

程式設計

Ch10. File Processing

Chuan-Chi Lai 賴傳淇

Department of Communications Engineering
National Chung Cheng University

Spring Semester, 2024

- 1 資料流 (Stream)
- 2 檔案開啓模式 (File Open Modes)
- 3 純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)
- 4 二進位檔案的建立與讀取 (Creating and Reading of Binary Files)
- 5 隨機地讀寫資料 (Reading and Writing Data Randomly)

資料流 Stream

資料流 (Stream)

- 以往我們處理資料時，均將資料存放在變數和陣列當中。變數和陣列裡的資料均位於記憶體，當程式結束後這種資料都將會遺失。
- 程式會利用檔案 (files) 來長期地保存大量的資料。
- 另外，程式有時也須與環境中的硬體溝通，例如鍵盤、螢幕等周邊設備，也是使用檔案的概念溝通。
- 透過讀取和寫入檔案，以便與目標環境溝通的程式。檔案可以是：
 - 可以重複讀取與寫入的資料集。
 - 程式所產生的位元組資料流 (例如 pipe)。
 - 接收自或傳送到週邊裝置的位元組資料流。

資料流 (Stream)

- C 將每個檔案視為連續的位元組串流。
- 在程式開始執行時，會預期下列三個檔案已經開啓：標準輸入 (standard input)、標準輸出 (standard output) 和標準錯誤 (standard error)，並自動開啓與這些檔案結合的資料流 (stream)。
- 標準輸入、標準輸出、和標準錯誤可以分別以下列三個檔案結構指標來進行操作：stdin、stdout 和 stderr。

資料流 (Stream)

- 資料流 (stream) 提供程式與檔案之間的通訊管道。例如標準輸入流讓程式能從鍵盤讀取資料，標準輸出流讓程式可以將資料印出至終端。
- 資料流包含三項基本作業：
 - 讀取 (Reading)：從資料流將資料傳送至資料結構。
 - 寫入 (Writing)：從資料來源將資料傳送至資料流。
 - 搜尋 (Seeking)：查詢和修改資料流內的目前位置。

檔案開啓模式 File Open Modes

檔案開啓模式 (File Open Modes)

- 在程式開啓檔案時，需要指定檔案的開啓模式 (file open mode)。開啓模式以字串表示，如下表：

模式	說明
"r" read	開啓用來讀取的檔案。
"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	開啓或建立一個用來將資料寫到檔案結尾的檔案。

檔案開啓模式 (File Open Modes)

```
FILE *fopen(const char *filename, const char *mode);  
int fclose(FILE *stream);
```

- 使用 `fopen` 函式可以開啓檔案，參數 `filename` 為開檔的檔案路徑與名稱 (可使用絕對路徑或相對路徑)，參數 `mode` 為檔案開啓模式。
- 若開檔成功，`fopen` 會回傳該檔案的檔案結構指標，若開檔失敗時會回傳 `NULL`。
- 使用 `fclose` 函式可以關閉檔案，若關檔成功回傳 `0`，否則回傳 `EOF`。

檔案開啓模式 (File Open Modes)

- 當開啓檔案之後，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。

檔案開啓模式 (File Open Modes)

- 如當 mode 爲 “r”(即 read mode) 時，若 filename 於電腦中不存在，會導致開檔失敗。在檔案操作前，請記得先檢查是否開檔成功。

```
1 #include <stdio.h>
2 #include <assert.h>
3
4 int main() {
5     FILE *fptr = fopen("path/to/your/file/testfile.txt", "r");
6     if (fptr == NULL) {
7         puts("Failed to open file");
8         assert(0); //Exception occurred, all abort
9     }
10    puts("File opened successfully.");
11    // Do something with the file
12    // ...
13    fclose(fptr);
14 }
```

純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

純文字檔案的建立與讀取 Creating and Reading of Plain Text Files

純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

- 標準輸入輸出函式庫 `<stdio.h>` 提供了純文字與二進位檔案的讀寫。
- 本章著重於純文字檔案的字元與字串讀寫，將介紹以下函式：
 - 格式化輸入輸出函式：`fscanf`、`fprintf`
 - 字元輸入輸出函式：`fgetc`、`getc`、`fputc`
 - 字串輸入輸出函式：`fgets`、`fputs`

純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

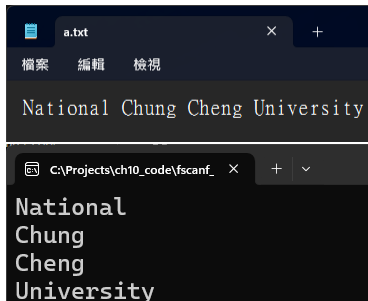
```
int fscanf(FILE *stream, const char *format, ...);  
int fprintf(FILE *stream, const char *format, ... );
```

- 如同 scanf 與 printf 可以格式化地從 stdin 讀取資料或寫入資料至 stdout，fscanf 與 fprintf 則可以格式化地由指定資料流讀入或輸出。
- fscanf 會回傳成功讀取的項目個數，當 end of file 則會回傳 EOF。
- fprintf 會回傳寫入的字元數。

純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

- 以下是使用 fscanf 與 fprintf 輸入輸出的範例：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     FILE *fptr = fopen("a.txt", "w");
6     char str[100];
7     fprintf(fptr, "%s\n",
8         "National Chung Cheng University");
9     fclose(fptr);
10    fptr = fopen("a.txt", "r");
11    while (fscanf(fptr, "%s", str) != EOF)
12        puts(str);
13    fclose(fptr);
14 }
```



純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

```
int fgetc(FILE *stream);  
int fputc(int character, FILE *stream);
```

- fgetc 會從 stream 讀取一個字元，並回傳該字元的 ASCII 碼。若讀取至 EOF，則會回傳 EOF(值為-1)。
- fputc 會輸出一個字元 character 至 stream，並回傳輸出的字元。
- 另外，函式 getc 與 fgetc 是等價的，與 getchar 並不相同。

純文字檔案的建立與讀取 (Creating and Reading of Plain Text Files)

```
char *fgets(char *str, int num, FILE *stream );  
int fputs(const char *str, FILE *stream );
```

- fgets 會由 stream 讀入字串到位址 str，直到 '\n' 或直到 EOF (fgets 會將 '\n' 存入字串 str)。
- 若讀取成功則回傳 str，若讀取開始時即遇到 EOF，並無讀取到任何文字則回傳 NULL 且不更動位址 str 的內容。
- fputs 會輸出字串 str 至 stream，且不會在最後換行。

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

二進位檔案的建立與讀取 Creating and Reading of Binary Files

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

- 先前介紹的輸出存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 (或稱二進位檔案) 中的每一筆紀錄通常都具固定長度，可以用來直接存取。
- 在二進位開啓檔案時，需要指定二進位檔案的開啓模式，如下表：

模式	說明
"rb"	開啓用來讀取的二進位檔案。
"wb"	建立用來寫入的二進位檔案，若檔案已經存在，則會刪除原本的內容。
"ab"	開啓或建立一個用來將資料寫到檔案結尾的二進位檔案。

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

```
size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream);  
size_t fread(void *ptr, size_t size, size_t count, FILE *stream);
```

- 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 `fwrite` 與 `fread`。通常會以一個 `struct` 作為一筆紀錄。
- `fwrite` 會從 `ptr` 讀取連續記憶體資料，每筆紀錄大小為 `size`，共 `count` 筆紀錄，並寫入至 `stream`。
- `fread` 會從 `stream` 讀取，每筆紀錄大小為 `size`，共 `count` 筆紀錄，並寫入至記憶體位址 `ptr` 之後的連續空間。
- `fread` 會回傳成功讀取的紀錄個數，當 EOF 時回傳 0。

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

- 下兩頁的範例會示範二進位的寫檔與讀檔，寫檔與讀檔的兩支程式都有相同的標頭與結構定義如下：

```
1  #include <stdio.h>
2  #include <assert.h>
3
4  typedef struct {
5      char name[32];
6      int sid;
7      double score;
8  } Student;
9
```

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

● 將輸入的資料寫檔：

```
10 int main() {
11     int n;
12     FILE *fptr = fopen("student.dat", "wb");
13     assert(fptr);
14     printf("Number of students: ");
15     scanf("%d", &n);
16     for (int i = 1; i <= n; i++) {
17         Student tmp = {0};
18         printf("Student %d.\n Name: ", i);
19         scanf(" "); //skip blank character
20         gets(tmp.name);
21         printf(" Student ID: ");
22         scanf("%d", &tmp.sid);
23         printf(" Score: ");
24         scanf("%d", &tmp.score);
25         fwrite(&tmp, sizeof(Student), 1, fptr);
26     }
27     fclose(fptr);
28     puts("File saved.");
29 }
```

student.dat

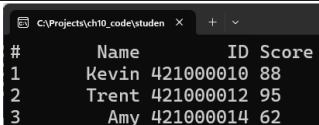
Offset	Hex	Decoded Text
00000000	48 65 76 69 00 00 00 00 00 00 00 00 00 00 00 00	Kevin
00000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020	CA 5C 7F 18 5A 00 00 00 4A 75 73 74 69 6E 00 00	.. \ . . Z . . Justin . .
00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040	00 00 00 00 00 00 00 00 CC 5C 7F 18 61 00 00 00 \ . .
00000050	4A 69 6D 6D 79 00 00 00 00 00 00 00 00 00 00 00	Jimmy
00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070	CE 5C 7F 18 41 00 00 00	.. \ . . A . .

```
C:\Projects\ch10_code\student x + v
Number of students: 3
Student 1.
Name: Kevin
Student ID: 421000010
Score: 88
Student 2.
Name: Trent
Student ID: 421000012
Score: 95
Student 3.
Name: Amy
Student ID: 421000014
Score: 62
File saved.
```

二進位檔案的建立與讀取 (Creating and Reading of Binary Files)

- 讀取已經建立的檔案：

```
11 int main() {  
12     int n = 1;  
13     Student tmp;  
14     FILE *fptr = fopen("student.dat", "rb");  
15     assert(fptr); // IF fptr == 0, the program will terminate  
16     printf("# %10s %9s %2s\n", "Name", "ID", "Score");  
17     while (fread(&tmp, sizeof(Student), 1, fptr)) {  
18         printf("%d %10s %9d %2d\n",  
19             n++, tmp.name, tmp.sid, tmp.score);  
20     }  
21     fclose(fptr);  
22 }
```



#	Name	ID	Score
1	Kevin	421000010	88
2	Trent	421000012	95
3	Amy	421000014	62

隨機地讀寫資料

Reading and Writing Data Randomly

隨機地讀寫資料 (Reading and Writing Data Randomly)

- 在這個章節，我們會介紹能夠同時進行讀寫的檔案開啓模式，以及介紹如何控制檔案位置指標 (file position pointer，即接下來在檔案中會輸入或輸出的位置)。
- 我們將會介紹以下用來進行 repositioning 的函式：
 - rewind：將檔案位置指標重新指回到檔案開頭。
 - fseek：將檔案位置指標進行位移至指定位置。

隨機地讀寫資料 (Reading and Writing Data Randomly)

- 以下是檔案開啓模式的比較，將 r、w、a 後方加上 + 表示開啓一個用來更新資料的檔案，可同時支援讀寫。

模式	讀檔	寫檔	當檔案已存在	當檔案不存在	repositioning
r	可	不可	從頭開始讀取	開啟失敗	可
w	不可	可	刪除內容	建立檔案	可
a	不可	可	從尾端開始寫入	建立檔案	無視
r+	可	可	從頭開始讀取	開啟失敗	可
w+	可	可	刪除內容	建立檔案	可
a+	可	可	從尾端開始寫入	建立檔案	寫入位置無視

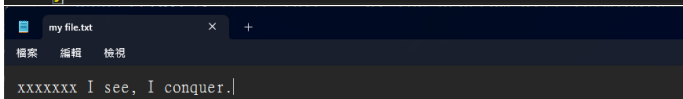
- 若須開啓二進位檔案，則可在 r、w、a 後方或 + 號後方加上 b，如 rb、wb、ab、r+b、w+b、a+b(或 ab+、wb+、ab+)

隨機地讀寫資料 (Reading and Writing Data Randomly)

```
void rewind( FILE *stream );
```

- 將檔案位置指標重新指回到檔案開頭。當想要重頭開始讀檔時可以使用。當呼叫 `rewind` 並開始寫檔，新的資料會覆蓋掉舊的資料。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  int main() {
6      FILE *fptr = fopen("my file.txt", "w");
7      assert(fptr); // IF fptr == 0, the program will terminate
8      fputs("I come, I see, I conquer.", fptr);
9      rewind(fptr);
10     fputs("xxxxxxx", fptr);
11     fclose(fptr);
12 }
```



隨機地讀寫資料 (Reading and Writing Data Randomly)

```
int fseek ( FILE *stream, long offset, int origin );
```

- 將 stream 的檔案位置指標，以 origin 為基準進行位移 offset 個位元。
- origin 可以為以下 3 種其中之一：
 - SEEK_SET：檔案開頭
 - SEEK_CUR：目前檔案位置指標
 - SEEK_END：檔案尾端
- fseek(fPtr, 0, SEEK_SET); 的效果如同 rewind(fPtr);

隨機地讀寫資料 (Reading and Writing Data Randomly)

- 例如我們可以控制檔案位置指標，以隨機地讀取在上一節中建立的 student.dat 中，任一個學生的資訊，而不用循序地讀取。
- 以下為 student.dat 的內容與每組紀錄的結構：

```
1  #include <stdio.h>
2  #include <assert.h>
3
4  typedef struct {
5      char name[32];
6      int sid;
7      double score;
8  } Student;
9
```

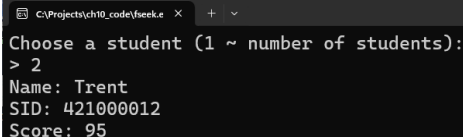
≡ student.dat

⚙	00 01 02 03 04 05 06 07	Decoded Text
00000000	4B 65 76 69 6E 00 00 00	K e v i n . . .
00000008	00 00 00 00 00 00 00 00
00000010	00 00 00 00 00 00 00 00
00000018	00 00 00 00 00 00 00 00
00000020	CA 5C 7F 18 5A 00 00 00	. \ . . Z . . .
00000028	4A 75 73 74 69 6E 00 00	J u s t i n . .
00000030	00 00 00 00 00 00 00 00
00000038	00 00 00 00 00 00 00 00
00000040	00 00 00 00 00 00 00 00
00000048	CC 5C 7F 18 61 00 00 00	. \ . . a . . .
00000050	4A 69 6D 6D 79 00 00 00	J i m m y . . .
00000058	00 00 00 00 00 00 00 00
00000060	00 00 00 00 00 00 00 00
00000068	00 00 00 00 00 00 00 00
00000070	CE 5C 7F 18 41 00 00 00	. \ . . A . . .

隨機地讀寫資料 (Reading and Writing Data Randomly)

- 使用 `fseek` 可以跳到任意檔案中的位置以讀取指定資料：

```
11 int main() {
12     int n;
13     Student tmp;
14     FILE *fptr = fopen("student.dat", "rb");
15     assert(fptr); // IF fptr == 0, the program will terminate
16     printf("Choose a student (1 ~ number of students):\n> ");
17     scanf("%d", &n);
18     fseek(fptr, (n - 1) * sizeof(Student), SEEK_SET);
19     fread(&tmp, sizeof(Student), 1, fptr);
20     printf("Name: %s\nSID: %d\nScore: %d\n",
21           tmp.name, tmp.sid, tmp.score);
22     fclose(fptr);
23 }
```



```
C:\Projects\ch10_code\fseek.e x + v
Choose a student (1 ~ number of students):
> 2
Name: Trent
SID: 421000012
Score: 95
```

Q & A