6 ox
 7 pig
 8 rat
 9 rooster
10 sheep
11 snake
12 tiger
```

將字串排序


zptr
zodiac
rat
ox
tiger
hare
dragon
snake
horse
sheep
monkey
rooster
dog
pig


zptr
zodiac
rat
ox
tiger
hare
dragon
snake
horse
sheep
monkey
rooster
dog
pig

函式原型 28: (strchr-strrchr)

```
1 char *strchr(char *string, int c);
2 char *strrchr(char *string, int c);
```

- strchr 在一個字串 string 中搜尋一個字元 c。
  - 如果找到字元 c，則回傳第一個字元 c 的。
  - 如果找不到，則回傳 NULL。
- strrchr 由字串的後面往前面找 c。

範例程式 29: (strchr.c) 將一個路徑名中的各子目錄抽出

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char pathname[40];
7    char file[40][40];
8    int file_count = 0;
9    char *start = pathname;
10   char *slash;
11   int i;
12   scanf("%s", pathname);
```

```
14    if (*start == '/')
15      start++;
16    while (start != NULL) {
17      slash = strchr(start, '/');
18      if (slash == NULL) {
19        strcpy(file[file_count], start);
20        file_count++;
21        start = NULL;
22      } else {
23        strncpy(file[file_count], start,
24                slash - start);
25        file[file_count][slash - start] = '\0';
26        file_count++;
27        start = slash + 1;
28      }
29    }
```

```
31    for (i = 0; i < file_count; i++)
32      printf("%s\n", file[i]);
33    return 0;
34  }
```

抽出子目錄

- 以一個指標 `start` 記住子目錄的開始位置。
- 再以 `strchr` 找下一個 '/' 在何處。
  - 如果找不到，表示 `start` 後面已經沒有 '/'，此時直接將 `start` 開始的字串複製進 `file` 即可。
  - 如果找得到，則 `start` 一直到 '/' 之前都是子目錄的名字，我們可使用 `strncp` 將其複製進 `file`。
- 將所有收集到的子目錄印出。收集的方法是放進一個二維陣列，同時用一個計數器 `file_count` 紀錄收集到子目錄的個數。

輸入

```
1 /usr/local/bin/emacs
```

輸出

```
1 usr
2 local
3 bin
4 emacs
```

函式原型 30: (strstr)

```
1 char *strstr(char *string1, char *string2);
```

- strstr 在 string1 中找是否有 string2。
  - 若有出現則回傳在 string1 中 string2 出現的。
  - 若沒有出現則回傳 NULL。

範例程式 31: (strstr.c) 找包含 er 的生肖字串

```
3  int main(void)
4  {
5    char *ptr;
6    char zodiac[12][40];
7    int i;
8    for (i = 0; i < 12; i++)
9      scanf("%s", zodiac[i]);
10   for (i = 0; i < 12; i++) {
11     ptr = strstr(zodiac[i], "er");
12     if (ptr == NULL)
13       printf("No er in %s\n", zodiac[i]);
14     else
15       printf("er at %d-th in %s\n",
16              ptr - zodiac[i], zodiac[i]);
17   }
18   return 0;
19 }
```

輸入

```
1  rat
2  ox
3  tiger
4  hare
5  dragon
6  snake
7  horse
8  sheep
9  monkey
10 rooster
11 dog
12 pig
```

輸出

```
1  No er in rat
2  No er in ox
3  er at 3-th in tiger
4  No er in hare
5  No er in dragon
6  No er in snake
7  No er in horse
8  No er in sheep
9  No er in monkey
10 er at 5-th in rooster
11 No er in dog
12 No er in pig
```

函式原型 32: (strspn-strcspn)

```
1 char *strspn(char *string, char *chars);
2 char *strcspn(char *string, char *chars);
```

- strspn 掃描第一個參數 string，然後回傳 string 開始的部分有幾個連續字元出現在第二個字串參數 chars 中。
- strcspn 掃描第一個字串參數 string，然後回傳 string 字串開始的部分有幾個連續字元**不出現**在第二個字串參數 chars 中。

範例程式 33: (strcspn.c) 將一個路徑名中的各子目錄抽出

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void)
4  {
5    char letters[] = "abcdefghijklmnopqrstuvwxyz";
6    char pathname[40], file[40][40];
7    int file_count = 0;
8    char *start = pathname;
9    int i, skip_length, copy_length;
10   scanf("%s", pathname);
```

```
12    skip_length = strcspn(start, letters);
13    while (skip_length < strlen(start)) {
14      start += skip_length;
15      copy_length = strspn(start, letters);
16      strncpy(file[file_count], start, copy_length);
17      file[file_count][copy_length] = '\0';
18      file_count++;
19      start += copy_length;
20      skip_length = strcspn(start, letters);
21    }
22    for (i = 0; i < file_count; i++)
23      printf("%s\n", file[i]);
24    return 0;
25  }
```

- 假設各子目錄均由小寫字母構成。
- 先使用 `strcspn` 跳到第一個小寫字母， 再用 `strspn` 跳到第一個非字母，此時時跳過的必為字母， 就複製進 `file` 陣列。
- 用指標 `start` 記住現在處理到何處，
- `skip_length` 是 `strcspn` 回傳要跳過的字母數，
- `copy_length` 是 `strspn` 回傳要複製的字母數。

輸入

```
1 /usr/local/bin/emacs
```

輸出

```
1 usr
2 local
3 bin
4 emacs
```

函式原型 34: (strtok)

```
1 char *strtok(char *string, char *delimeters);
```

- strtok (string to token) 把第一個字串參數 string 切成一段一段的 token。
- token 是第一個字串中被第二個字串參數 delimeters 中的任何字元所隔開的部分。

1. 首先呼叫 strtok 一次。 第一個參數用要切成 token 的字串 string，第二個參數給用來分隔 token 的字元所組成的字串 delimeters。回傳值指定給一個指標變數 start。
2. 然後進入一個 while 迴圈檢查 start 是否為 NULL。如果 start 不為 NULL，start 就會指到原字串 string 中的一個字串。
3. 最後我們呼叫 strtok 使 start 跳到下一個 token。注意此時與第一次呼叫 strtok 不同，我們要使用 NULL 當第一個參數，而非 string。

片語 **35:** strtok

```
1 start = strtok(string, delimeters);
2 while (start != NULL) {
3   process string at start;
4   start = strtok(NULL, delimeters);
5 }
```

範例程式 36: (strtok.c) 使用 strtok 將字串切成 token

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6     char delimeters[] = "/";
7     char pathname[40];
8     char file[40][40];
9     int file_count = 0;
10    char *start = pathname;
11    int copy_length;
12    int i;
13    scanf("%s", pathname);
```

```
15    start = strtok(start, delimeters);
16    while (start != NULL) {
17      strcpy(file[file_count], start);
18      file_count++;
19      start = strtok(NULL, delimeters);
20    }
21    for (i = 0; i < file_count; i++)
22      printf("%s\n", file[i]);
23    printf("After strtok pathname becomes %s\n",
24           pathname);
25    return 0;
26  }
```

輸入

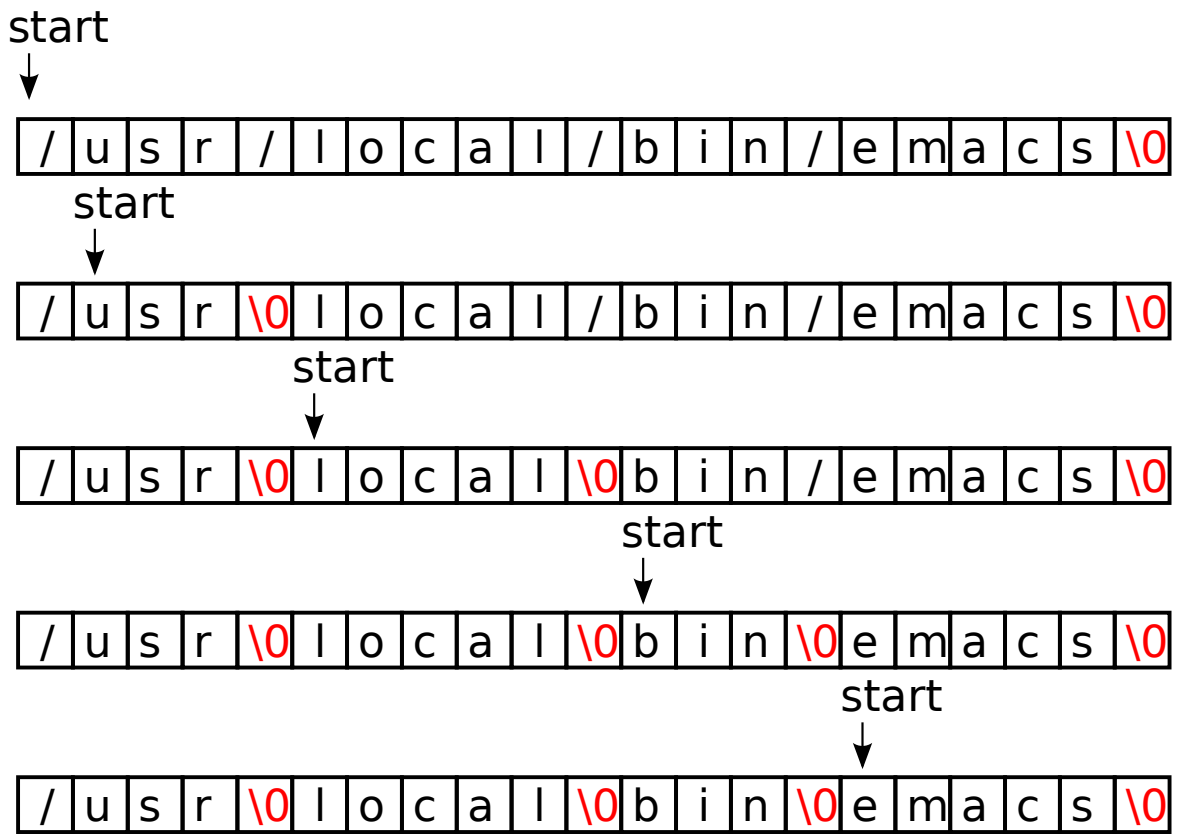
```
1 /usr/local/bin/emacs
```

輸出

```
1 usr
2 local
3 bin
4 emacs
5 After strtok pathname becomes /usr
```

- 注意因為 strtok 已經將 '\0' 加在 start 所指到的字串後面，可以直接用 strcpy 將 token 複製進 file，
- 之後再印出 pathname 就只得到 /usr， 因為 'r' 後面的 '/' 已經被改成 '\0'了。

將字串切成 token


start
/ u s r / l o c a l / b i n / e m a c s \0
start
/ u s r \0 l o c a l / b i n / e m a c s \0
start
/ u s r \0 l o c a l \0 b i n / e m a c s \0
start
/ u s r \0 l o c a l \0 b i n \0 e m a c s \0
start
/ u s r \0 l o c a l \0 b i n \0 e m a c s \0

範例程式 37: (strtok-vowl.c) 使用 strtok 將字串切成 token

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void)
4  {
5     char delimeters[] = "aeiou";
6     char pathname[40], file[40][40];
7     char *start = pathname;
8     int file_count = 0, copy_length, i;
9     scanf("%s", pathname);
```

```
11    start = strtok(start, delimeters);
12    while (start != NULL) {
13      strcpy(file[file_count], start);
14      file_count++;
15      start = strtok(NULL, delimeters);
16    }
17    for (i = 0; i < file_count; i++)
18      printf("%s\n", file[i]);
19    printf("After strtok pathname becomes %s\n",
20            pathname);
21    return 0;
22  }
```

輸入

```
1 /usr/local/bin/emacs
```

輸出

```
1 /
2 sr/l
3 c
4 l/b
5 n/
6 m
7 cs
8 After strtok pathname becomes /
```