計算機程式設計

Computer Programming

File I/O

W17: Self-Learning

GitHub repo

Outline

- Introduction to File I/O
- **Operations for a text file**
  - File Buffering (with the standard I/O library)
  - Open a file from a disk directly (unbuffered; not using the standard I/O library)
- **Operations for a binary file**
  - File Buffering (with the standard I/O library)
  - Open a file from a disk directly (unbuffered; not using the standard I/O library)

[Introduction] File I/O

- I/O means “Input” and “Output”.
- A program can be designed to work with a file (like .txt or others).
  - File operations: Read / Write
- File I/O usually relies on the Functions in <stdio.h> to read or write data.

[Definition] Streams

- In C, the term “stream” means any source of input **to** any destination for output.
- Many programs obtain all their input from one stream (the keyboard) and write all their output to another stream (the screen), like scanf we use before.
- In these slides, we mainly use streams for file operations in a program.

[Introduction] Files usually used in C

hello_world.c

C code

- The code we write in C.

hello_world.exe

C executable file

- We will not see this file on Ideone.
- On unix systems, there won't be extensions in default.

data.txt

data file

- We are going to learn in this tutorial.

[Notes] Files for C Programming

- To learn how to perform operation to files in C, we cannot use Ideone.
  - We can only work with a single C script on Ideone.
- Therefore, we can start to use a local code editor, like Dev C++ or Visual Studio Code (depending on your favor).
- Visual Studio Code with Copilot is not recommended for beginners.
- For Assignment 5, you cannot use Ideone to finish the work because we need to work with a text file. This, you need to use a local code editor.

[Guide] Dev C++

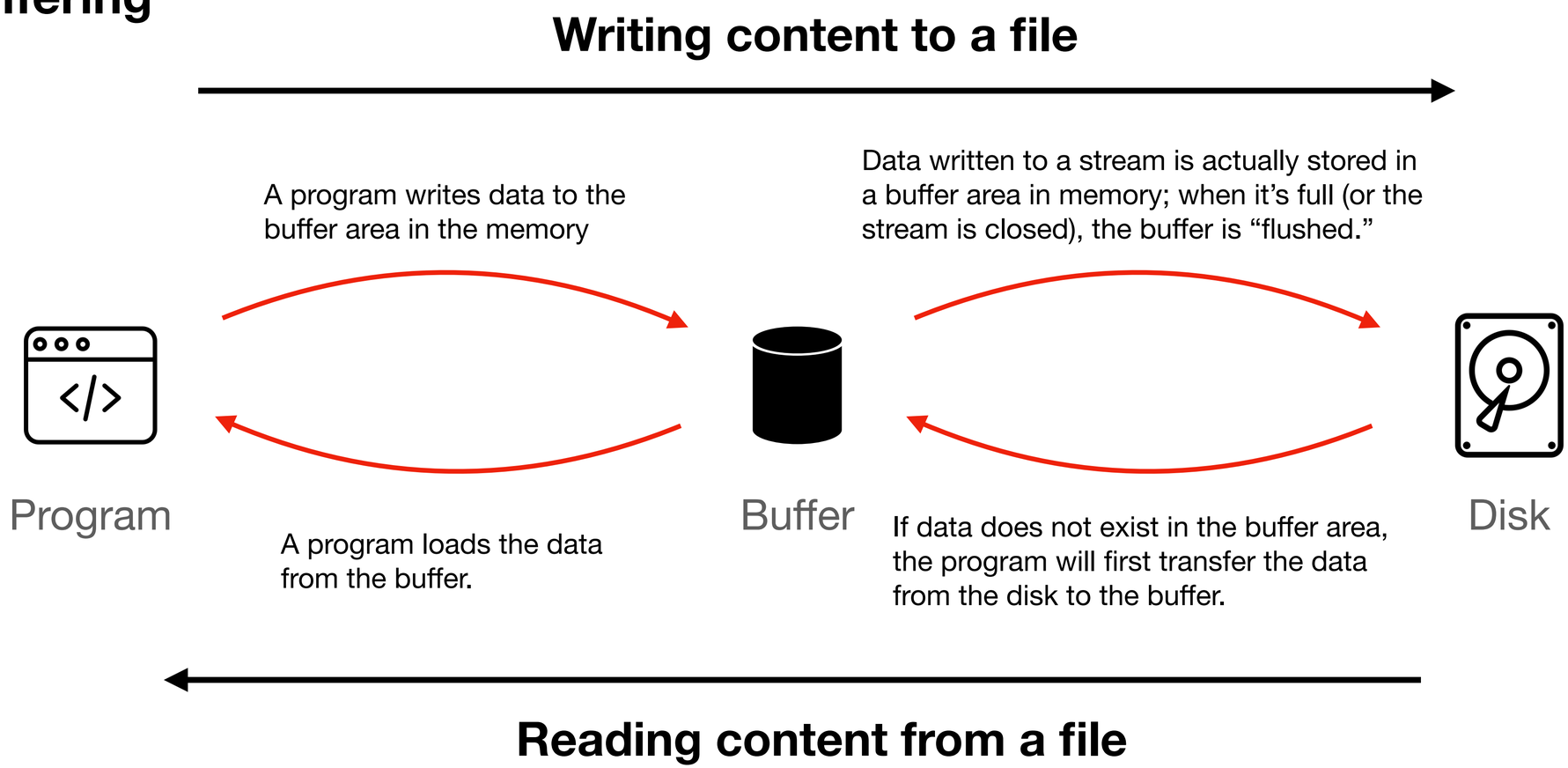
- (If you are familiar with Dev C++, you can skip this step.)
- Dev C++ Tutorials:
  - How to install (English)
    - https://www.youtube.com/watch?v=F9LcfFlDIJs
  - How to use (Chinese)
    - https://hackmd.io/@WTYang/BkTPMbEuQ?type=view

[Usage] Two types to interact with a file in C

- **File Buffering (文件緩衝區)**
  - The standard I/O library maintains an internal buffer in the memory that can store data from a file temporarily.
  - Usually Faster (File Buffering reduces the number of direct accesses to the disk, improving efficiency.)
- **Open a file from a disk directly (直接從硬碟操作檔案)**
  - Opening a file directly refers to using system calls (i.e., not using the standard I/O library). This does not automatically maintain a buffer area in the memory.
  - Usually Slower (Transferring data to or from a disk drive is a relatively slow operation.)

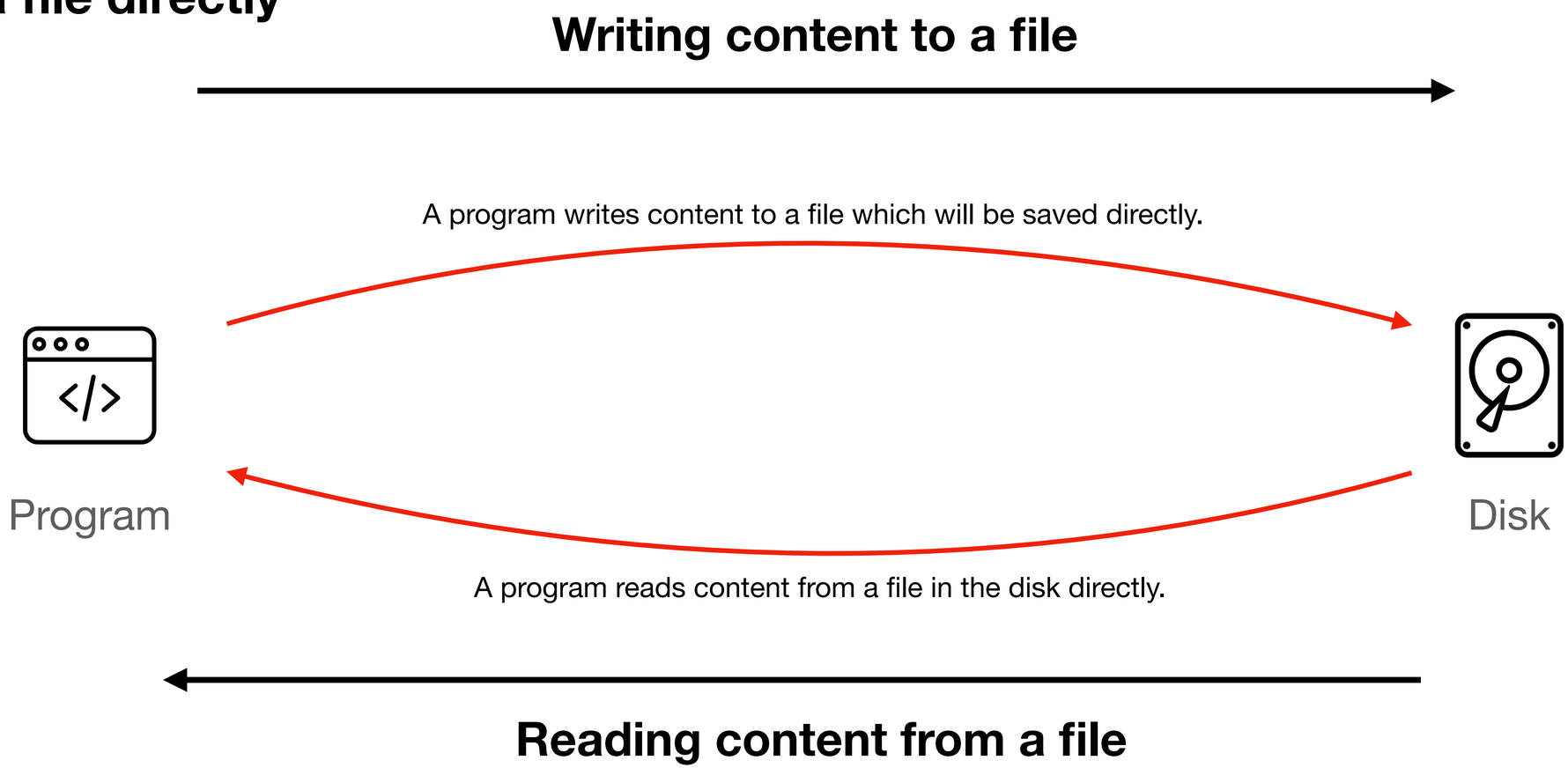
[Illustration] File Buffering

- **File Buffering**



[Illustration] Open a file from a disk directly

- **Open a file directly**



Common functions for file operations

注意輸入與輸出！

Purpose	Function name	Head file	Prototype	Return
有緩衝區				
開啟檔案	fopen	stdio.h	FILE * fopen (const char * filename, const char * mode);	File pointer
關閉檔案	fclose	stdio.h	int fclose (FILE * stream);	0 or EOF
讀取檔案	fread	stdio.h	size_t fread (void * ptr, size_t size, size_t count, FILE * stream);	Num of elements
寫入檔案	fwrite	stdio.h	size_t fwrite (const void * ptr, size_t size, size_t count, FILE * stream);	Num of elements
字元讀取	fgetc	stdio.h	int fgetc (FILE * stream);	Current character
字元寫入	fputc	stdio.h	int fputc (int character, FILE * stream);	Current character
無緩衝區				
開啟檔案	open	fcntl.h	int open (const char *pathname, int oflags, …);	File descriptor
關閉檔案	close	unistd.h	int close (int fd);	0 or -1
讀取檔案 字元讀取	read	unistd.h	size_t read (int fd, void *buffer, size_t count);	Number of bytes
寫入檔案 字元寫入	write	unistd.h	size_t write (int fd, const void *buffer, size_t count);	Number of bytes

File Buffering

(with the standard I/O library)

[Declaration] File Pointers

- Accessing a stream is done through a file pointer, which has type FILE *.

```
FILE *ptr_name; // Declaration of a pointer that points to a file
```

- Usually, file buffering is the easiest way to interact with a file in C code.
- To use file buffering, we have to declare a file pointer first, which is FILE *.

[Usage] Mode for fopen

Mode	“r”	“w”	“a”
Meaning	Read: Open file for input operations. The file must exist.	Write: Create an empty file for output operations.	Append: Open file for output at the end of a file. Output operations always write data at the end of the file, expanding it.

注意mode要使用雙引號!

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

宣告檔案指標，對File I/O來說，宣告檔案指標是必須的

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

以fopen建立文件和程式之間的資料交換管道(stream; 資料流)，且fopen會回傳檔案指標，讀檔案我們使用read模式

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

如果找不到檔案的話fopen會回傳NULL，
這時我們可以自行設立錯誤訊息

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

以fgetc逐次讀入character，直到檔案最底
fgetc會取到EOF，就停止

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

完成我們需要程式做的事情後，用fclose關閉stream

[Important Notes] Stream, fopen, fclose

- 執行fopen時就搭起了資料與程式之間的橋樑 (stream)，此時程式會自動開啟一段位於記憶體的緩衝區 (buffer)，所有對於檔案的改動都會先存到緩衝區
- 執行fclose後，資料與程式之間的橋樑 (stream) 就關閉了，此時程式會將對於檔案的改動寫回硬碟中的目標檔案
- 如果沒有執行fclose，可能會導致記憶體洩漏、資料損毀或文件開啟衝突等問題，因此在對檔案的操作完成後，寫一行fclose是一個良好習慣

Count the number of characters in a file

C-course-materials/10-FileIO/buffer_txt_count.c
C-course-materials/10-FileIO/buffer_txt_count_fscanf.c

那可不可以使用scanf呢？(左程式碼同上一頁)

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while ((ch = fgetc(fptr)) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *fptr;
    char ch;
    int count = 0;

    fptr = fopen("hello.txt", "r");
    if (fptr != NULL){
        while (fscanf(fptr, "%c", &ch) != EOF){
            printf("%c", ch);
            count++;
        }
        fclose(fptr);
        printf("\nNum of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

- fgetc和fscanf都需要檔案指標作為輸入，但fscanf還需要format specifier以及欲賦予值的變數位置
- 我們不能用一般的scanf，因為一般的scanf是接收鍵盤輸入的值，但fscanf是從檔案接收值

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/buffer_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    FILE *fptr1, *fptr2;
    char ch;
    int count = 0;

    fptr1 = fopen("hello.txt", "r");
    fptr2 = fopen("hello_copy.txt", "w");
    if (fptr1 != NULL && fptr2 != NULL){
        while ((ch = fgetc(fptr1)) != EOF){
            fputc(ch, fptr2);
            count++;
        }
        fclose(fptr1);
        fclose(fptr2);
        printf("Num of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

以read模型開啟A檔案，並以write模式開啟B檔案，因為我們要寫入內容至B檔案。此時因為是write模式，如果B檔案不存在，程式會幫我們創建B檔案

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/buffer_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    FILE *fptr1, *fptr2;
    char ch;
    int count = 0;

    fptr1 = fopen("hello.txt", "r");
    fptr2 = fopen("hello_copy.txt", "w");
    if (fptr1 != NULL && fptr2 != NULL){
        while ((ch = fgetc(fptr1)) != EOF){
            fputc(ch, fptr2);
            count++;
        }
        fclose(fptr1);
        fclose(fptr2);
        printf("Num of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

fptr2 == NULL 時可能代表無法建立B檔案，可能是路徑錯誤或是程式所在的檔案位置權限不足

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/buffer_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    FILE *fptr1, *fptr2;
    char ch;
    int count = 0;

    fptr1 = fopen("hello.txt", "r");
    fptr2 = fopen("hello_copy.txt", "w");
    if (fptr1 != NULL && fptr2 != NULL){
        while ((ch = fgetc(fptr1)) != EOF){
            fputc(ch, fptr2);
            count++;
        }
        fclose(fptr1);
        fclose(fptr2);
        printf("Num of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

利用fputc逐一寫入字元到fptr2

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/buffer_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    FILE *fptr1, *fptr2;
    char ch;
    int count = 0;

    fptr1 = fopen("hello.txt", "r");
    fptr2 = fopen("hello_copy.txt", "w");
    if (fptr1 != NULL && fptr2 != NULL){
        while ((ch = fgetc(fptr1)) != EOF){
            fputc(ch, fptr2);
            count++;
        }
        fclose(fptr1);
        fclose(fptr2);
        printf("Num of characters: %d\n", count);
    }
    else{
        printf("Failed to open file\n");
    }
}
```

操作完成後，關閉兩個檔案的streams

Append content to a file

此範例展示如何在鍵盤中輸入字串後寫入檔案中

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

int main(void){
    FILE *fptr;
    char str[MAX], ch;
    int i = 0;
    fptr = fopen("hello.txt", "a");
    putc('\n', fptr);

    printf("Please enter a string: ");
    fgets(str, MAX, stdin);
    fwrite(str, sizeof(char), strlen(str), fptr);
    fclose(fptr);
    printf("Data written to file\n");
}
```

宣告array時指定長度時，長度本身需要是constant，不能是variable
我們可以用#define來建立一個global constant

Append content to a file

此範例展示如何在鍵盤中輸入字串後寫入檔案中

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

int main(void){
    FILE *fptr;
    char str[MAX], ch;
    int i = 0;
    fptr = fopen("hello.txt", "a");
    putc('\n', fptr);

    printf("Please enter a string: ");
    fgets(str, MAX, stdin);
    fwrite(str, sizeof(char), strlen(str), fptr);
    fclose(fptr);
    printf("Data written to file\n");
}
```

append 模式不會覆蓋原檔案內容

Append content to a file

此範例展示如何在鍵盤中輸入字串後寫入檔案中

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

int main(void){
    FILE *fptr;
    char str[MAX], ch;
    int i = 0;
    fptr = fopen("hello.txt", "a");
    putc('\n', fptr);

    printf("Please enter a string: ");
    fgets(str, MAX, stdin);
    fwrite(str, sizeof(char), strlen(str), fptr);
    fclose(fptr);
    printf("Data written to file\n");
}
```

利用fgets取得鍵盤輸入的內容

Append content to a file

此範例展示如何在鍵盤中輸入字串後寫入檔案中

C-course-materials/10-FileIO/buffer_txt_count.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

int main(void){
    FILE *fptr;
    char str[MAX], ch;
    int i = 0;
    fptr = fopen("hello.txt", "a");
    putc('\n', fptr);

    printf("Please enter a string: ");
    fgets(str, MAX, stdin);
    fwrite(str, sizeof(char), strlen(str), fptr);
    fclose(fptr);
    printf("Data written to file\n");
}
```

點我回去看fwrite的函數原形

將整段string寫入到檔案中

Read content from a file to the program (1/2)

C-course-materials/10-FileIO/buffer_txt_read.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

void print_str(char str1[], int size){
    for (int i = 0; i < size; i++){
        if (str1[i] == '\0'){
            printf("\\0");
        }
        else{
            printf("%c", str1[i]);
        }
    }
}
```

[Recap] We can write a function to print a string.

Read content from a file to the program (2/2)

C-course-materials/10-FileIO/buffer_txt_read.c

```c
int main(void){
    FILE *fptr;
    char str[MAX];
    int num_of_chars = 0;
    fptr = fopen("hello.txt", "r");
    size_t bytes_read;
        如果有成功讀到檔案的話，fread會回傳大於零的整數，所以我們可以用來判斷檔案是否已經讀完
    while ((bytes_read = fread(str, sizeof(char), MAX, fptr)) > 0){
        num_of_chars += bytes_read;
        if (num_of_chars < MAX){
            str[num_of_chars] = '\0';
        }
        // printf("%s", str);
        print_str(str, num_of_chars+1);
    }
    fclose(fptr);
}
```

Read content from a file to the program (2/2)

C-course-materials/10-FileIO/buffer_txt_read.c

```c
int main(void){
    FILE *fptr;
    char str[MAX];
    int num_of_chars = 0;
    fptr = fopen("hello.txt", "r");
    size_t bytes_read;

    while ((bytes_read = fread(str, sizeof(char), MAX, fptr)) > 0){
        num_of_chars += bytes_read;
        if (num_of_chars < MAX){
            str[num_of_chars] = '\0'; fread不會幫我們在最後面加上\0，但我們可以自己加
        }
        // printf("%s", str);
        print_str(str, num_of_chars+1);
    }
    fclose(fptr);
}
```

Read content from a file to the program (2/2)

C-course-materials/10-FileIO/buffer_txt_read.c

```c
int main(void){
    FILE *fptr;
    char str[MAX];
    int num_of_chars = 0;
    fptr = fopen("hello.txt", "r");
    size_t bytes_read;

    while ((bytes_read = fread(str, sizeof(char), MAX, fptr)) > 0){
        num_of_chars += bytes_read;
        if (num_of_chars < MAX){
            str[num_of_chars] = '\0';
        }
        // printf("%s", str); 我們可以直接用%s印出str，但不會看到\0
        print_str(str, num_of_chars+1);
    }
    fclose(fptr);
}
```

Read content from a file to the program (2/2)

C-course-materials/10-FileIO/buffer_txt_read.c

```c
int main(void){
    FILE *fptr;
    char str[MAX];
    int num_of_chars = 0;
    fptr = fopen("hello.txt", "r");
    size_t bytes_read;

    while ((bytes_read = fread(str, sizeof(char), MAX, fptr)) > 0){
        num_of_chars += bytes_read;
        if (num_of_chars < MAX){
            str[num_of_chars] = '\0';
        }
        // printf("%s", str);
        print_str(str, num_of_chars+1);
    }
    fclose(fptr);
}
```

想觀察\0的話，長度需要再加上1，
因為我們上面加了\0在str最後面

Open a file from a disk directly

[Important Notes] Open a file from a disk directly

- Opening a file from a disk directly is an unbuffered method.
  - The program does not automatically allocate a memory for file buffering.
- This method does not belong to file I/O because it does not create a stream. Instead, a program works directly to the file.
- This method requires lower-level system calls rather than the higher-level functions defined in stdio.h.

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/unbuffered_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define SIZE 100

int main(void){
    char buffer[SIZE];
    int f1, f2, bytes;
    f1 = open("hello.txt", O_RDONLY); // Linux
    f2 = open("hello_unbuffered.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    if (f1 != -1 || f2 != -1){
        while ((bytes = read(f1, buffer, SIZE)) > 0){
            write(f2, buffer, bytes);
        }
        close(f1);
        close(f2);
        printf("File copied successfully\n");
    }
    else{
        printf("Error in opening file\n");
    }
}
```

開啟檔案時的操作模式，若要兼容多種模式的話要加上|

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/unbuffered_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define SIZE 100

int main(void){
    char buffer[SIZE];
    int f1, f2, bytes;
    f1 = open("hello.txt", O_RDONLY); // Linux
    f2 = open("hello_unbuffered.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    if (f1 != -1 || f2 != -1){
        while ((bytes = read(f1, buffer, SIZE)) > 0){
            write(f2, buffer, bytes);
        }
        close(f1);
        close(f2);
        printf("File copied successfully\n");
    }
    else{
        printf("Error in opening file\n");
    }
}
```

儲存檔案時的操作模式，若要兼容多種模式的話要加上|

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/unbuffered_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define SIZE 100

int main(void){
    char buffer[SIZE];
    int f1, f2, bytes;
    f1 = open("hello.txt", O_RDONLY); // Linux
    f2 = open("hello_unbuffered.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    if (f1 != -1 || f2 != -1){
        while ((bytes = read(f1, buffer, SIZE)) > 0){
            write(f2, buffer, bytes);
        }
        close(f1);
        close(f2);
        printf("File copied successfully\n");
    }
    else{
        printf("Error in opening file\n");
    }
}
```

O_CREAT: 如果檔案不存在，則建立新檔案，若檔案存在的話則不需要此標籤

O_TRUNC: 如果檔案存在，則檔案原本的內容都會全部被清除

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/unbuffered_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define SIZE 100

int main(void){
    char buffer[SIZE];
    int f1, f2, bytes;
    f1 = open("hello.txt", O_RDONLY); // Linux
    f2 = open("hello_unbuffered.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    if (f1 != -1 || f2 != -1){
        while ((bytes = read(f1, buffer, SIZE)) > 0){
            write(f2, buffer, bytes);
        }
        close(f1);
        close(f2);
        printf("File copied successfully\n");
    }
    else{
        printf("Error in opening file\n");
    }
}
```

-1代表無法開啟檔案，值得注意的是不用預先宣告檔案指標

Copy the content of a file to another

此範例展示如何將A檔案的內容複製到B檔案

C-course-materials/10-FileIO/unbuffered_txt_copy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define SIZE 100

int main(void){
    char buffer[SIZE];
    int f1, f2, bytes;
    f1 = open("hello.txt", O_RDONLY); // Linux
    f2 = open("hello_unbuffered.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    if (f1 != -1 || f2 != -1){
        while ((bytes = read(f1, buffer, SIZE)) > 0){
            write(f2, buffer, bytes);
        }
        close(f1);
        close(f2);
        printf("File copied successfully\n");
    }
    else{
        printf("Error in opening file\n");
    }
}
```

寫入方法與file buffering雷同

點我回去看write的函數原形

[Usage] oflags

- Mode for file operations

```
int open ( const char *pathname, int oflags, … );
```

oflags	Meaning	oflags	Meaning
O_RDONLY	Read only for the opened file	S_IWRITE	Read only for the file to be saved
O_WRONLY	Write only for the opened file	S_IREAD	Write only for the file to be saved
O_RDWR	Readable and writable for the opened file	S_IREAD | S_IWRITE	Readable and writable for the file to be saved

Mode for opening a file

Mode for saving a (new) file

Binary File Operation (Buffered)

[Introduction] Text Files versus Binary Files

- When data is written to a file, it can be stored in text form or in binary form.
- For example, one way to store the number 32767 in a file would be to write it in text form as the characters 3, 2, 7, 6, and 7:
  - ‘3’: 0011 0011
  - ‘2’: 0011 0010
  - ‘7’: 0011 0111
  - ‘6’: 0011 0110
  - ‘7’: 0011 0111

Total: 40 bytes

Another way is to store 32767 **in binary.** We can store its binary value as 01111111 11111111, which only takes 16 bytes.

Thus, storing in binary can often save space.

Write content to a binary file (1/2)

點我回去看fwrite的函數原形

C-course-materials/10-FileIO/buffer_binary_write.c

此範例展示寫入數值到 binary file

```c
#include <stdio.h>
#include <stdlib.h>

// write binary function
void write_binary(const char *filename, int numbers[], int size) {
    FILE *writeFile = fopen(filename, "wb"); 模式使用 write binary
    if (writeFile == NULL) {
        printf("無法開啟檔案進行寫入");
    }
    if (fwrite(numbers, sizeof(int), size, writeFile) != size) {
        printf("寫入檔案時發生錯誤");
        fclose(writeFile);
    }
    fclose(writeFile);
    printf("成功將整數陣列寫入檔案 '%s'\n", filename);
}
```

Write content to a binary file (2/2)

C-course-materials/10-FileIO/buffer_binary_write.c

```c
int main(void) {
    const char *filename1 = "file1.bin";
    int numbers1[] = {90, 70, 30, 100, 50};
    int size1 = sizeof(numbers1) / sizeof(numbers1[0]);

    write_binary(filename1, numbers1, size1);
}
```

Read content from a binary file

此範例展示從 binary file 讀取資料

C-course-materials/10-FileIO/buffer_binary_read.c

點我回去看fread的函數原形

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int main(void) {
    const char *filename1 = "file1.bin";
    int numbers1[MAX];
    int size1;

    // read files to the arrays
    FILE *readFile1 = fopen(filename1, "rb"); 模式使用 read binary
    size1 = fread(numbers1, sizeof(int), MAX, readFile1);
    for (int i = 0; i < size1; i++) {
        printf("%d ", numbers1[i]);
    }
}
```

Binary File Operation (Unbuffered)

Write content to a binary file (1/2)

C-course-materials/10-FileIO/unbuffered_binary_write.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

// write binary function
void write_binary(const char *filename, int numbers[], int size) {
    int fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (fd == -1) {
        perror("無法開啟檔案進行寫入");
    }

    size_t bytes_written = write(fd, numbers, size * sizeof(int));
    if (bytes_written != size * sizeof(int)) {
        perror("寫入檔案時發生錯誤");
        close(fd);
    }

    close(fd); // 關閉檔案
    printf("成功將整數陣列寫入檔案 '%s'\n", filename);
}
```

fd (file descriptor): 類似於file buffering的檔案指標

使用perror可以自動顯示出系統錯誤訊息

Write content to a binary file (2/2)

C-course-materials/10-FileIO/unbuffered_binary_write.c

```c
int main(void) {
    const char *filename1 = "file1.bin";
    int numbers1[] = {90, 70, 30, 100, 50};
    int size1 = sizeof(numbers1) / sizeof(numbers1[0]);

    write_binary(filename1, numbers1, size1);
}
```

Read content from a binary file

C-course-materials/10-FileIO/unbuffered_binary_read.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#define MAX 10

int main(void) {
    const char *filename1 = "file1.bin";
    int numbers1[MAX], bytes;

    int fd = open(filename1, O_RDONLY); // file descriptor
    if (fd == -1) {
        perror("無法開啟檔案進行讀取");
    }

    bytes = read(fd, numbers1, sizeof(numbers1));
    if (bytes == -1) {
        perror("讀取檔案時發生錯誤");
        close(fd);
    }

    bytes /= sizeof(int); // actual number of elements read
    for (int i = 0; i < bytes; i++) {
        printf("%d ", numbers1[i]);
    }
    close(fd);
}
```

類似於file buffering的檔案指標