

程式設計 (一)

CH13. 檔案處理

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Last Semester, 2021

本章目錄

1. 檔案和串流
2. 循序讀取檔案與開啟模式
3. 循序讀取檔案的存取
4. 循序讀取檔案的更新
5. 隨機存取檔案與開啟模式
6. 建立隨機存取檔案
7. 隨機的寫入、讀出資料

檔案和串流

儲存在變數和陣列裡的資料是暫時的，當程式結束後這種資料都將會遺失。程式會利用檔案 (Files) 來長期保存大量資料。

檔案和串流

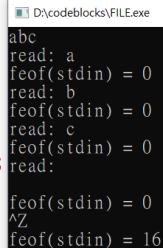
C 將每個檔案視為連續的位元組串流。在程式開始執行時，會有三個檔案，還有和這些檔案結合的資料流也會自動開啟，它們是標準輸入 (standard input)、標準輸出 (standard output) 和標準錯誤 (standard error)。

標準輸入、標準輸出、和標準錯誤會分別以下列三個指標來進行操作：stdin、stdout 和 stderr。

檔案和串流

我們平常在判斷輸入是否結束的 EOF，事實上是用來判斷檔案內容是否結束的旗標值 (flag)。
我們也可以使用函式 feof 來判斷 stdin (或其他檔案指標) 是否已經在檔案結尾。

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     char tmp;
6     while(scanf("%c", &tmp) != EOF){
7         printf("read: %c\n", tmp);
8         printf("feof(stdin) = %d\n", feof(stdin));
9     }
10    printf("feof(stdin) = %d\n", feof(stdin));
11 }
```



```
D:\codeblocks\FILE.exe
abc
read: a
feof(stdin) = 0
read: b
feof(stdin) = 0
read: c
feof(stdin) = 0
read:
feof(stdin) = 0
^Z
feof(stdin) = 16
```

循序讀取檔案與開啟模式

循序讀取檔案與開啟模式

一個簡單的開檔寫檔程式

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt","w");
6     if(fptr == NULL)
7         puts("Open file failed");
8     else{
9         fprintf(fptr, "My first file.\n");
10        puts("Printed");
11    }
12    fclose(fptr);
13 }
```



fopen 回傳值 - FILE 結構指標

當開啟檔案之後，檔案的 FCB (file control block，檔案控制塊) 會複製到記憶體中，而 C 會建立一個 FILE 結構，此 FILE 結構包含了對應到的 FCB 的描述子，並由 fopen 回傳此 FILE 結構的位址。

之後再進行檔案存取時，C 會根據 FILE 結構中的描述子找到 FCB，並呼叫作業系統提供的服務 (system call)，來完成硬碟中檔案的輸入輸出。

開檔與關檔

- `FILE* fopen(const char* filename, const char* mode);`
開啟檔案，`filename` 為檔案路徑與檔名，`mode` 為開檔模式，若開檔成功則回傳此檔案的 `FILE` 結構位址，若開檔失敗則回傳 `NULL`。
- `int fclose(FILE* stream);`
關閉檔案，`stream` 為欲關閉檔案的 `FILE` 結構位址，若關檔成功則回傳 `0`，否則回傳 `EOF (-1)`。

檔案開啟模式

模式	意思	說明
"r"	read	開啟用來讀取的檔案。
"w"	write	建立用來寫入的檔案，若檔案已經存在則會刪除原本的內容。
"a"	append	開啟或建立一個用來將資料寫到檔案結尾的檔案。

循序讀取檔案的存取


格式化輸入輸出函式

- `int fscanf(FILE* stream, const char* format, ...);`
檔案格式化輸入。
- `int fprintf(FILE* stream, const char* format, ...);`
檔案格式化輸出。

循序讀取檔案的存取

使用 fprintf 印出文字到檔案中

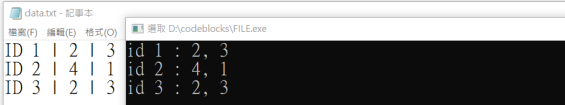
```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt","w");
6     int data[3][2] = { {2, 3}, {4, 1}, {2, 3} };
7     if(fptr == NULL)
8         puts("Open file failed");
9     else{
10         for(int i = 0; i < 3; i++){
11             fprintf(fptr, "ID %d | %d | %d\n", i + 1, data[i][0], data[i][1]);
12             puts("Printed");
13         }
14         fclose(fptr);
15     }
16 }
17
```



循序讀取檔案的存取

使用 fscanf 讀入檔案中的資料

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt","r");
6     int id, vala, valb;
7     if(fptr == NULL)
8         puts("Open file failed");
9     else{
10         while(fscanf(fptr, "ID %d | %d | %d\n", &id, &vala, &valb) != EOF){
11             printf("id %d : %d, %d\n", id, vala, valb);
12         }
13     }
14     fclose(fptr);
15 }
16
```



The screenshot shows a code editor window titled 'data.txt - 記事本' and a terminal window titled 'D:\codeblocks\FILE.exe'. The code in the editor is a C program that opens 'data.txt' for reading and uses `fscanf` to read lines of data in the format 'ID %d | %d | %d\n'. The program prints the data to the console as 'id %d : %d, %d\n'. The terminal output shows the following lines:

```
id 1 : 2, 3
id 2 : 4, 1
id 3 : 2, 3
```

字元輸入輸出函式

- `int fgetc(FILE* stream);`
從 `stream` 輸入一個字元。
- `int fputc(int character, FILE* stream);`
輸出一個字元到 `stream` 。

字串輸入輸出函式

- `char* fgets(char* str, int num, FILE* stream);`
從 `stream` 輸入字串直到字串長度到達 `num-1` 或讀到換行符號或檔案結尾。(須注意 `fgets` 會將 `'\n'` 讀入，而 `gets` 不會將 `'\n'` 讀入。)
- `int fputs(const char* str, FILE* stream);`
輸出一個字串到 `stream`。(須注意：`fputs` 並不會在輸出字串之後自動換行。)

設定檔案位置指標

- `void rewind(FILE* stream);`
將檔案位置指標 (下一個要讀取或寫入的位元組) 重新指向檔案開頭。

循序讀取檔案的存取

若先印出資料後將檔案位置指標重新指回到檔案開頭，
之後印出的資料會覆蓋原本的資料。

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt", "w");
6     char s[100];
7     if(fptr == NULL)
8         puts("Open file failed");
9     else{
10         fputs("AAAAAAA", fptr);
11         rewind(fptr);
12         fputs("BBBB", fptr);
13         puts("Printed");
14     }
15     fclose(fptr);
16 }
```



循序讀取檔案的存取

- `int fseek(FILE* stream, long int offset, int origin);`
將檔案位置指標進行位移，`origin` 可以為以下三種其中一種。
 - `SEEK_SET`: 檔案開頭
 - `SEEK_CUR`: 目前檔案位置指標
 - `SEEK_END`: 檔案結尾
- `fseek(fPtr, 0, SEEK_SET);` 的效果如同 `rewind(fPtr);`。

循序讀取檔案的存取

下方程式碼第 11 行，使用 `fseek` 將檔案位置指標往前 4 個位元組。

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt","w");
6     char s[100];
7     if(fptr == NULL)
8         puts("Open file failed");
9     else{
10         fputs("AAAAAAA", fptr);
11         fseek(fptr, -4, SEEK_CUR);
12         fputs("B", fptr);
13         puts("Printed");
14     }
15     fclose(fptr);
16 }
```



D:\codeblocks\FILE.exe
Printed
AAAABAAA

循序讀取檔案的更新

循序讀取檔案的更新

在前面介紹的三個檔案開啟模式"r"、"w"、"a"都只能讀檔或只能寫檔，但有沒有同時支援讀寫的開啟模式呢？有的，這時候需要使用"r+"、"w+"、"a+"。

- "r+"：read / update
- "w+"：write / update
- "a+"：append / update

開啟模式比較

	支援讀檔	支援寫檔	檔案開啟前須存在	開啟時清除資料	開檔時檔案位置指標
r	是	否	須	不會	開頭處
w	否	是	不須	會	開頭處
a	否	是	不須	不會	結尾處
r+	是	是	須	不會	開頭處
w+	是	是	不須	會	開頭處
a+	是	是	不須	不會	結尾處

循序讀取檔案的更新

使用“w+” 模式輸出文字後再讀入剛剛輸出的文字

```
1 //FILE
2 #include <stdio.h>
3
4 int main(){
5     FILE* fptr = fopen("data.txt","w+");
6     char s[100];
7     if(fptr == NULL)
8         puts("Open file failed");
9     else{
10         fputs("I have an apple\n", fptr);
11         fputs("I have a pan\n", fptr);
12         puts("Printed");
13         rewind(fptr);
14         while(fgets(s, 100, fptr) != 0){
15             printf("Read: %s", s);
16         }
17     }
18     fclose(fptr);
19 }
```



隨機存取檔案與開啟模式

隨機存取檔案與開啟模式

先前介紹的輸出文字到檔案中的函式，其函式所產生的紀錄 (每筆資料)，其長度不一定每個都一樣。而隨機存取檔案 (或稱二進制檔案) 中的每一筆紀錄通常都具固定長度，可以用來直接存取。

隨機存取檔案開啟模式

	支援讀檔	支援寫檔	檔案開啟前須存在	開啟時清除資料	開檔時檔案位置指標
rb	是	否	須	不會	開頭處
wb	否	是	不須	會	開頭處
ab	否	是	不須	不會	結尾處
rb+	是	是	須	不會	開頭處
wb+	是	是	不須	會	開頭處
ab+	是	是	不須	不會	結尾處

建立隨機存取檔案

建立隨機存取檔案

在隨機存取檔案的存取，我們不會使用 `fscanf`、`fprintf` 等函式，而是使用 `fread`、`fwrite`，並通常會一次寫入一個 `struct` 作為一筆紀錄。

- `size_t fread(void* ptr, size_t size, size_t count, FILE* stream);`
讀取二進制資料。
- `size_t fwrite(const void* ptr, size_t size, size_t count, FILE* stream);`
寫入二進制資料。

建立隨機存取檔案

例如下列的 csv 檔，我們想把它建成一個隨機存取檔案。



classdata.csv - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
name,height,weight
Kevin,170.3,77.5
Jon,165.8,68.9
Vivian,158.1,48.6
Jason,172.3,89.3
Tim,162.4,60.2
```

建立隨機存取檔案

首先我們應該先定義一筆紀錄的結構：

```
typedef struct{  
    char name[10];  
    float weight;  
    float height;  
}Student;
```



建立隨機存取檔案

```
1 //FILE
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 typedef struct{
6     char name[10];
7     float weight;
8     float height;
9 }Student;
10
11 int main(){
12     FILE* rfile = fopen("classdata.csv","r");
13     FILE* wfile = fopen("classdata.dat","wb");
14     Student tmp;
15     char buf[100], t;
16     if(rfile == NULL || wfile == NULL)
17         puts("Open file failed");
```

建立隨機存取檔案

承上一頁，開檔之後讀入純文字檔 classdata.csv 的資料，並寫入二進制檔案 classdata.dat。

```
18 else{
19     fgets(buf, 100, rfile);
20     while(1){
21         t = fscanf(rfile, "%[^,], %f, %f\n",
22                 tmp.name, &tmp.height, &tmp.weight);
23         if(t == EOF)
24             break;
25         printf("Read: %s %f %f\n",
26                tmp.name, tmp.height, tmp.weight);
27         fwrite(&tmp, sizeof(Student), 1, wfile);
28     }
29 }
30 fclose(rfile);
31 fclose(wfile);
32 }
33
34
```



classdata.csv - 記事本

檔案(F)	編輯(E)	格式(O)	檢視(V)	說明
name,height,weight	Kevin	@	汨	*CJon n
Jon,165.8,68.9	@	ffBB?	CJason	@ ?氫汨,C
Vivian,158.1,48.6				Tim n
Jason,172.3,89.3				@ 汨pBff"C
Tim,162.4,60.2				

classdata.dat - 記事本

檔案(F)	編輯(E)	格式(O)	檢視(V)	說明
Read: Kevin	170.300003	77.500000		
Read: Jon	165.800003	68.900002		
Read: Vivian	158.100006	48.599998		
Read: Jason	172.300003	89.300003		
Read: Tim	162.399994	60.200001		

隨機的寫入、讀出資料

依序讀取二進制檔案資料

延續上一節建立的 `classdata.dat` 檔案，我們想將此檔案的資料讀取出來。

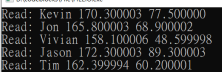
在讀取二進制檔案時，須定義與寫入二進制檔案時相同的 `struct`。

```
1 //FILE
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 typedef struct{
6     char name[10];
7     float weight;
8     float height;
9 }Student;
```

隨機的寫入、讀出資料

承上一頁，使用 fread 讀入資料，並用 feof 判斷是否已經讀到檔案結尾。

```
1 //FILE
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 typedef struct{
6     char name[10];
7     float weight;
8     float height;
9 }Student;
10
11 int main(){
12     FILE* rfile = fopen("classdata.dat","rb");
13     Student tmp;
14     if(rfile == NULL)
15         puts("Open file failed");
16     else{
17         while(1){
18             fread(&tmp, sizeof(Student), 1, rfile);
19             iffeof(rfile)
20                 break;
21             printf("Read: %s %f %f\n",
22                 tmp.name, tmp.height, tmp.weight);
23         }
24     }
25     fclose(rfile);
26 }
27
```



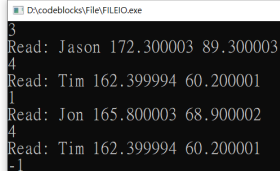
隨機的讀取資料

因為取二進制檔案的每個紀錄都是根據同樣的 struct ，
每個紀錄所佔的位元組均相同，因此我們可以使用
fseek 和 fread 來快速的隨機讀取紀錄。

隨機的寫入、讀出資料

輸入整數 n ，讀取檔案中第 n 個紀錄。

```
11 int main(){
12     FILE* rfile = fopen("classdata.dat", "rb");
13     Student tmp;
14     int n;
15     if(rfile == NULL)
16         puts("Open file failed");
17     else{
18         while(scanf("%d", &n) && n >= 0){
19             fseek(rfile, sizeof(Student) * n, SEEK_SET);
20             fread(&tmp, sizeof(Student), 1, rfile);
21             printf("Read: %s %f %f\n",
22                 tmp.name, tmp.height, tmp.weight);
23         }
24     }
25     fclose(rfile);
26 }
27
```



```
D:\codeblocks\File\FILEIO.exe
3
Read: Jason 172.300003 89.300003
4
Read: Tim 162.399994 60.200001
1
Read: Jon 165.800003 68.900002
4
Read: Tim 162.399994 60.200001
-1
```

隨機的寫入、讀出資料

隨機的寫入資料

```
5 typedef struct{
6     char name[10];
7     float weight;
8     float height;
9 }Student;
10
11 Student readData(FILE* f, int n){
12     Student tmp;
13     fseek(f, sizeof(Student) * n, SEEK_SET);
14     fread(&tmp, sizeof(Student), 1, f);
15     printf("Read: %s %f %f\n", tmp.name, tmp.height, tmp.weight);
16     return tmp;
17 }
18
19 void writeData(FILE* f, int n, Student s){
20     fseek(f, sizeof(Student) * n, SEEK_SET);
21     fwrite(&s, sizeof(Student), 1, f);
22 }
```


隨機的寫入、讀出資料

承上一頁，此程式可讓使用者選擇檔案中的第 n 筆資料進行修改。

```
5 typedef struct{
6     char name[10];
7     float weight;
8     float height;
9 }Student;
10
11 Student readData(FILE* f, int n){
12     Student tmp;
13     fseek(f, sizeof(Student) * n, SEEK_SET);
14     fread(&tmp, sizeof(Student), 1, f);
15     printf("Read: %s %f %f\n", tmp.name, tmp.height, tmp.weight);
16     return tmp;
17 }
18
19 void writeData(FILE* f, int n, Student s){
20     fseek(f, sizeof(Student) * n, SEEK_SET);
21     fwrite(&s, sizeof(Student), 1, f);
22 }
```

參考資料： Deitel, H. M., & Deitel, P. J. (2015). C: How to program.
Upper Saddle River, N.J: Prentice Hall.